ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ГОСТ Р ИСО/HL7 27951— 2016**

# Информатизация здоровья

# ОБЩИЕ ТЕРМИНОЛОГИЧЕСКИЕ СЛУЖБЫ

## Выпуск 1

**(ISO/HL7 27951:2009 Health informatics — Common terminology services, release 1, IDT)**

Издание официальное

Москва
Стандартинформ
2017

## Предисловие

1 ПОДГОТОВЛЕН Федеральным государственным бюджетным учреждением «Центральный научно-исследовательский институт организации и информатизации здравоохранения Министерства здравоохранения Российской Федерации» (ЦНИИОИЗ Минздрава) на основе собственного перевода на русский язык англоязычной версии международного стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 468 «Информатизация здоровья» при ЦНИИОИЗ Минздрава — постоянным представителем ISO/TC 215

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 30 ноября 2016 г. № 1894-ст

4 Настоящий стандарт идентичен международному стандарту ISO/HL7 27951:2009 «Информатизация здоровья. Общие терминологические службы. Выпуск 1» (ISO/HL7 27951:2009 «Health informatics — Common terminology services, release 1», IDT).

Наименование настоящего стандарта изменено относительно наименования указанного международного стандарта для приведения в соответствие с ГОСТ Р 1.5 (подраздел 3.6)

5 ВВЕДЕН ВПЕРВЫЕ

*Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ежемесячном информационном указателе «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.gost.ru)*

© Стандартинформ, 2017

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

## Содержание

1 Область применения. Общие терминологические службы (ОТС) . . . 1
2 Нормативные ссылки . . . 3
3 Термины и определения . . . 3
4 Условные обозначения и сокращения . . . 4
5 Модули ОТС . . . 5
5.1 API сообщений и словаря . . . 5
5.2 Функции времени исполнения и обозревания . . . 6
6 Краткий обзор функций модулей . . . 7
6.1 Функции платформы времени исполнения, предназначенной для уровня сообщений . . . 7
6.2 Функции платформы времени исполнения, предназначенной для уровня словаря . . . 9
6.3 Функции отображения кодов . . . 10
6.4 Функции обозревателя уровня сообщений . . . 11
6.5 Функции обозревателя уровня словаря . . . 13
7 Модель API сообщений ОТС . . . 14
7.1 Введение . . . 14
7.2 Словарный домен . . . 15
7.3 Набор значений . . . 16
7.4 Зарегистрированная система кодирования . . . 20
8 Спецификация API сообщений ОТС . . . 21
8.1 Введение . . . 21
8.2 Общие элементы сообщений ОТС . . . 21
8.3 API времени исполнения уровня сообщений . . . 25
8.4 API обозревателя на уровне сообщений . . . 33
9 Модель API словаря . . . 38
9.1 Введение . . . 38
9.2 Система кодирования . . . 38
9.3 Кодированное понятие CodedConcept . . . 39
9.4 Обозначение понятия ConceptDesignation . . . 40
9.5 Свойство понятия ConceptProperty . . . 40
9.6 Отношение понятий ConceptRelationship . . . 41
10 Спецификация API словаря . . . 41
10.1 Введение . . . 41
10.2 Базовые типы данных . . . 41
10.3 API времени исполнения на уровне словаря . . . 43
10.4 API обозревателя на уровне словаря . . . 46
11 Модель отображения кодов . . . 55
11.1 Введение . . . 55
11.2 Отображение кодов . . . 55
11.3 Класс MapEntry . . . 56
12 Спецификация отображения кодов . . . 56
12.1 Введение . . . 56
12.2 Идентификация службы отображения . . . 56
12.3 Отображение кода . . . 56

12.4 Исключения . . . 56
12.5 Отображение кода понятия . . . 57
13 Привязка ОТС к языкам программирования . . . 57
13.1 Преобразование спецификации на языке IDL в программный код на языке Java . . . 57
13.2 Преобразование спецификации на языке IDL в описание на языке WSDL . . . 62
14 Сводные требования к службам . . . 67
14.1 Введение . . . 67
14.2 Информационные службы сервера . . . 67
14.3 Службы, предоставляющие метаданные терминологий . . . 67
14.4 Службы, предоставляющие доступ к содержанию терминологии . . . 68
15 Системы кодирования, используемые в API ОТС . . . 72
16 Спецификация API ОТС на языке IDL . . . 74
16.1 Базовые типы данных HL7 . . . 74
16.2 Определение API сообщений ОТС на языке IDL . . . 76
16.3 Определение API словаря ОТС на языке IDL . . . 98
17 Описание API ОТС на языке WSDL . . . 119
17.1 Описание API времени исполнения на уровне сообщений . . . 119
17.2 Описание API обозревателя на уровне сообщений . . . 129
17.3 Описание API времени исполнения на уровне словаря . . . 148
17.4 Описание API обозревателя на уровне словаря . . . 154
17.5 Описание API отображения кодов . . . 170
Приложение А (справочное) Эталонная информационная модель HL7 . . . 178
Приложение В (справочное) Типы данных. Абстрактная спецификация . . . 280
Приложение С (справочное) Словарные домены HL7 . . . 421
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов национальным стандартам . . . 430
Библиография . . . 430

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Информатизация здоровья**

# ОБЩИЕ ТЕРМИНОЛОГИЧЕСКИЕ СЛУЖБЫ

**Выпуск 1**

Health informatics. Common terminology services. Release 1

**Дата введения — 2018—01—01**

## 1 Область применения. Общие терминологические службы (ОТС)

Содержание настоящего стандарта должно стать основной международной платформы разработки интерфейса прикладных программ API (Application Programming Interface), который может использоваться программным обеспечением обработки сообщений для доступа к терминологическому содержанию. Сам по себе такой интерфейс не обеспечит полную терминологическую службу. Стандарты Health Level Seven (HL7) Версии 3 (использующие язык XML) основаны на эталонной информационной модели RIM (Reference Information Model), обладающей общей и гибкой структурой. Представление информации в этой модели опирается на доступность терминологических ресурсов, которые могут использоваться для наполнения свойств модели требуемым семантическим содержанием. По возможности стандарт HL7 Версии 3 использует существующие терминологические ресурсы, а не создает новые внутри себя.

Поскольку внешние терминологические ресурсы могут значительно различаться как по содержанию, так и по структуре, ОТС идентифицируют общие функциональные характеристики, которые может предоставлять внешняя терминология. Например, терминологическая служба, совместимая со стандартом HL7, должна быть способной определить, является ли данный код понятия действительным в конкретном ресурсе. Вместо описания таблицы, строки которой содержат идентификаторы ресурса и коды понятий, спецификация ОТС описывает вызов интерфейса прикладных программ API (Application Programming Interface), который может на входе получить идентификатор ресурса и код понятия, а на выходе возвратить значение true (истинный) или false (ложный). Каждый разработчик терминологии свободен реализовать этот вызов API по собственному усмотрению.

Спецификация ОТС не рассчитана на нижеследующее:

- текущая версия не предназначена к использованию в качестве полной терминологической службы. Ее область применения ограничена функциональностью, необходимой для конструирования, реализации и развертывания пакетов программ, соответствующих стандарту HL7 Версии 3. В той же степени, как язык XML связан с языком SGML, ОТС HL7 рассчитаны на представление достаточного подмножества функций, которые могут быть предоставлены более сложными API, например, теми, что соответствуют спецификации TQS организации OMG;
- она не предназначена для использования в качестве языка запросов общего назначения. В ней определены только конкретные службы, необходимые для реализации стандарта HL7;
- в ней не задан способ реализации службы. В ней намеренно опущены требования к информированию о службе и к ее обнаружению, к установке соединений и к управлению ими, а также требования

к доставке и маршрутизации сообщений. Предполагается, что реализация ОТС будет использовать нижележащую архитектуру, наиболее подходящую для конкретных условий реализации.

Разработка спецификации ОТС HL7 основана на следующих общих принципах:

1) должно быть нетрудно написать программу, использующую ОТС HL7;

2) ОТС HL7 предназначены для описания только базовых требований;

3) конструкция ОТС HL7 должна быть формальной и точной;

4) первичная технология реализации ОТС HL7 должна быть основана на языке XML;

5) ОТС HL7 должны быть совместимы с номенклатурой, моделью и подходом, представленными в документе HL7 Vocabulary, модели RIM стандарта HL7 Версии 3 и производными структурами,

6) по возможности ОТС HL7 должны оставаться согласованным подмножеством Служб терминологических запросов TQS (Terminology Query Services) организации Object Management Group (OMG), пока это не противоречит другим принципам конструирования ОТС HL7. Если будет обнаружено, что модель TQS противоречит принципам конструирования ОТС HL7 или является неполной либо некорректной, то должны быть предприняты все необходимые шаги, чтобы уведомить об этом соответствующую рабочую группу по пересмотру;

7) ОТС HL7 должны ограничиться теми предположения о форме и структуре терминологии, которая необходима для поддержки реализации стандартов HL7.

Хотя общераспространенного стандарта терминологических служб не существует, на эту тему есть несколько источников материала:

**- спецификация OMG Terminology Query Services (TQS).**

Спецификация TQS описывает полную терминологическую службу, но она широко не реализована и ее поддержка со стороны производителей программного обеспечения минимальна. Некоторые считают, что она слишком «тяжеловесна» и при этом опирается на конкретное техническое решение (CORBA). Поскольку стандарт HL7 не предполагает опираться на стандарт TQS, то необходим более общий подход к терминологическим службам, по крайней мере в тех областях, где стандарт HL7 зависит от терминологии;

**- DAML + OIL и язык Web Ontology Language (OWL).**

Эти документы не являются спецификациями терминологического сервера, но при этом они содержат элементы представления онтологических аспектов, релевантные некоторым масштабным терминологиям, например, SNOMED Clinical Terms, NHS Clinical Terms Version 3 и GALEN. Однако это предложение, основанное на веб-технологиях, также является тяжеловесным и вряд ли приведет к ранней широко распространенной реализации;

**- спецификации API, рассчитанные на конкретные терминологии.**

Примером может служить спецификация API «Read Code — Version 3», разработанная по проекту клинической терминологии NHS Clinical Terms в 1996 году и пересмотренная в 1998 году. На основе похожих принципов ведется работа по созданию API SNOMED CT. Неофициально признается, что этот конкретный API сольется с ОТС или будет использовать подходящие элементы ОТС там, где это целесообразно. Реализация этого API на основе технологии COM обеспечивается по меньшей мере одной свободно распространяемой машиной кодирования (CLUE). Спецификации этого типа идентифицируют многие общие функции, необходимые для доступа к терминологии. Однако они неизбежно специфичны для нужд конкретной терминологии. Явная поддержка единственной определенной модели терминологии позволяет обеспечить ее эффективную реализацию в операционной среде в ущерб гибкости, требуемой для доступа к другим терминологиям;

**- интерфейсы к системам реляционных баз данных, включая SQL и ODBC.**

Для «элементарных» списков кодов простой запрос на языке SQL может оказаться наиболее эффективным способом извлечения кода. Однако многие схемы кодирования в дополнение к парам «значение кода» — «описание значения» имеют другие релевантные свойства, которые могут быть доступными только с помощью вторичной службы. Это не препятствует применению языка SQL, но требует описания общей модели, в соответствии с которой могут исполняться запросы, и эффективных средств, возвращающих требуемые свойства. Такие дополнительные свойства присущи как всей схеме, так и отдельным элементам терминологии;

**- язык терминологических запросов TQL (Terminology Query Language).**

Язык TQL, ранее основанный на синтаксисе, подобном языку SQL, в настоящее время реализуется на основе унифицированных идентификаторов ресурсов URI, специфичных для терминологических серверов, разработанных Майклом Хогартом (Michael Hogarth) и его коллегами в Калифорнийском университете. В языке TQL предусмотрен богатый механизм, предназначенный для оперирования

общими свойствами и отношениями в терминологических моделях. Работающая реализация языка TQL на платформе Java может быть бесплатно загружена из сети Интернет.

## 2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие документы (для датированных ссылок следует использовать только указанное издание, для недатированных — последнее издание указанного документа, включая все поправки):

ISO/HL7 21731:2006, Health informatics — HL7 Version 3 — Reference information model — release 1 (Информатизация здоровья. HL7, Версия 3. Эталонная информационная модель. Выпуск 1)[1)]

HL7 2008, Datatype Specification. Abstract Data Types (Спецификация типов данных. Абстрактные типы данных)

## 3 Термины и определения

Следует обратить внимание, что существует большое число терминов, используемых для описания базовых понятий в здравоохранении, приведенных в публикациях организаций ИСО, CEN, HL7 и других международных и национальных организаций для разных целей. Термины и определения, приведенные в настоящем стандарте, не претендуют на замену ранее данных. Они предназначены для совместного использования с национальными или региональными требованиями, и в случае конфликта национальные/региональные требования должны иметь приоритет. Для целей настоящего стандарта к терминологическим ресурсам, используемым в среде электронной передачи сообщений, применяются приведенные ниже термины и определения. Их выбор обусловлен задачей доступа к терминологическим ресурсам в среде электронной передачи сообщений.

3.1 **интерфейс прикладных программ** (application programming interface, API): Совокупность функций, процедур, методов или классов, предоставляемых операционной системой, библиотекой или службой для использования во внешних программных продуктах.

3.2 **система кодирования** (code system): В целях настоящего стандарта под «системой кодирования» понимается система кодов, обозначений, свойств и отношений. К другим общим именам этой сущности относятся «словарь», «терминология», «схема кодирования», «схема классификации» и «онтология».

3.3 **ограничение** (constraint): Логическое выражение, накладывающее ограничение на значения объектов, принадлежащих некоторому множеству, и тем самым задающее логическое подмножество этого множества.

3.4

**язык описания интерфейсов** (interface description language, IDL): Язык описания интерфейсов используется для описания интерфейса компонента программного обеспечения. Языки IDL описывают интерфейсы способом, не зависящим от языка программирования, чтобы компоненты программного обеспечения, написанные на разных языках, могли обмениваться данными.

[OMG; 2002 [1]]

3.5 **идентификатор объекта** (object identifier): Число, постоянное, присвоенное объекту и используемое для его уникальной идентификации в коллекции объектов.

3.6 **эталонная информационная модель** (reference information model, RIM): Информационная модель, разработанная комитетом HL7, используемая для построения всех других информационных моделей, например, уточненных информационных моделей сообщений RMIM и самих сообщений.

3.7 **уточненная информационная модель сообщений** (refined message information model, RMIM): Структура информации, используемая в стандарте передачи сообщений HL7 Версии 3, описывающая требования к комплексу сообщений.

3.8 **стандарт** (standard): Техническая спецификация требований к процессам деятельности, реализованная в конкурентоспособной промышленной продукции и в необходимых случаях признаваемая официальными органами стандартизации, например, ИСО.

---

[1)] В настоящее время действует версия ISO/HL7 21731:2014.

3.9

**терминология** (terminology): Любое организованное множество кодов, включая сущности, обычно называемые «наборами кодов», «онтологиями», «словарями», «системами классификации» и т. д.

[Европейская директива 2001/20/EC, статья 2e]

3.10

**язык описания веб-служб** (Web services description language, WSDL): WSDL представляет собой формат описания сетевых служб на языке XML, представляющего эти службы как совокупность конечных точек, оперирующих сообщениями, содержащими информацию, ориентированную на документы или процедуры. Операциям и сообщениям дается абстрактное описание; для определения конечной точки они привязываются к конкретному сетевому протоколу и формату сообщений. Конкретные логически связанные конечные точки объединяются в абстрактные конечные точки (службы). Для целей описания конечных точек и их сообщений независимо от форматов сообщений или сетевых протоколов, используемых для обмена данными, язык WSDL является расширяемым. При этом описание того, как использовать конкретную конечную точку в сочетании с SOAP 1.1, HTTP GET/POST и MIME, задается исключительно в привязках.

[W3C; 2001 [2]]

## 4 Условные обозначения и сокращения

В настоящем стандарте используются следующие сокращения:

API — интерфейс прикладных программ (Application Programming Interface);

OTC — общая терминологическая служба (Common Terminology Service, CTS);

CEN — Европейский комитет по стандартизации, федерация из 28 национальных органов стандартизации, которые также входят в ИСО (Comité Européen de Normalisation);

EU — Европейский союз (European Union);

HL7 — Health Level Seven Inc.;

HMD — иерархическое описание сообщения (Hierarchical Message Description);

IDL — язык описания интерфейсов (Interface Description Language);

ITS — спецификация реализуемой технологии (Implementable Technology Specification);

ICH — Международная конференция по гармонизации технических требований к регистрации лекарственных препаратов для человека (The International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use);

ИСО — International Organization for Standardization, ISO;

ОИДо — объектный идентификатор (Object IDentifier);

RIM — эталонная информационная модель (Reference Information Model);

RMIM — уточненная информационная модель сообщений (Refined Message Information Model);

SDO — Организация по разработке стандартов (Standards Development Organization);

SGML — стандартный обобщенный язык разметки (standardized generalized markup language). Стандарт ИСО по платформенно-независимому описанию структурированной информации;

SNOMED — систематизированная номенклатура терминов медицины и ветеринарии (The Systematized Nomenclature of Human and Veterinary Medicine);

SNOMED-CT — систематизированная номенклатура терминов клинической медицины (Systematized Nomenclature of Medicine-Clinical Terms Medicine);

UML — унифицированный язык моделирования (Unified Modelling Language);

W3C — Консорциум всемирной паутины (World Wide Web Consortium);

XML — расширяемый язык разметки (eXtensible Markup Language).

## 5 Модули ОТС

### 5.1 API сообщений и словаря

Между целевыми словарями и программами обработки сообщений, соответствующих стандарту HL7 Версии 3, предусмотрены два различных уровня, показанных на рисунке 1. Верхний уровень, API сообщений, взаимодействует с программами обработки сообщений в терминах словарных доменов, контекстов, наборов значений, кодированных атрибутов и других артефактов модели сообщений HL7. Нижний уровень, API словаря, взаимодействует с программным обеспечением терминологических служб в терминах систем кодирования, кодов понятий, обозначений, отношений и других сущностей, специфичных для терминологий.

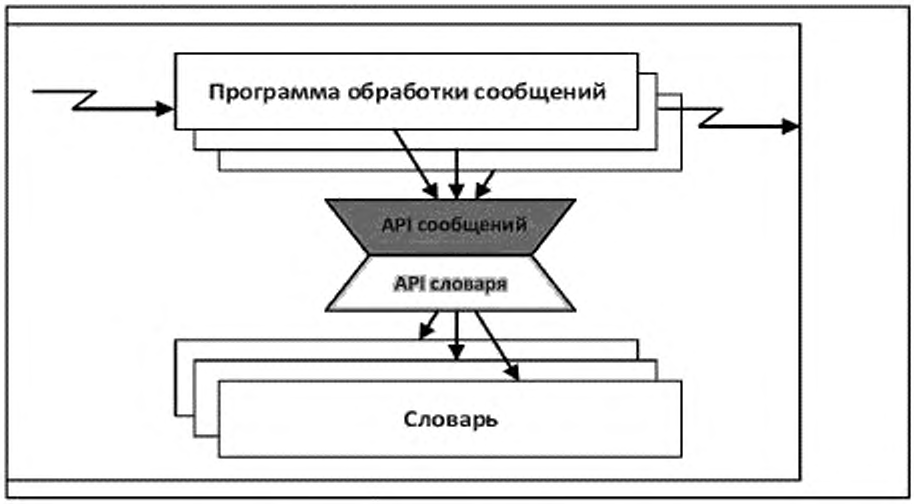


Рисунок 1 — API сообщений и словаря

API сообщений специфичен для HL7. Его основная цель — обеспечить широкому спектру программ обработки сообщений удобные и воспроизводимые возможности создания, проверки и преобразования типов данных, произведенных от типа CD (coded data — кодированные данные).

API словаря рассчитан на общее применение[1)]. Он позволяет прикладным программам осуществлять запросы к различным терминологиям[2)] хорошо определенным, согласованным способом. API сообщений включает в себя применение API словаря. На рисунке 2 показан пример взаимодействия программы обработки сообщений со словарем. В этом примере программа обработки сообщений обращается к службе времени исполнения уровня сообщений для заполнения детальных сведений об атрибуте, имеющем тип данных, произведенный от типа CD. Эта служба, в свою очередь, выполняет несколько обращений к службе уровня словаря для получения обозначений кода понятия, имен системы кодирования, версий ее выпусков и т. д.

---

[1)] Хотя API словаря и рассчитан на общее применение, он представляет некоторое подмножество тех возможностей, которые должен обеспечивать общий словарный API. Кроме того, значительная часть номенклатуры, используемой в API словаря ОТС, так или иначе связана со стандартами HL7, и для более общего применения может потребоваться преобразование этого API.

[2)] В контексте данного документа слово «терминология» означает описание любого организованного множества кодов, включая сущности, обычно называемые «наборами кодов», «онтологиями», «словарями», «системами классификации» и т. д.

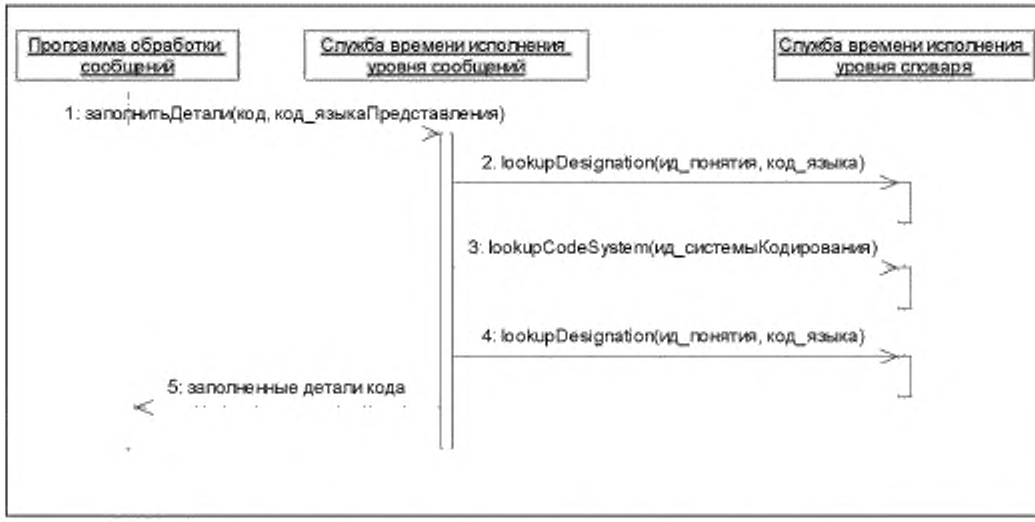


Рисунок 2 — Пример взаимодействия с API сообщений и словаря

### 5.2 Функции времени исполнения и обозревания

#### 5.2.1 Общие классы пользователей (действующих лиц) ОТС HL7

Классы действующих лиц, предполагаемых пользователями API ОТС HL7, включают в себя:

**Программное обеспечение создания сообщений** — программное обеспечение, создающее сообщения, соответствующие стандарту HL7. С точки зрения применения словарей процесс создания сообщений включает в себя преобразование внутренних структур сообщений и данных в синтаксис и семантику сообщений стандарта HL7 Версии 3.

**Программное обеспечение обработки сообщений** — программное обеспечение, получающее и декодирующее сообщения стандарта HL7 Версии 3 и действующее в соответствии с содержанием сообщений. Процесс обработки сообщений включает в себя шаги проверки и преобразования сообщений, а также принятие решений о дальнейших действиях.

**Разработчики модели RIM** — сочетание людей и инструментальных средств, создающих содержание и определения сообщений стандарта HL7.

**Разработчики программного обеспечения** — люди, создающие программное обеспечение, создающее, проверяющее и обрабатывающее сообщения стандарта HL7 Версии 3.

**Трансляторы словарей** — сочетание людей и инструментальных средств, преобразующих абстрактную спецификацию стандарта HL7 Версии 3 в структуры и термины, используемые конкретными приложениями обработки данных.

#### 5.2.2 Раздельные профили требований

Первые два класса действующих лиц — создатели и обработчики сообщений — имеют особый профиль требований, включающий в себя следующие три группы:

1) требования к производительности. Высокая пропускная способность и масштабируемость являются крайне важными при создании и обработке сообщений. Процессы преобразования словарей и моделирования гораздо менее чувствительны к вариабельным и потенциально субоптимальным временам ответа;

2) требования к надежности. Программное обеспечение, создающее и обрабатывающее сообщения, должно быть очень надежным, в то время как среда обработки словарей и среда моделирования могут допускать некоторую степень снижения надежности и случайных сбоев;

3) требования к функциональности. После согласования функциональные требования к программному обеспечению создания и обработки сообщений остаются фиксированными. Любые дополнительные возможности сверх этих требований использоваться не будут. В то же время в процессах разработки словарей и моделирования будут постоянно возникать различные новые запросы к извлечению

и просмотру информации, для удовлетворения которых время от времени могут создаваться любые (полезные) функциональные возможности.

В дальнейшем изложении профиль требований к созданию и обработке сообщений будет именоваться профилем времени исполнения, а профиль разработки словаря и моделирования — профилем обозревателя.

**5.2.3 Трансляция**

Необходимо выделить еще одну дополнительную функциональную область — трансляцию, то есть преобразование кодов понятий из одной системы кодирования в другую. Возможность трансляции наборов кодов для различных областей применения является существенной частью API сообщений. Она определена как отдельный интерфейс на уровне словаря, поскольку она не специфична для конкретного словаря. Службы трансляции потенциально могут разрабатываться независимо от одной или нескольких терминологий, охваченных процессом трансляции.

**5.2.4 Отдельные компоненты спецификации**

При сочетании уровней сообщения и словаря с двумя профилями требований образуются пять отдельных модулей, указанных в таблице 1.

Таблица 1 — Компоненты спецификации

| | Платформа времени исполнения | Обозреватель |
|---|---|---|
| API сообщений | Платформа сообщений | Обозреватель сообщений |
| API словаря | Платформа словаря | Обозреватель словаря |
| Преобразование | Преобразование словаря | |

Стиль изложения, принятый в настоящем документе, позволяет независимую реализацию требований каждой из этих областей, сохраняющую интероперабельность. Некоторые разработчики терминологических систем предпочтут сконцентрироваться на реализации API словаря, другим может понадобиться только реализация платформ времени исполнения. Теоретически API сообщений достаточно реализовать только один раз, поскольку структура сообщений HL7, положенных в основу этого API, опубликована и доступна каждому специалисту. Однако на практике для достижения желаемой производительности может понадобиться более тесная привязка платформы сообщений к платформе словаря.

## 6 Краткий обзор функций модулей

Следующие подразделы содержат краткий обзор функций каждого из пяти модулей, указанных в таблице 1. В целях большей наглядности некоторые параметры и вспомогательные функции опущены. Здесь даны только обзорные сведения, а в следующих разделах настоящего документа все функции, перечисленные в таблицах 2—6, будут описаны гораздо детальнее.

### 6.1 Функции платформы времени исполнения, предназначенной для уровня сообщений

Модуль платформы времени исполнения, предназначенной для уровня сообщений, предоставляет службы, используемые программным обеспечением создания, обработки и преобразования сообщений. Функции этих служб перечислены в таблице 2.

Таблица 2 — Функции платформы времени исполнения, предназначенной для уровня сообщений

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getServiceName | | Имя службы | Возвращает имя, присвоенное службе ее поставщиком |
| getServiceVersion | | Идентификатор версии | Возвращает текущую версию программного обеспечения службы |

*Продолжение таблицы 2*

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getServiceDescription | | Описание службы | Возвращает описание функции службы, авторов, сведений об авторских правах и т. д. |
| getHL7ReleaseVersion | | Идентификатор версии | Возвращает идентификатор версии выпуска стандарта HL7, которая в настоящее время поддерживается службой |
| getCTSVersion | | Старший и младший номер версии | Возвращает номер версии ОТС, реализованной службой |
| getSupportedMatch Algorithms | | Список алгоритмов совпадения | Возвращает список алгоритмов совпадения строк, реализованных данной службой |
| getSupportedVocabulary Domains | Поисковый образец и алгоритм совпадения, ограничения времени и размера | Список имен словарных доменов | Возвращает список словарных доменов, распознаваемых данной службой, имена которых совпадают с переданным образцом |
| validateCode | Имя словарного домена, проверяемый код, прикладной контекст (область применения), признак, указывающий, должны ли в проверке участвовать только активные понятия, а также признак, указывающий, надо ли определять ошибки и предупреждения или только ошибки | Список ошибок и предупреждений | Сравнивает значение кодированного атрибута с заданным словарным доменом и его контекстом |
| validateTranslation | Имя словарного домена, кодированный атрибут, содержащий одно или более проверяемых преобразований, прикладной контекст (область применения), признак, указывающий, должны ли в проверке участвовать только активные понятия, а также признак, указывающий, надо ли определять ошибки и предупреждения или только ошибки | Список ошибок и предупреждений | Сравнивает компоненты преобразования, если таковые присутствуют в переданном значении типа CD, с заданным словарным доменом и его контекстом |
| translateCode | Имя словарного домена, преобразуемый кодированный атрибут, целевая система кодирования и целевой прикладной контекст (область применения) | Преобразование кодированного атрибута | Преобразование значения заданного кодированного атрибута в форму, использующую целевую систему кодирования или ту систему кодирования, которая является наиболее подходящей для заданного контекста |

*Окончание таблицы 2*

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| fillInDetails | Кодированный атрибут и код целевого языка | Значение кодированного атрибута с дополнительными деталями | Заполняет необязательные компоненты значения кодированного атрибута, например, изображаемое имя понятия, имя и версию системы кодирования |
| subsumes | Родительский кодированный атрибут, дочерний кодированный атрибут | True/False | Определяет, действительно ли значение переданного родительского кодированного атрибута охватывает (или включает в себя) переданное значение дочернего атрибута |
| areEquivalent | Первый кодированный атрибут, второй кодированный атрибут | True/False | Определяет, являются ли значения двух кодированных атрибутов эквивалентными |
| lookupValueSet Expansion | Имя словарного домена, прикладной контекст (область применения), язык текста расширения, признак, указывающий, выполнить ли полное расширение или только расширение на один уровень, ограничения времени и размера | Иерархическое расширение набора значений, ассоциированного с доменом в переданном контексте | Возвращает иерархический список понятий, выбираемых для заданного словарного домена и контекста |
| expandValueSet ExpansionContext | Непрозрачный контекст расширения, возвращенный ранее вызванными функциями lookupValueSet Expansion или expand ValueSetExpansionContext | Дальнейшее иерархическое расширение набора значений, ассоциированного с доменом в переданном контексте | Возвращает дальнейшее расширение вложенного содержания набора значений |

### 6.2 Функции платформы времени исполнения, предназначенной для уровня словаря

Функции, перечисленные в таблице 3, используются службой времени исполнения на уровне сообщений и службой обозревателя уровня сообщений, а также службой обозревателя уровня словаря. Они используются как самостоятельные.

Таблица 3 — Функции платформы времени исполнения, предназначенной для уровня словаря

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getServiceName | | Имя службы | Возвращает имя, присвоенное службе ее поставщиком |
| getServiceVersion | | Идентификатор версии | Возвращает текущую версию программного обеспечения службы |
| getServiceDescription | | Описание службы | Возвращает описание функции службы, авторов, сведений об авторских правах и т. д. |
| getCTSVersion | | Старший и младший номер версии | Возвращает номер версии ОТС, реализованной службой |

*Окончание таблицы 3*

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getSupportedCode Systems | Ограничения времени и размера | Список систем кодирований и их версий, поддерживаемых данной реализацией службы | Возвращает идентификаторы, имена и идентификаторы выпусков всех систем кодирования, которые поддерживаются данной службой |
| lookupCodeSystemInfo | Имя или идентификатор системы кодирования | Сведения о системе кодирования, включая имя, идентификатор, версию, поддерживаемые языки, поддерживаемые отношения, поддерживаемые свойства и т. д. | Возвращает детальную информацию об указанной системе кодирования |
| isConceptIdValid | Идентификатор системы кодирования, код понятия и признак, указывающий, считаются ли неактивные понятия допустимыми | True/False | Определяет, является ли код понятия допустимым для текущей версии заданной системы кодирования |
| lookupDesignation | Идентификатор системы кодирования, код понятия и код целевого языка | Текст обозначения понятия | Возвращает предпочтительное обозначение кода понятия на указанном языке |
| areCodesRelated | Идентификатор системы кодирования, код исходного понятия, код целевого понятия, код отношения, квалификаторы отношения и признак, указывающий, использовать ли только непосредственно связанные коды или транзитивное замыкание отношения | True/False | Определяет, существует ли поименованное отношение между понятиями, заданными исходным и целевым кодом |

### 6.3 Функции отображения кодов

Функции отображения кодов перечислены в таблице 4.

Таблица 4 — Функции отображения кодов

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getServiceName | | Имя службы | Возвращает имя, присвоенное службе ее поставщиком |
| getServiceVersion | | Идентификатор версии | Возвращает текущую версию программного обеспечения службы |
| getServiceDescription | | Описание службы | Возвращает описание функции службы, авторов, сведений об авторских правах и т. д. |
| getCTSVersion | | Старший и младший номер версии | Возвращает номер версии ОТС, реализованной данной службой |

*Окончание таблицы 4*

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getSupportedMaps | | Список именованных наборов, строки которого состоят из идентификатора, имени и версии исходной системы кодирования, идентификатора, имени и версии целевой системы кодирования, а также описания отображения исходной системы на целевую | Возвращает список отображений, обеспечиваемых данной службой |
| mapConceptCode | Идентификатор исходной системы кодирования, код понятия в этой системе, идентификатор целевой системы кодирования и имя ресурса отображения | Соответствующий код понятия в целевой системе кодирования и признак качества отображения | Возвращает представление в целевой системе кодирования кода понятия, взятого из исходной системы кодирования, полученное с помощью поименованного ресурса отображения |

### 6.4 Функции обозревателя уровня сообщений

Функции обозревателя уровня сообщений перечислены в таблице 5.

Таблица 5 — Функции обозревателя уроеня сообщений

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getServiceName | | Имя службы | Возвращает имя, присвоенное службе ее поставщиком |
| getServiceVersion | | Идентификатор версии | Возвращает текущую версию программного обеспечения службы |
| getServiceDescription | | Описание службы | Возвращает описание функции службы, авторов, сведений об авторских правах и т. д. |
| getHL7ReleaseVersion | | Идентификатор версии | Возвращает идентификатор версии выпуска стандарта HL7, которая в настоящее время поддерживается службой |
| getCTSVersion | | Старший и младший номер версии | Возвращает номер версии ОТС, реализованной службой |
| getSupportedMatch Algorithms | | Список алгоритмов совпадения | Возвращает список алгоритмов совпадения строк, реализованных данной службой |
| lookupVocabulary Domain | Имя словарного домена | Имя домена, описание, список доменов, ограниченных данным доменом, список атрибутов модели RIM, использующих данный домен, и список наборов значений, представляющих данный домен | Извлечение всей информации, известной о заданном словарном домене |

*Продолжение таблицы 5*

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| lookupValueSet | Идентификатор или имя набора значений | Детальные сведения о наборе значений, включая имя, идентификатор, описание, список наборов значений, использованных для составления данного набора, наборы значений, в определении которых участвует данный набор, список кодов понятий, на которые ссылается данный набор, и т. д. | Извлечение детальной информации о наборе значений (включая словарные домены, конструкторы и т. д.) |
| lookupCodeSystem | Идентификатор или имя системы кодирования | Имя, идентификатор, авторские права, выпуск и информация о регистрации | Извлечение детальной информации о системе кодирования |
| lookupValueSet ForDomain | Имя словарного домена и прикладной контекст (область применения) | Имя и идентификатор набора значений, использованного для данного словарного домена | Возвращает идентификатор набора значений, который мог бы использоваться в заданном контексте (если таковой существует) |
| lookupVocabulary Domain | Имя словарного домена | Имя домена, описание, список доменов, ограниченных данным доменом, список атрибутов модели RIM, использующих данный домен, и список наборов значений, представляющих данный домен | Извлечение всей информации, известной о заданном словарном домене |
| lookupValueSet | Идентификатор или имя набора значений | Детальные сведения о наборе значений, включая имя, идентификатор, описание, список наборов значений, использованных для составления данного набора, наборы значений, в определении которых участвует данный набор, список кодов понятий, на которые ссылается данный набор, и т. д. | Извлечение детальной информации о наборе значений (включая словарные домены, конструкторы и т. д.) |
| lookupCodeSystem | Идентификатор или имя системы кодирования | Имя, идентификатор, авторские права, выпуск и информация о регистрации | Извлечение детальной информации о системе кодирования |
| lookupValueSetFor Domain | Имя словарного домена и прикладной контекст (область применения) | Имя и идентификатор набора значений, использованного для данного словарного домена | Возвращает идентификатор набора значений, который мог бы использоваться в заданном контексте (если таковой существует) |

*Окончание таблицы 5*

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| isCodeInValueSet | Идентификатор или имя набора значений, идентификатор системы кодирования и код понятия, а также признак, включать ли такой «ведущий код» в состав набора значений | True/False | Определяет, является ли указанный код понятия допустимым для заданного набора значений |

### 6.5 Функции обозревателя уровня словаря

Функции обозревателя уровня словаря перечислены в таблице 6.

Таблица 6 — Функции обозревателя уровня сообщений

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| getServiceName | | Имя службы | Возвращает имя, присвоенное службе ее поставщиком |
| getServiceVersion | | Идентификатор версии | Возвращает текущую версию программного обеспечения службы |
| getServiceDescription | | Описание службы | Возвращает описание функции службы, авторов, сведений об авторских правах и т. д. |
| getCTSVersion | | Старший и младший номер версии | Возвращает номер версии ОТС, реализованной службой |
| getSupportedMatch Algorithms | | Список алгоритмов совпадения | Возвращает список алгоритмов совпадения строк, реализованных данной службой |
| getSupportedAttributes | Поисковый образец и алгоритм совпадения, ограничения времени и размера | Список атрибутов модели RIM, известных обозревателю | Возвращает список атрибутов модели RIM, известных обозревателю, чьи имена совпадают с заданным образцом |
| getSupported CodeSystems | Поисковый образец и алгоритм совпадения, ограничения времени и размера | Список систем кодирования, известных обозревателю | Возвращает список систем кодирования, известных обозревателю, чьи имена совпадают с заданным образцом |
| lookupConcept CodesByDesignation | Идентификатор системы кодирования, поисковый образец и алгоритм совпадения, код целевого языка, признак, указывающий, должны ли извлекаться неактивные понятия, ограничения времени и размера | Список идентификаторов систем кодирования и кодов понятий | Возвращает список кодов понятий, чьи обозначения совпадают с заданным образом на заданном языке, если таковые найдены |
| lookupConcept CodesByProperty | Идентификатор системы кодирования, поисковый образец и алгоритм совпадения, код целевого языка, признак, указывающий, | Список идентификаторов систем кодирования/кодов понятий | Возвращает список кодов понятий, чьи свойства соответствуют заданным критериям |

*Окончание таблицы 6*

| Функция | Входные параметры | Выходные данные | Описание |
|---|---|---|---|
| | должны ли извлекаться неактивные понятия, необязательный список типов среды (mime) свойств, ограничения времени и размера | | |
| lookupComplete CodedConcept | Идентификатор системы кодирования и код понятия | Все, что известно о понятии (обозначения, свойства, отношения и т. д.) | Возвращает полное описание заданного кода понятия |
| lookupDesignations | Идентификатор системы кодирования и код понятия, поисковый образец и алгоритм совпадения, целевой язык | Список обозначений | Возвращает все обозначения заданного кода понятия, соответствующие заданным критериям |
| lookupProperties | Идентификатор системы кодирования и код понятия, поисковый образец и алгоритм совпадения, список кодов искомых свойств, список типов среды (mime), участвующих в поиске совпадений, код целевого языка | Список свойств понятия (код свойства, значение, язык, тип среды) | Возвращает свойства заданной системы кодирования/кода понятия, соответствующих заданным критериям |
| lookupCodeExpansion | Идентификатор системы кодирования и код понятия, код отношения, признак направления отношения, код целевого языка, ограничения времени и размера | Иерархический список расширения кода | Рекурсивно перечисляет коды понятий, связанных отношением с данным понятием, включая предпочтительные обозначения кодов |

## 7 Модель API сообщений OTC

### 7.1 Введение

В настоящем разделе описана модель, положенная в основу API сообщений OTC. Она определяет отношения между кодированными атрибутами HL7 и словарем. Она основана на метамодели HL7_V3_Meta-Model Version 1.16 и, исключая цветовое кодирование, по возможности остается совместимой с ее нотацией и типами данных. В настоящем стандарте не представлена полная или глубокая модель всех логических сущностей, образующих систему кодирования[1)]. Его основная цель — описать классы и отношения, имеющие прямое отношение к содержанию кодированных атрибутов HL7 с точки зрения метамодели.

Модель API словаря описывает модель OTC с точки зрения словаря и предусматривает дополнительную информацию о кодах понятий, обозначениях, отношениях и т. д.

#### 7.1.1 Нотация

На приведенных ниже диаграммах те классы, за чьи экземпляры полностью отвечают группы моделирования HL7, выделены зеленым цветом, а классы, представляющие содержание, контролируемые комитетом HL7 либо другим поставщиком терминологии, выделены бледно-желтым цветом. Авторы настоящего документа не обладают сколько-нибудь существенным контролем над содержанием или структурой этих существующих классов — они включены в модель исключительно для ссылок.

[1)] В настоящем стандарте термин «система кодирования» означает систему кодов, описаний, обозначений, свойств и отношений. К другим общим именам этой сущности относятся «словарь», «терминология», «схема кодирования», «схема классификации» и «онтология».

Обозначения, принятые в настоящей модели, используют полуформальную нотацию, основанную на содержании модели.

- имена классов выделены полужирным шрифтом (например, **VocabularyDomain**, **ValueSet**);
- имена атрибутов выделены курсивом (например, *vocabularyDomain_name*, *valueSet_id*);
- имена отношений подчеркнуты (например, экземпляр класса **VocabularyDomain** представлен нулем или более экземплярами классов **VocabularyDomainValueSets**).

При необходимости кратности отношений выражаются приведенными ниже формами или близкими к ним:

- 1..1 «экземпляр класса **Class 1** должен иметь отношение ровно с одним экземпляром класса **Class 2**»;
- 0..1 «экземпляр класса **Class 1** может иметь отношение не более чем с одним экземпляром класса **Class 2**»;
- 1..* «экземпляр класса **Class 1** должен иметь отношение с одним или несколькими экземплярами класса **Class 2**»;
- 1..* «экземпляр класса **Class 1** может иметь отношение с нулем или несколькими экземплярами класса **Class 2**».

### 7.2 Словарный домен

Словарный домен служит связующим звеном между кодированным атрибутом HL7 и набором (наборами) допустимых кодов понятий этого атрибута. Словарный домен представляет абстрактное концептуальное пространство наподобие «страны мира», «пол лица, используемый для административных целей» и т. д.

На рисунке 3 показано отношение между словарными доменами и атрибутами HL7.

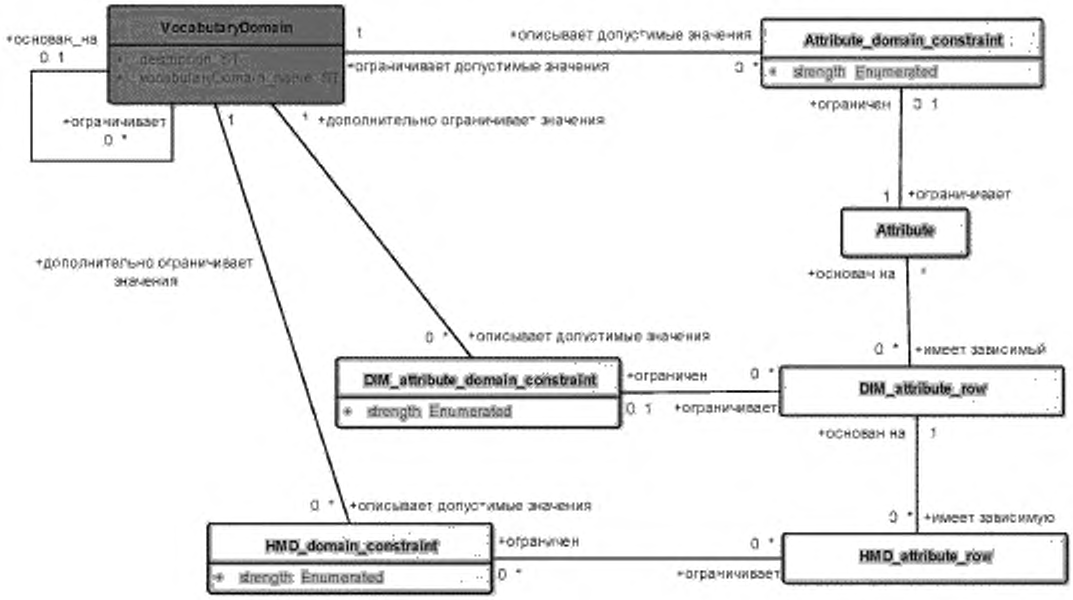


Рисунок 3 — Атрибуты модели RIM и словарные домены

Экземпляр класса **Attribute_domain_constraint** (ограничение домена атрибута) должен ограничивать допустимые значения ровно одного экземпляра класса **Attribute** (атрибута модели RIM). Он ограничивает допустимые значения атрибута теми, что описаны ровно одним экземпляром класса **VocabularyDomain** (словарный домен). Экземпляр класса **VocabularyDomain** может описывать допустимые значения одного или нескольких экземпляров класса **Attribute_domain_constraint** (ограничение домена атрибута). Экземпляр класса **DIM_attribute_row** (строка атрибута модели DIM) должен быть

основан ровно на одном экземпляре класса **Attribute**, взятом из модели HL7, и используется для указания на присутствие атрибута в специфичной информационной модели домена DIM (domain information model). Экземпляр класса **DIM_attribute_row** наследует все ограничения экземпляра класса **Attribute**, на котором он основан, и, в свою очередь, может быть ограничен нулем или более экземплярами класса **DIM_attribute_domain_constraint** (ограничение домена атрибута модели DIM).

Иерархическое описание сообщения HMD (hierarchical message description) полностью определяет структуру совокупности сообщений. Экземпляры класса **HMD_attribute_row** (строка атрибута в HMD) являются компонентами HMD, и каждый экземпляр класса **HMD_attribute_row** основан ровно на одном экземпляре класса **DIM_attribute_row**. Экземпляр класса **HMD_attribute_row** наследует все ограничения соответствующего экземпляра класса **DIM_attribute_row** и может быть далее ограничен не более чем одним экземпляром класса **HMD_domain_constraint** (ограничение домена HMD).

Каждый экземпляр класса **VocabularyDomain** имеет уникальное имя (атрибут vocabularyDomain_name), а также описание пространства понятий (атрибут description), которое он представляет. Экземпляры класса **VocabularyDomain**, которые описывают допустимые значения экземпляров класса **DIM_attribute_domain_constraint** или класса **HMD_domain_constraint**, ограничивают пространство понятий соответствующего экземпляра класса **DIM_attribute_domain** или класса **Attribute_domain_constraint**, на котором основан домен.

Пример, приведенный в таблице 7, показывает, каким образом словарные домены могут быть ограничены в зависимости от атрибута, строки атрибута модели DIM или строки иерархического описания сообщения HMD.

Таблица 7 — Ограничения словарных доменов

| Атрибут | Словарный домен | Описание | Ограничиваемый домен |
|---|---|---|---|
| sourceCountry | Country | Страна мира | — |
| sourceCountry(DIM) | HL7MemberCountry | Страна — официальный член комитета HL7 | Country |
| sourceCountry(HMD) | EUHL7MemberCountry | Европейская страна — официальный член комитета HL7 | HL7MemberCountry |

### 7.2.1 Дополнительные ограничения

Примечание — В этом контексте ограничение относится к самой модели, а не к доменам атрибутов.

1. Экземпляр класса **Attribute** должен быть ограничен ровно одним экземпляром класса **Attribute_domain_constraint** в том и только том случае, если атрибут имеет кодированный тип данных.

2. Любой экземпляр класса **VocabularyDomain**, описывающий допустимые значения экземпляра класса **DIM_attribute_domain_constraint**, должен быть основан на экземпляре класса **VocabularyDomain**, соответствующем экземпляру класса **Attribute**.

3. Любой экземпляр класса **VocabularyDomain**, описывающий допустимые значения экземпляра класса **RIM_attribute_domain_constraint**, должен быть основан на экземпляре класса **VocabularyDomain**, соответствующем экземпляру класса **DIM_attribute_row**, если таковой существует, либо экземпляру класса **Attribute**.

## 7.3 Набор значений

Словарный домен описывает «понятийное пространство», из которого могут быть взяты значения атрибута. Но прежде чем атрибут может быть использован в сообщении, необходимо определить фактический список кодов понятий. Список допустимых кодов понятий называется набором значений. Структура набора значений и его ассоциации показаны на рисунке 4.

Экземпляр класса **VocabularyDomain** может быть представлен нулем или несколькими экземплярами класса **ValueSet** (набор значений). Хотя абстрактные атрибуты моделей RIM и DIM не обязаны быть представлены какими-либо экземплярами класса **ValueSet**, экземпляр класса **VocabularyDomain**, описывающий допустимые значения кодированных экземпляров класса **Attribute**, используемых в фактических сообщениях, должен быть представлен не менее чем одним экземпляром класса **Value_set**.

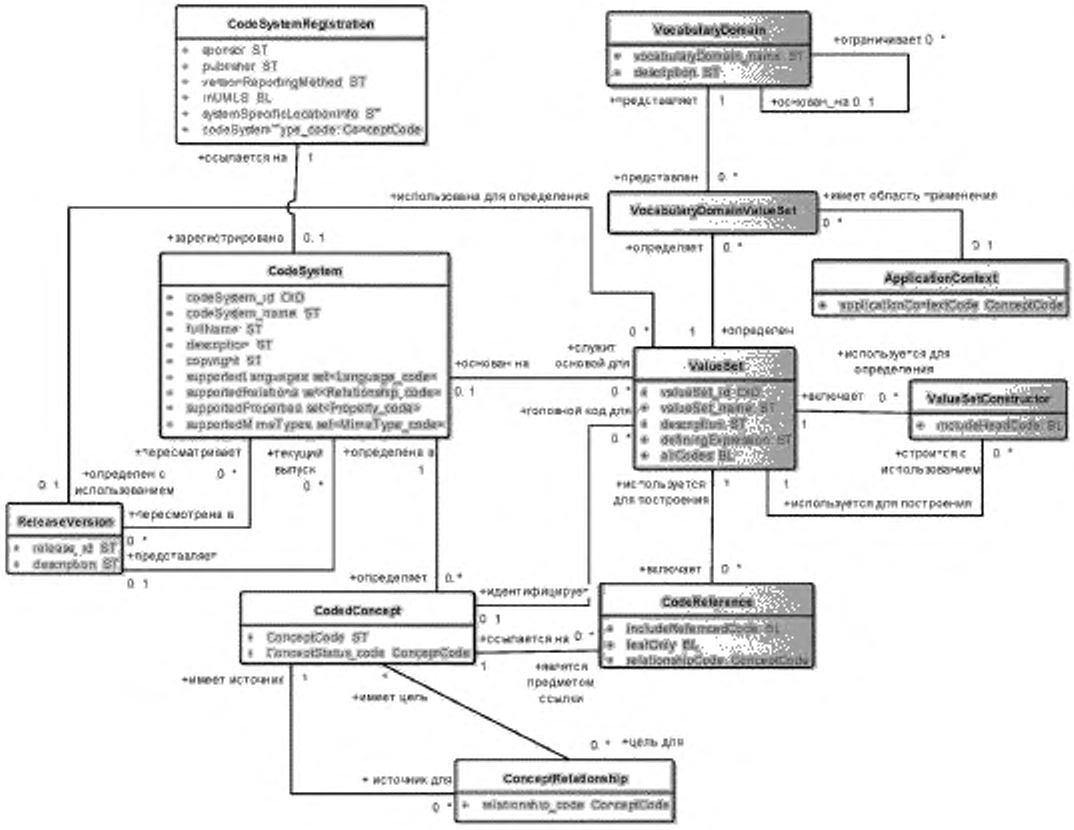


Рисунок 4 — Наборы значений

**7.3.1 Связи между словарными доменами и наборами значений**

Экземпляр класса **VocabularyDomainValueSet** (набор значений в словарном домене) представляет ассоциацию ровно между одним экземпляром класса **VocabularyDomain** и одним экземпляром класса **ValueSet**. Каждая ассоциация между экземпляром класса **VocabularyDomain** и экземпляром класса **ValueSet** может применяться в нуле или более экземпляров класса **ApplicationContext** (контекст применения). Экземпляр класса **ApplicationContext** именует конкретную геополитическую единицу (например, Европейский союз, Канада) и/или отрасль (например, ветеринарную медицину, общественное здоровье) и т. д. и может служить областью применения для нуля или более ассоциаций, представленных экземплярами класса **VocabularyDomainValueSet**.

**7.3.2 Определение наборов данных**

Экземпляр класса **ValueSet** может включать в себя список, состоящий из нуля и более экземпляров класса **CodedConcept** (кодированное понятие), взятых из одного и того же экземпляра класса **CodeSystem** (система кодирования). Экземпляр класса **ValueSet** может представлять:

- все кодированные понятия **CodedConcept**, определенные ровно в одной системе кодирования **CodeSystem**;
- конкретный список кодированных понятий **CodedConcept**, определенных ровно в одной системе кодирования **CodeSystem**;
- совокупность кодированных понятий **CodedConcept**, представленных другим набором значений **ValueSet**.

Детальные сведения о каждой из этих форм приведены в следующих подразделах. Вначале будут описаны характеристики, общие для всех наборов значений.

Ожидается, что наборов значений столь много, что просто невозможно присвоить каждому из них уникальную мнемонику или значащее имя. Основным идентификатором экземпляра класса **ValueSet** является атрибут *valueSet_id*, имеющий числовое значение, не имеющее смысловой нагрузки[1]. При необходимости атрибут *valueSet_name* может также содержать уникальный «смысловой» идентификатор набора значений. Этот атрибут предназначен для коммуникаций между углеродными формами жизни.

Класс **ValueSet** имеет атрибут *description* (описание), описывающий назначение и цель создания набора значений. Он имеет также атрибут *definingExpression* (определяющее выражение), предназначенный для хранения формального машиночитаемого выражения, которое может использоваться для конструирования набора значений. Значение и интерпретация этого выражения в данном документе не обсуждаются. Оба атрибута (*description* и *definingExpression)* не обязательны. Еще один атрибут класса **ValueSet**, а именно, *allCodes* (все коды), описан ниже.

**7.3.3 Определение содержания набора значений**

Набор значений **ValueSet** может быть построен из одного другого набора значений **ValueSet** либо может быть основан ровно на одной системе кодирования **CodeSystem**, либо может быть построен из того и другого[2]. Система кодирования **CodeSystem** определяет нуль или более кодированных понятий **CodedConcepts**, которые, в свою очередь, обозначают релевантные классы или сущности в конкретной области интереса. Системы кодирования могут ранжироваться от простой таблицы пола человека до классификаторов типа МКБ-9 и до семантически богатых систем, основанных на логических представлениях, например, OpenGalen или SNOMED-CT. Поскольку многие системы кодирования регулярно пересматриваются, полезно регистрировать версию выпуска **ReleaseVersion**, использованную для определения данного набора значений **ValueSet** в конкретный момент времени. Набор значений **ValueSet** может быть определен с использованием не более одной версии **ReleaseVersion**. Определение с помощью отношения служит исключительно для справочных целей и для представления содержания набора значений службы **ValueSet** могут использовать более позднюю версию системы кодирования[3]. Более детальное обсуждение систем кодирования приведено ниже, в подразделе, описывающем API словаря ОТС.

7.3.3.1 Представление всего содержания системы кодирования

Присвоив значения TRUE атрибуту *allCodes*, можно указать, что в данный набор значений **ValueSet** должны быть включены все кодированные понятия **CodedConcept**, определенные в системе кодирования **CodeSystem**, на которой основан этот набор. Такой набор значений **ValueSet** не может дополнительно включать в себя ссылки на кодированные понятия **CodeReference**. Пример набора значений, представляющего всю систему кодирования, приведен в таблице 8.

Таблица 8 — Набор значений, представляющий все содержание системы кодирования

| valueSet_id | valueSet_name | description | allCodes | CodeSystem | Версия |
|---|---|---|---|---|---|
| 2.16.840.1.113883.1.11.1 | AdministrativeGender | Пол лица, используемый для административных целей (в отличие от клинического пола) | True | 2.16.840.1.113883.5.1 | 5 |

7.3.3.2 Представление частей содержания системы кодирования

Набор значений **ValueSet** может включать в себя нуль или более кодированных понятий **CodedConcept**, определенных в системе кодирования **CodeSystem**, на которой он основан. Это выполняется с помощью включения в набор значений одной или нескольких ссылок **CodeReference**. Такая ссылка связывает кодированное понятие **CodedConcept** с набором значений **ValueSet**. Она должна ссылаться ровно на одно кодированное понятие **CodedConcept**. При этом атрибут *relation_code* подразумевает неявные ссылки на все кодированные понятия **CodedConcept**, являющиеся целью

[1] В качестве такого значения, скорее всего, будет использоваться объектный идентификатор ИСО с корнем, 2.16.840.1.113883.1.11, но это решение окончательно не принято.

[2] Можно сконструировать набор значений, ссылающийся на другие наборы значений, извлеченные более чем из одной системы кодирования. Однако конкретный набор значений может напрямую ссылаться только на понятия, извлеченные из единственной системы кодирования.

[3] Настоящий документ не содержит обсуждение того, как указать конкретную версию системы кодирования или состояние набора значений в определенный момент времени. Соответствующие разделы появятся в следующих выпусках документа.

отношения **ConceptRelationship** к явно включенному кодированному понятию **CodedConcept**. В ссылке **CodeReference** можно указать, что само это ссылочное понятие **CodedConcept** включено в набор значений или исключено из него. С помощью отношения **ConceptRelationship** можно включить все целевые коды понятий или только конечные узлы (листья). Эти различные возможности отражены в значениях атрибутов includeReferencedCode, relationship_code и leafOnly, приведенных в таблице 9.

Таблица 9 — Варианты значений атрибутов, управляющих ссылками на понятия

| allCodes | includeReferencedCode | relationship_code | leafOnly | Описание |
|---|---|---|---|---|
| True | — | — | — | Включает в набор значений все коды, описанные в системе кодирования |
| False | True | — | — | Включает в набор значений только ссылочный код понятия |
| False | True | hasSubtype | False | Включает в набор значений ссылочный код и все его подтипы |
| False | False | hasSubtype | False | Включает в набор значений все подтипы ссылочного кода, но не сам этот код |
| False | False | hasSubtype | True | Включает в набор значений только листовые подтипы ссылочного кода |

Иные сочетания значений атрибутов, кроме тех, что перечислены в таблице 9, не допустимы. По возможности значения атрибута relation_code должны браться из системы кодирования HL7 RelationshipCode.

**7.3.4 Включение в набор значений других наборов значений**

Набор значений **ValueSet** может быть также построен, используя нуль или более дополнительных наборов значений. Включение в его состав набора значений **ValueSet** означает, что все кодированные понятия **CodedConcept**, представленные включаемым набором, должны стать частью результирующего набора. Класс **ValueSetConstructor** (конструктор набора значений) служит двум целям:

1) С его помощью можно включать наборы значений по ссылке, что позволяет всем изменениям этих наборов автоматически отражаться в результирующем наборе.

2) Использование этого класса позволяет включать в результирующий набор ValueSet кодируемые понятия CodedConcept, взятые более чем из одной системы кодирования CodeSystem.

В таблице 10 приведен пример конструирования набора значений HL7ConformanceInclusion, включающего в себя набор значений InclusionNotMandatory, который, в свою очередь, включает в себя набор InclusionNotRequired.

Таблица 10 — Пример конструирования наборов значений

| usedToBuildSet | (имя) | includedSet | (имя) | includeHeadCode |
|---|---|---|---|---|
| 2.16.840.1.1138 83.1.11.10010 | HL7ConformanceInclusion | 2.16.840.1.1138 83.1.11.10012 | InclusionNot Mandatory | False |
| 2.16.840.1.1138 83.1.11.10012 | InclusionNotMandatory | 2.16.840.1.1138 83.1.11.10015 | InclusionNot Required | True |

**7.3.5 Головные коды**

Набор ValueSet значений ссылается на совокупность кодированных понятий CodedConcept. Нередко ассоциация между набором значений ValueSet и коллекцией кодированных понятий CodedConcept означает категоризацию, отношение «часть — целое» или иное иерархическое отношение, в котором сам набор значений представляет «целое» (родителя), а коды понятий — отдельные «части» (потомки). В этом случае в системе кодирования может существовать соответствующий код понятия, представляющий подобное понятие «целого» или «родителя». Такой код будет именоваться

«головным кодом» ассоциированного набора значений. Многие кодированные атрибуты модели HL7 RIM могут иметь разные степени общности. В примере набора значений, указанном в таблице 10, набор значений **InclusionNotRequired** имеет головной код NR (не требуется) и два подчиненных кода — X (исключен) и RE (требуется, но может быть пустым). Значение True атрибута *includeHeadCode*, указанное во второй строке этой таблицы, означает, что для этого набора допустимыми значениями могут быть X (исключен), RE (требуется, но может быть пустым) и NR (включение не требуется по неспецифичной причине).

Если набор значений используется для построения другого набора, то головной код может быть или не быть частью включаемого набора. Тем самым обеспечивается возможность того, что сообщество HL7 называет «абстрактными» и «специализируемыми» наборами значений. Набор значений **ValueSet** может быть идентифицирован не более чем одним кодированным понятием, называемым «головным кодом». Кодированное понятие **CodedConcept** может служить головным кодом для нуля и более наборов значений **ValueSet**.

### 7.4 Зарегистрированная система кодирования

Многие системы кодирования, используемые в стандартах HL7, предоставляются внешними организациями. Класс CodeSystem, который будет определен в этом документе позже, представляет характеристики, общие для всех систем кодирования, используемых в среде HL7. Комитет HL7 ведет также внутренний регистр систем кодирования, содержащий их метаданные. Цель этого регистра — служить центральным хранилищем метаданных всех систем кодирования, которые могут использоваться в сообщениях, соответствующих стандартам HL7, вне зависимости от того, являются ли они внутренними для места реализации или системы либо общепринятыми и санкционированными для ряда систем.

С точки зрения комитета HL7, регистрация не означает «санкционирование». Она является просто ссылкой. В процессе регистрации системе кодирования присваивается объектный идентификатор (ОИД), если его у нее еще не было. На рисунке 5 показана диаграмма классов зарегистрированной системы кодирования.

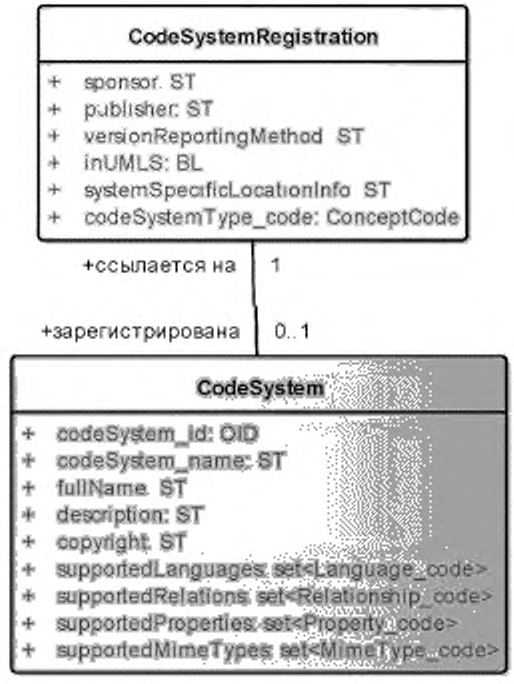


Рисунок 5 — Зарегистрированная система кодирования

Класс **CodeSystemRegistration** (регистрация системы кодирования) имеет следующие атрибуты:

- *sponsor* — именование, адрес и другие подобные данные лица либо организации, которая официально спонсирует эту систему кодирования для комитета HL7;
- *publisher* — наименование, адрес и другие подобные данные организации, ответственной за создание и ведение этой системы кодирования;
- *versionReportingMethod* — описание того, как и сколь часто создаются и распространяются новые версии;
- *licensingInformation* — описание требуемых лицензий, цены и способа приобретения;
- *inUMLS* — значение TRUE указывает, что кодированные термины, определенные в этой системе кодирования, включены в Унифицированную систему медицинского языка UMLS (Unified Medical Language System);
- *systemSpecificLocatorInfo* — информация, специфичная для издателя системы кодирования. Она служит для определения или идентификации конкретной системы кодирования. Комитет HL7 может иногда использовать атрибут systemSpecificLocatorInfo для идентификации соответствующей таблицы стандарта HL7 2.x, если таковая имеется;
- *codeSystemType_code* — код, имеющий значения I (система кодирования является внутренней, ведется и распространяется комитетом HL7), E (ведется и распространяется третьей стороной), EI (ведется третьей стороной, но распространяется комитетом HL7).

## 8 Спецификация API сообщений OTC

### 8.1 Введение

API сообщений OTC (CTSMAPI — CTS Message API) состоит из двух разделов — раздел времени исполнения, определяющий комплекс функций, необходимых для каждодневных операций программного обеспечения, обрабатывающего сообщения, соответствующие стандарту HL7 Версии 3, и раздел обозревателя, определяющего функции, предназначенные для разработки и конструирования этих сообщений. Программному обеспечению, использующему функции обозревателя, может понадобиться доступ к функциям времени исполнения, в то время как программное обеспечение времени исполнения может не иметь доступа к функциям обозревателя.

### 8.2 Общие элементы сообщений OTC

#### 8.2.1 Основные элементы данных

В следующем пункте определены основные типы данных, используемые в API сообщений OTC. Все типы, имена которых имеют префикс «types::», основаны на стандарте HL7 Версии 3 Data Types и далее здесь не обсуждаются.

Основные типы данных в API сообщений OTC включают в себя:

- CTSVersionId — структура, содержащая старшую и младшую части номера версии спецификации OTC. У текущей версии старшая часть равна 1, младшая — равна 0 (1.0);
- CodeSystemId — уникальный идентификатор системы кодирования. В контексте стандартов HL7 им должен быть объектный идентификатор ИСО (ОИД), присвоенный комитетом HL7, если таковой имеется. Другие виды идентификаторов, например UUID, определенный для среды распределенных вычислений DCE (Distributed Computing Environment), и т. д., могут использоваться в качестве идентификаторов систем кодирования вне среды применения стандартов HL7. В этих случаях разработчик должен предупреждать любые конфликты пространств имен, которые могут возникать между ОИД и другими идентификаторами;
- CodeSystemName — имя системы кодирования. Как идентификатор системы кодирования, так и ее имя уникальны в пространстве имен, контролируемом стандартом HL7 Версии 3, но в общем случае уникальность не гарантируется;
- ConceptCode — код, уникально представляющий класс или понятие в данной системе кодирования;
- VocabularyDomainName — уникальное имя словарного домена HL7;
- ReleaseVersionId — идентификатор, уникально обозначающий выпуск/версию в контексте системы кодирования;
- ValueSetId — объектный идентификатор ИСО (ОИД), уникально идентифицирующий набор значений HL7;

- ValueSetName — уникальное имя или мнемокод набора значений. Не у всех наборов значений есть и идентификатор, и имя, но если таковые имеются, то они должны быть уникальны;

- ConceptId — сочетание идентификаторов системы кодирования и кода понятия, являющееся глобально уникальным именем понятия;

- ReleaseVersionId — уникальный идентификатор версии или выпуска одной или нескольких систем кодирования;

- ExpansionContext — непрозрачный большой двоичный объект, используемый для передачи контекстной информации между сервером и клиентом.

**8.2.2 Коды понятий**

В API сообщений используется следующий ряд кодированных атрибутов:

- LanguageCode — код разговорного или письменного языка, конструируемый по правилам, описанным в документе IETF RFC 3066 — Tag for Identification of Languages (теги идентификации языка). Этот код состоит из нескольких субтегов, разделенных дефисами («-»). Первый субтег идентифицирует основной код языка. По возможности он должен быть взят из ИСО 639-1 «Codes for the representation of names of languages — Part 1: Alpha-2 code» (Коды для представления названий языков. Часть 1. Двухбуквенный код). Если двухбуквенный код отсутствует, то надо взять код из ИСО 639-2 «Codes for the representation of names of languages — Part 2: Alpha-3 code» (Коды для представления названий языков. Часть 2. Трехбуквенный код). Существует также дополнительный механизм спецсимволов, который в настоящем стандарте не описан.

Второй субтег не обязателен. Если он присутствует, то должен иметь длину от 2 до 8 символов. Если его длина равна двум, то он должен содержать двухбуквенный код страны, взятый из ИСО 3166-1 «Codes for the representation of names of countries and their subdivisions — Part 1: Country codes» (Коды для представления названий стран и единиц их административно-территориального деления. Часть 1. Коды стран). Если длина субтега от 3 до 8 символов, то он должен содержать код, взятый из регистра тегов языков организации IANA. Дополнительные субтеги используются, если надо уточнить информацию о языке;

- RelationshipCode — код понятия, уникально определяющего конкретное отношение, имеющее место в системе кодирования. По возможности коды отношений должны браться из системы кодирования HL7 ConceptRelationship;

- ApplicationContextCode — код, идентифицирующий контекст или область применения, например, геополитическое образование, профессию или другую сферу.

- DataTypeCode — код, идентифицирующий тип данных атрибута модели RIM (например, CD, CE, CS, BIN, ST и т. д.).

- CodingStrengthCode — код, указывающий, как должны трактоваться некодированные значения атрибута, определенного в стандарте HL7 (CWE — кодированный с разрешением исключений, CNE — кодированный без исключений);

- ValueSetNodeTypeCode — код, определяющий тип набора значений, возвращаемого в виде иерархического списка. Коды имеют следующие значения: A (абстрактный) означает, что набор значений не может быть выбран; S (специализируемый) означает, что набор значений может быть выбран, но имеет дальнейшие уточнения; L (list) означает, что узел представляет выбираемый код понятия, которое не имеет дальнейших уточнений;

- CodeSystemTypeCode — код, имеющий значения I (система кодирования является внутренней, которая ведется комитетом HL7), E (ведется третьей стороной), EI (ведется третьей стороной, но в распоряжение комитета HL7 предоставлена внутренняя копия);

- MatchAlgorithmCode — код, идентифицирующий алгоритм совпадения строк, используемый во внутренних функциях поиска.

В таблице перечислены системы кодирования и идентификаторы ОИД, используемые в MAPI сообщений, предоставляемом ОТС.

Таблица 11 — Идентификаторы ОИД и имена кодируемых элементов данных

| Элемент данных MAPI | ОИД системы кодирования | Имя системы кодирования | Описание |
|---|---|---|---|
| LanguageCode | 2.16.840.1.11388 3.6.99 | ISO 639-1 Two character Alpha Language Codes | Двухсимвольные коды языков ИСО 639-1 |

*Окончание таблицы 11*

| Элемент данных MAPI | ОИД системы кодирования | Имя системы кодирования | Описание |
|---|---|---|---|
| LanguageCode | 2.16.840.1.11388 3.6.100 | ISO 639-2 Three character Alpha Language Codes | Трехсимвольные коды языков ИСО 639-2 |
| RelationshipCode | 2.16.840.1.11388 3.5.1088 | ConceptCodeRelationship | Отношения кодов понятий |
| ApplicationContextCode | 2.16.840.1.11388 3.5.147 | VocabularyDomainQualifier. RealmOfUse | Квалификатор словарного домена — область применения |
| DataTypeCode | 2.16.840.1.11388 3.5.1007 | DataType | Тип данных |
| CodingStrengthCode | 2.16.840.1.11388 3.5.147 | VocabularyDomainQualifier | Квалификатор словарного домена |
| ValueSetNodeTypeCode | 2.16.840.1.11388 3.5.24 | ConceptGenerality | Степень общности понятия |
| CodeSystemType | 2.16.840.1.11388 3.5.1085 | CodeSystemType | Тип системы кодирования |
| MatchAlgorithmCode | 2.16.840.1.11388 3.5.1094 | MatchAlgorithm | Алгоритм совпадения |

8.2.2.1 Алгоритмы совпадения строк

Некоторым описанным ниже функциям могут передаваться строковые значения в качестве критериев поиска. Этим значениям сопутствует «код алгоритма совпадения», определяющий, как будут применяться эти значения. В таблице перечислен список предопределенных алгоритмов совпадения. Если в графе «Обязателен» указано значение TRUE, то все реализации службы, объявляющие совместимость с данным стандартом, должны использовать этот алгоритм. Если в графе «Обязателен» указано значение FALSE, то служба не обязательно должна реализовать этот алгоритм, но если реализует, то в качестве его идентификатора должна использовать код, указанный в таблице 12. Данный список алгоритмов совпадения не является исчерпывающим. При необходимости реализация службы может расширить этот список собственными алгоритмами совпадения, но в целях будущей интероперабельности разработчикам службы настоятельно рекомендуется зарегистрировать код алгоритма в комитете HL7.

Таблица 12 — Коды алгоритмов совпадения

| Код алгоритма совпадения | Описание | Обязателен |
|---|---|---|
| IdenticalIgnoreCase | Целевой текст, представленный в нижнем регистре, должен точно совпадать с представлением параметра matchText в нижнем регистре | TRUE |
| Identical | Целевой текст должен точно совпадать со значением параметра matchText | FALSE |
| StartsWithIgnoreCase | Целевой текст, представленный в нижнем регистре, должен начинаться с подстроки, совпадающей с представлением значения параметра matchText в нижнем регистре | TRUE |
| StartsWith | Целевой текст должен начинаться с подстроки, совпадающей со значением параметра matchText | FALSE |
| EndsWithIgnoreCase | Целевой текст, представленный в нижнем регистре, должен заканчиваться подстрокой, совпадающей с представлением значения параметра matchText в нижнем регистре | TRUE |
| EndsWith | Целевой текст должен заканчиваться подстрокой, совпадающей со значением параметра matchText | FALSE |

*Окончание таблицы 12*

| Код алгоритма совпадения | Описание | Обязателен |
|---|---|---|
| ContainsPhraseIgnoreCase | Целевой текст, представленный в нижнем регистре, должен содержать подстроку, совпадающую с представлением значения параметра matchText в нижнем регистре | TRUE |
| ContainsPhrase | Целевой текст должен содержать подстроку, совпадающую со значением параметра matchText | FALSE |
| WordsAnyOrderIgnoreCase | Целевой текст должен содержать все слова подстроки, совпадающей со значением параметра matchText, но не обязательно в том же порядке | FALSE |
| WildCardsIgnoreCase | Значение параметра matchText может содержать нуль или более «заместителей», обозначаемых символом звездочки (*). Заместитель совпадает с нулем или более символов целевой строки. Спецсимвол обратной косой черты (\) означает, что значение «a*b*» параметра matchText совпадает с любой строкой, начинающейся символами «a*b» | FALSE |
| RegularExpression | Значение параметра matchText может содержать регулярные выражения, синтаксис которых определен в документе XML Schema Part 2: Datatypes (XML-схема. Часть 2. Типы данных) | FALSE |
| NYSIIS | Фонетическое кодирование, принятое в системе New York State Identification and Intelligence System | FALSE |

**8.2.3 Идентифицирующий раздел службы**

API времени исполнения и API обозревателя наследуют общий интерфейс идентификации, предусматривающий следующие вызовы:

- getServiceName — возвращает имя, присвоенное службе ее поставщиком;
- getServiceVersion — возвращает идентификатор версии, зависящий от реализации службы;
- getServiceDescription — возвращает описание службы, сведения о ее авторе, назначении и т. д.;
- getHL7ReleaseVersion — возвращает идентификацию наиболее позднего выпуска словаря HL7 (а не модели RIM), предоставляемого службой;
- getCTSVersion — возвращает специфичную версию ОТС, реализуемую службой (например, 1.0).

**8.2.4 Исключения**

Ниже приведен перечень исключений, которые могут генерироваться одним или несколькими методами, описанными в настоящем разделе. Исключения вызываются аномальными событиями, препятствующими нормальному завершению работы.

В приведенном ниже описании исключений предполагается, что базовая информация об исключении содержит текстовое поле, описывающие специфичные детали исключения. Дополнительные атрибуты, указанные в этом списке, предоставляют информацию, дополняющую базовый текст.

Исключения, генерируемые API сообщений:

- возникла ошибка, не предусмотренная в настоящей спецификации. К примеру, ошибки оперативной памяти, ошибки ввода-вывода, ошибки доступа к базе данных и любые другие неожиданные события, препятствующие нормальному завершению вызова API. В атрибуте *possible_cause* может передаваться более детальная информация о том, что произошло на самом деле;
- служба не распознала значение параметра vocabularyDomain_name;
- служба не распознала значение параметра applicationContext_code. Возможны следующие варианты:

1) идентификатор набора значений передан, но служба не распознала его;

2) передано только имя набора значений, но служба не распознала его;

3) не переданы ни имя, ни идентификатор;

- при вызове метода переданы оба параметра — valueSet_id и valueSet_name. Параметр valueSet_id правильно идентифицирует набор значений, но он отличается от того, что указан в параметре valueSet_name, который может идентифицировать другой набор значений или вообще не указывать на существующий набор значений;
- параметр concept_id.concept_code не распознан как правильный код понятия, принадлежащий системе кодирования, идентифицированной значением параметра concept_id.codeSystem_id;

- служба не поддерживает проверку категоризации для системы кодирования, идентифицированной значением параметра codeSystem_id;
- переданный квалификатор отношения не распознан службой;
- служба не может выполнить требуемую трансляцию:

1) параметр codeSystem_id передан, но не распознан службой;

2) параметр codeSystem_id не передан, а параметр codeSystem_name передан, но его значение не распознано службой;

3) ни один из параметров codeSystem_id и codeSystem_name не передан;

- клиент предоставил неправильный параметр ExpansionContext. Это может означать, что контекст каким-то образом поврежден либо превышены ограничения по времени и контекст перестал быть действительным;
- служба не поддерживает проверку категоризации посткоординируемого выражения;
- служба не может определить по значениям параметров vocabularyDomain_name и applicationContext_code, какой набор значений следует использовать;
- значение параметра codeSystem_name не идентифицирует ту же систему кодирования, что и значение параметра codeSystem_id, либо вообще не именует систему кодирования;
- превышены ограничения по времени, заданные при вызове функции;
- текст, переданный в параметре matchText, синтаксически не корректен по отношению к указанному алгоритму совпадения;
- служба не поддерживает алгоритм совпадения, заданный при ее вызове.

### 8.3 API времени исполнения уровня сообщений

#### 8.3.1 Идентифицирующий раздел API времени исполнения уровня сообщений

В следующих пунктах описаны методы интерфейса той части API уровня сообщений, которая отвечает за функции времени исполнения.

Раздел API времени исполнения наследует идентифицирующую информацию от раздела идентификации и имеет два дополнительных метода:

- getSupportedMatchAlgorithms — возвращает список алгоритмов совпадения строк, реализованных данной службой;
- getSupportedVocabularyDomains — возвращает список словарных доменов, распознаваемых данной службой.

Параметры:

- matchText — если присутствует и имеет непустое значение, то возвращается идентификация только тех словарных доменов, чьи имена совпадают с этим значением. Если же этот параметр отсутствует или пуст, то возвратится идентификация всех словарных доментов;
- matchAlgorithm_code — если параметр matchText присутствует и имеет непустое значение, то параметр matchAlgorithm_code указывает, каким образом определяется совпадение имен со значением параметра matchText. Детали см. в 8.2.2.1 «Совпадение строк»;
- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- BadlyFormedMatchText;
- UnknownMatchAlgorithm;
- TimeoutError;
- UnexpectedError.

#### 8.3.2 Проверка кодированного атрибута

8.3.2.1 Структура объекта ValidateCodeReturn

Класс ValidationDetail содержит детальное описание ошибки или предупреждения. Она имеет следующие поля:

- codeInError — код понятия, с которым связана данная ошибка или предупреждение. Если ошибка обнаружена в квалификаторе понятия или в его трансляции, то этот код содержит то, в чем найдена ошибка: имя квалификатора, значение квалификатора либо значение трансляции;

- isError — значение TRUE указывает, что элемент ValidationDetail описывает ошибку. Значение FALSE указывает, что он описывает предупреждение;

- error_id — идентификатор ошибки (см. коды возврата методов validateCode и validateTranslation, перечисленные в таблице);

- errorText — текст описания ошибки или предупреждения.

В экземпляре класса ValidateCodeReturn возвращается общая и детальная информация о результате вызова метода validateCode, включая следующие поля:

- nErrors — число обнаруженных ошибок. Если оно больше нуля, то код или трансляция ошибочны;

- nWarnings — число возвращенных предупреждений. Если результат проверки кода или трансляции содержит только предупреждения, то код или трансляцию можно передавать и обрабатывать, даже если у них какие-то атрибуты могут быть некорректными;

- detail — список отдельных ошибок и/или предупреждений.

8.3.2.2 Описание метода validateCode

Метод validateCode определяет, является ли значение кодированного атрибута допустимым представлением понятия для заданного словарного домена и прикладного контекста. Если параметр errorCheckOnly имеет значение false, то этот метод также проверяет поля codeSystemName и displayName, если они присутствуют. Он рекурсивно проверяет квалификаторы, используя те же самые критерии. Коды понятий, у которых заполнено поле исходного текста originalText, а поле code отсутствует, всегда признаются ошибочными. Метод validateCode не проверяет трансляции.

Входные параметры:

- vocabularyDomain_name — имя словарного домена, используемого для переданного кодированного атрибута;

- codeToValidate — подлежащий проверке код понятия, определенный в стандарте HL7. Как минимум этот код должен содержать поле кода и поле системы кодирования;

- applicationContext_code — контекст или «область применения», где должен использоваться код;

- activeConceptsOnly — значение TRUE (используемое по умолчанию) указывает, что допустимыми считаются только коды понятий, которые в настоящее время активны. Значение FALSE указывает, что код считается допустимым все то время, пока он присутствует в системе кодирования;

- errorCheckOnly — значение TRUE указывает, что код понятия проверяется только на наличие ошибок. Предупреждения в этом случае не возвращаются.

Исключения:

- UnknownVocabularyDomain;

- UnknownApplicationContextCode;

- UnexpectedError.

**8.3.3 Проверка трансляций кодированного атрибута**

8.3.3.1 Метод validateTranslation

Метод validateTranslation проверяет основной код и все трансляции кодированного атрибута, определенного в стандарте HL7. Обнаруженные ошибки и предупреждения возвращаются в экземпляре класса ValidateCodeReturn.

Примечание — В настоящем стандарте не приводятся детальные сведения о том, что является правильной трансляцией. Предполагается, что существует некоторая внешняя сторона, задающая правила трансляции, и что метод validateTranslation обеспечивает стандартный способ доступа к этим правилам. Модель трансляции описана в разделе 11.

Входные параметры:

- vocabularyDomain_name — имя словарного домена, используемого для переданного кодированного атрибута;

- codeToValidate — подлежащий проверке код понятия, определенный в стандарте HL7. Как минимум этот код должен содержать поле кода и поле системы кодирования;

- applicationContext_code — контекст или «область применения», где должен использоваться код;

- activeConceptsOnly — значение TRUE (используемое по умолчанию) указывает, что допустимыми считаются только коды понятий, которые в настоящее время активны. Значение FALSE указывает, что код считается допустимым все то время, пока он присутствует в системе кодирования;

- errorCheckOnly — значение TRUE указывает, что код понятия проверяется только на наличие ошибок. Предупреждения в этом случае не возвращаются.

Исключения:
- UnknownVocabularyDomain;
- UnknownApplicationContextCode;
- UnexpectedError.

8.3.3.2 Коды возврата методов validateCode и validateTranslation

В таблице 13 перечислены идентификаторы и текст ошибок, которые могут быть возвращены методами validateCode и validateTranslation.

Таблица 13 — Коды возврата методов validateCode и validateTranslation

| Идентификатор | Тип | Текст | Описание |
|---|---|---|---|
| E001 | E | Неизвестная система кодирования | Служба не распознала систему кодирования |
| E002 | E | Код понятия недопустим для системы кодирования | Код понятия недопустим для заданной системы кодирования (или подразумеваемой в случае типа данных CS) |
| E003 | E | Система кодирования недопустима для словарного домена | Служба распознала систему кодирования, но эта система недопустима для словарного домена и прикладного контекста |
| E004 | E | Код понятия неактивен | Код понятия допустим для домена, но не является активным. Эта ошибка возникает только в том случае, если параметр activeConceptsOnly имеет значение TRUE |
| E005 | E | Код понятия недопустим для словарного домена | Код понятия допустим для системы кодирования, но не разрешен для словарного домена и прикладного контекста |
| E008 | E | В данном словарном домене имя роли недопустимо | Система кодирования и код понятия роли квалификатора правильны, но недопустимы для словарного домена |
| E009 | E | В данном словарном домене имя роли должно присутствовать | Присутствовал квалификатор, не имевший компонента имени, требуемого для данного словарного домена |
| E010 | E | Недопустимое значение квалификатора | Квалификатором является правильный код понятия, но этот код недопустим в контексте квалификатора |
| E011 | E | Недопустимая трансляция | Система кодирования и код понятия трансляции правильны, но данная трансляция недопустима для содержащего ее кода понятия |
| E013 | E | Отсутствует код понятия | Поле кода понятия пустое |
| E014 | E | Неизвестный код обоснования кодирования | Код обоснования кодирования не распознан |
| W002 | W | Имя системы кодирования не совпадает с системой кодирования | Имя системы кодирования не соответствует идентификатору системы кодирования |
| W003 | W | Неизвестная версия системы кодирования | Версия заданной системы кодирования не распознана |
| W004 | W | Неверное имя представления кода понятия | Неверное имя представления переданного кода понятия |
| W005 | W | Отсутствует трансляция, обоснованная HL7 | Ни одна из трансляций не имеет код обоснования SH (источник и HL7) или HL7 |
| W006 | W | Код понятия не активен | Переданный код понятия не активен, при этом параметр activeConceptsOnly имеет значение TRUE |

**8.3.4 Трансляция кодированного атрибута**

Метод translateCode преобразует значение параметра fromCode в код понятия, если таковой имеется, который допустим для целевой системы кодирования или прикладного контекста. Он возвращает полную копию значения параметра fromCode с новой трансляцией (если таковая имеется), добавленной в конец последовательности трансляций CD.translation. Если значение fromCode уже содержит трансляцию, допустимую для целевой системы кодирования или прикладного контекста, то возвращаемая копия параметра fromCode будет совпадать с исходной.

Входные параметры:

- vocabularyDomain_name — словарный домен кодированного атрибута, подлежащего трансляции;
- fromCode — транслируемый код;
- toCodeSystem_id (не обязателен) — если этот параметр указан, то код должен быть транслирован в код понятия (или коды понятий), взятый из системы кодирования, идентифицированной этим параметром, а не из системы кодирования, определяемой целевым контекстом;
- toApplicationContext_code (не обязателен) — прикладной контекст транслированного кода. Используется для определения целевого набора значений, по которому, в свою очередь, определяется целевая система кодирования. Может быть указан только один из параметров toCodeSystem_id и targetContext. Если ни одного из них не указано, то генерируется исключение UnableToTranslate.

Метод translateCode возвращает кодированный атрибут, определенный в стандарте HL7, транслированный в термины целевой системы кодирования, которая должна быть явно задана в параметрах вызова либо определена по значению контекста targetContext.

Исключения:

- UnknownVocabularyDomain;
- UnknownCodeSystem;
- UnknownApplicationContextCode;
- UnableToTranslate;
- UnexpectedError.

**8.3.5 Заполнение детальных сведений о кодируемом атрибуте**

Метод fillInDetails возвращает полную копию значения параметра codeToFillIn, в которую добавлены поля codeSystemName, codeSystemVersion и displayName, заполненные по базовому значению кода и его квалификаторам, если таковые имеются. Квалификаторы заполняются рекурсивно — если квалификаторы вложены или имеют другие квалификаторы, то детальные сведения о них также заполняются. Метод fillInDetails не изменяет трансляции кода.

Входные параметры:

- codeToFillIn — кодированный атрибут, о котором заполняются сведения;
- displayLanguage_code — язык, используемый для изображаемого имени или имен.

Исключения:

- UnknownCodeSystem;
- UnknownConceptCode;
- UnknownLanguage;
- UnexpectedError.

**8.3.6 Проверка, является ли один кодированный атрибут категорией другого**

Метод subsumes определяет, является ли родительский кодированный атрибут категорией дочернего атрибута (то есть вытекает из него). Если ни параметр parentCode, ни параметр childCode не имеют квалификаторов и оба взяты из одной и той же системы кодирования, то метод subsumes возвращает значение TRUE в том и только в том случае, если значение параметра childCode является транзитивным замыканием графа отношений, корнем которого является значение параметра parentCode. В настоящем стандарте не делается никаких дальнейших предположений о семантике категоризации сверх уже сказанного.

Если служба поддерживает определение категоризации для тех случаев, когда кодированные значения имеют квалификаторы и/или принадлежат разным системам кодирования, то она должна обеспечивать необходимую семантическую трансляцию. Если она не поддерживает определение категоризации кодированных значений, имеющих квалификаторы, то при наличии квалификаторов у параметра parentCode или childCode она должна генерировать исключение QualifiersNotSupported (квалификаторы не поддерживаются). Аналогично, если служба поддерживает определение категоризации кодированных значений, принадлежащих разным системам кодирования, то она должна генерировать исключение SubsumptionNotSupported (категоризация не поддерживается).

Параметры:

- parentCode — кодированный атрибут, предполагаемый родителем;
- childCode — кодированный атрибут, предполагаемый его потомком.

Метод subsumes должен возвращать значение TRUE, если значение параметра childCode является подтипом значения атрибута parentCode. Он должен также возвращать значение TRUE, если значения параметров childCode и parentCode эквивалентны. Трансляции этим методом игнорируются.

Исключения:

- UnknownCodeSystem;
- UnknownCode;
- SubsumptionNotSupported;
- UnrecognizedQualifier;
- QualifiersNotSupported;
- UnexpectedError.

**8.3.7 Проверка, являются ли два кодированных атрибута эквивалентными**

Метод areEquivalent определяет, представляют ли два переданных ему кода эквивалентное понятие с точки зрения службы. В том случае, если входные параметры принадлежат разным системам кодирования или хотя бы один из них имеет квалификаторы, параметры code1 и code2 считаются эквивалентными в том и только в том случае, если значение параметра code1 является подтипом значения параметра code2, а значение параметра code2 является подтипом значения параметра code1. Семантика категоризации определена в предыдущем пункте.

Параметры:

- code1 — первый код, проверяемый на эквивалентность;
- code2 — второй проверяемый код.

Исключения:

- UnknownCodeSystem;
- UnknownCode;
- SubsumptionNotSupported;
- UnrecognizedQualifier;
- QualifiersNotSupported;
- UnexpectedError.

**8.3.8 Раскрытие словарного домена**

8.3.8.1 Структура класса ValueSetDescriptor (описатель набора значений)

Каждый набор значений имеет уникальный идентификатор. Кроме того, некоторые наборы значений могут также иметь необязательные имена или мнемонику, которые, будучи присвоены, также должны быть уникальными:

- valueSet_id — уникальный идентификатор набора значений;
- valueSet_name — уникальное имя набора значений (необязательное).

8.3.8.2 Структура класса ValueSetExpansion (раскрытие набора значений)

Класс ValueSetExpansion имеет следующие поля:

- pathLength — целое значение, определяющее расстояние от корневого набора значений. У корневого набора значений это поле всегда имеет значение 0;

- nodeType_code — код понятия, взятый из системы кодирования HL7 ConceptGenerality (2.16.840.1.113883.5.24). Допустимы следующие коды:

«S» — специализируемый узел. Идентификатор concept_id этого узла может быть выбран, но узел может иметь дальнейшее раскрытие;

«A» — абстрактный узел. Идентификатор concept_id этого узла (если имеется) не может быть выбран. Чтобы можно было сделать выбор, к этому узлу должно быть применено дальнейшее раскрытие;

«L» — конечный узел (лист). Идентификатор concept_id этого узла может быть выбран, но дальнейшее раскрытие невозможно;

- valueSet — идентификатор и имя (если имеется) набора значений, ассоциированного с данным узлом;

- concept_id — система кодирования и код понятия, ассоциированные с данным узлом (если имеются);

- displayName — изображаемое имя, используемое для представления этого узла;

- isExpandable — значение TRUE указывает, что этот узел можно далее раскрыть с помощью функции expandValueSetExpansionContext. Поле isExpandable будет иметь значение TRUE только для

специализируемых и абстрактных узлов, и то только в том случае, если в исходном вызове метода lookupValueSetExpansion параметр expandAll (раскрыть все) имел значение FALSE;

- expansionContext — если параметр isExpandable имеет значение TRUE, то параметр expansionContext содержит те сведения, которые необходимы конкретной реализации службы для дальнейшего раскрытия узла при следующих вызовах. Он непрозрачен для клиентского программного обеспечения.

Ниже приведены варианты раскрытия различных типов наборов значений.

Вариант 1 (таблица 14): набор значений A ссылается на все коды неиерархической системы кодирования, содержащей коды понятий 1,2, ..., N.

Таблица 14 — Значения полей класса ValueSetExpansion для варианта 1

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| 0 | A (абстрактный) | — | (A).valueSet_name |
| 1 | L (лист) | 1 | (1).preferredName |
| 1 | L (лист) | 2 | (2).preferredName |
| 1 | L (лист) | … | … |
| 1 | L (лист) | N | (N).preferredName |

Вариант 2 (таблица 15): набор значений B ссылается на все коды иерархической системы кодирования. Коды понятий в поле concept_id отражают иерархию (1 — корень, 1.1 — первый потомок корня, 1.2 — второй потомок, 1.2.1 — первый потомок первого потомка и т. д.).

Таблица 15 — Значения полей класса ValueSetExpansion для варианта 2

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| 0 | A (абстрактный) | — | (B).valueSet_name |
| 1 | S (специализируемый) | 1 | (1).preferredName |
| 2 | S (специализируемый) | 1.1 | (1.1).preferredName |
| … | S (специализируемый) | … | … |
| n | L (лист) | 1.1.1..n | (1.1.1.n).preferredName |
| 1 | S (специализируемый) | 2 | (2).preferredName |
| 2 | S (специализируемый) | 2.1 | (2.1) preferredName |
| … | S (специализируемый) | … | … |
| M | L (лист) | 2.1..m | (2.1..m).preferredName |

Вариант 3 (таблица 16): набор значений C ссылается на конкретные коды понятий 1, 2, 3 из некоторой системы кодирования. Ни одна из ссылок не включает в себя код отношения. Головного кода также нет.

Таблица 16 — Значения полей класса ValueSetExpansion для варианта 3

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| | A (абстрактный) | — | €.valueSetName |
| 1 | L (лист) | 1 | (1).preferredName |
| 1 | L (лист) | 2 | (2).preferredName |
| 1 | L (лист) | 3 | (3).preferredName |

Вариант 4 (таблица 17): набор значений D ссылается на конкретные коды понятий 1, 2, 3, 4 из некоторой системы кодирования. Ни одна из ссылок не включает в себя код отношения. Код понятия 4 является головным.

Таблица 17 — Значения полей класса ValueSetExpansion для варианта 4

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| 0 | A (абстрактный) | 4 | (4).preferredName |
| 1 | L (лист) | 1 | (1).preferredName |
| 1 | L (лист) | 2 | (2).preferredName |
| 1 | L (лист) | 3 | (3).preferredName |

Вариант 5 (таблица 18): набор значений E ссылается на описанный выше набор значений D, но при этом поле includeHeadCode (включить головной код) имеет значение TRUE.

Таблица 18 — Значения полей класса ValueSetExpansion для варианта 5

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| 0 | A (абстрактный) | — | €.valueSet_name |
| 1 | S (специализируемый) | 4 | (4).preferredName |
| 2 | L (лист) | 1 | (1).preferredName |
| 2 | L (лист) | 2 | (2).preferredName |
| 2 | L (лист) | 3 | (3).preferredName |

Вариант 6 (таблица 19): набор значений F ссылается на описанный выше набор значений D, но при этом поле includeHeadCode (включить головной код) имеет значение FALSE.

Таблица 19 — Значения полей класса ValueSetExpansion для варианта 6

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| 0 | A (абстрактный) | — | (F).valueSet_name |
| 1 | A (абстрактный) | 4 | (4).preferredName |
| 2 | L (лист) | 1 | (1).preferredName |
| 2 | L (лист) | 2 | (2).preferredName |
| 2 | L (лист) | 3 | (3).preferredName |

Вариант 7 (таблица 20): набор значений G ссылается на коды понятий 1.1, 2.1 и 3.1 из иерархической системы кодирования. Все три ссылки используют код отношения hasSubtype. Ссылка на код 1.1 включает ссылочный код. Ссылка на код 1.2 исключает ссылочный код. У ссылки 3.1 поле leafOnly имеет значение TRUE. Набор значений G не имеет головного кода.

Таблица 20 — Значения полей класса ValueSetExpansion для варианта 7

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| 0 | A (абстрактный) | — | (G).valueSet_name |
| 1 | S (специализируемый) | 1.1 | (1.1).preferredName |
| 2 | S (специализируемый) | 1.1.1 | (1.1.1).preferredName |

*Окончание таблицы 20*

| pathLength | nodeType_code | concept_id | displayName |
|---|---|---|---|
| … | S (специализируемый) | 1.1…. | … |
| n | L (лист) | 1.1….n | (1.1….n).preferredName |
| 1 | A (абстрактный) | 1.2 | (1.2) preferredName |
| 2 | S (специализируемый) | 1.2.1 | (1.2.1)preferredName |
| … | S (специализируемый) | 1.2.1… | … |
| m | L (лист) | 1.2.1…m | (1.2…m)preferredName |
| 1 | A (абстрактный) | 1.3 | (1.3) preferredName |
| 2 | A (абстрактный) | 1.3.1 | (1.3.1) preferredName |
| … | A (абстрактный) | 1.3.1… | … |
| k | L (лист) | 1.3.1…k | (1.3.1..k)preferredName |

8.3.8.3 Метод lookupValueSetExpansion

Метод lookupValueSetExpansion возвращает раскрытие набора значений, ассоциированного с заданным словарным доменом и необязательным контекстом. Весь словарный домен может быть раскрыт сразу (если параметр expandAll имеет значение TRUE) или может быть раскрыт на один уровень за один раз (если параметр expandAll имеет значение FALSE). В последнем случае каждый узел может быть раскрыт далее, если у него поле isExpandable имеет значение TRUE, указывающее, что поле expansionContext может быть передано методу expandValueSetExpansionContext для дальнейшего раскрытия.

Параметры.

- vocabularyDomain_name — раскрываемый словарный домен;
- applicationContext_code — контекст, в котором должен раскрываться словарный домен (не обязателен);
- language_code — код языка, используемый для возвращаемых изображаемых имен;
- expandAll — значение TRUE указывает, что все узлы раскрываются на их полную глубину, значение FALSE означает раскрытие на один уровень (по умолчанию используется значение TRUE);
- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- UnknownVocabularyDomain;
- UnknownApplicationContextCode;
- UnknownLanguage;
- TimeoutError;
- UnexpectedError.

8.3.8.4 Метод expandValueSetExpansionContext

Метод expandValueSetExpansionContext использует непрозрачное значение параметра expansionContext, которое ранее было возвращено в экземпляре класса ValueSetExpansion, для дальнейшего раскрытия соответствующего узла на дополнительный уровень. Возвращаемое значение идентично тому, что возвращает метод lookupValueSetExpansion, и к вызову применяются все начальные ограничения последнего, включая значение параметра timeout.

Входные параметры:

- expansionContext — контекст, возвращенный предшествующим вызовом метода lookupValueSetExpansion или expandValueSetExpansionContext.

Исключения:
- InvalidExpansionContext;
- TimeoutError;
- UnexpectedError.

### 8.4 API обозревателя на уровне сообщений

Вторая часть API сообщений ОТС содержит комплекс функций, которые могут использоваться для изучения и просмотра атрибутов, словарных доменов и наборов значений.

#### 8.4.1 Идентифицирующая информация обозревателя сообщений

Обозреватель сообщений наследует идентифицирующую информацию от интерфейса, описанного в 8.2.3 «Идентифицирующий раздел службы».

#### 8.4.2 Раздел описания обозревателя сообщений

В этом пункте описано несколько структур, служащих «строительными блоками» и используемых при описании службы. Затем будет описан доступ к информации с помощью этой службы.

8.4.2.1 Строительные блоки описания обозревателя

8.4.2.1.1 Структура RIMCodedAttribute

Класс RIMAttributeId уникально идентифицирует кодированный атрибут модели RIM. Он имеет следующие поля:
- model_id — идентификатор модели RIM;
- class_name — имя класса модели RIM;
- attribute_name — имя атрибута в этом классе.

Класс RIMCodedAttribute описывает атрибут модели RIM. Он имеет следующие поля:
- RIMAttribute_id — уникальный идентификатор атрибута;
- dataType_code — код, идентифицирующий тип данных атрибута. Берется из системы кодирования HL7 DataType;
- codingStrength_code — сила кодирования атрибута. Берется из системы кодирования HL7 VocabularyDomainQualifier;
- vocabularyDomain_name — имя словарного домена, ассоциированного с атрибутом.

8.4.2.1.2 Класс CodeSystemDescriptor

Класс CodeSystemDescriptor состоит из идентификатора системы кодирования, ее имени, информации об авторских правах и списка доступных версий:
- codeSystem_id — объектный идентификатор ИСО (ОИД) системы кодирования;
- codeSystem_name — имя системы кодирования. Уникально в контексте стандарта HL7 Версии 3;
- copyright — информация об авторских правах на систему кодирования, если таковые имеются. Если присутствует, то эта информация должна предоставляться всякий раз, когда система кодирования используется или к ней осуществляется доступ;
- availableReleases — выпуск системы кодирования, которая на момент вызова поддерживается службой. В текущей версии настоящего документа предполагается, что для любой системы кодирования можно запросить не более одного выпуска. В будущих версиях это ограничение будет снято.

8.4.2.2 Описание обозревателя

Описание обозревателя сообщений содержит список различных сущностей, поддерживаемых конкретной службой.

Метод getSupportedAttributes возвращает список атрибутов модели RIM, известных службе и удовлетворяющих заданным критериям.

Входные параметры:
- matchText — если присутствует и не пуст, то возвращаются сведения только о тех атрибутах модели RIM, имена которых совпадают с текстом, переданным в этом параметре. Если параметр matchText отсутствует или пуст, то возвращаются сведения обо всех атрибутах модели RIM;
- matchAlgorithm_code — если параметр matchText присутствует и не пуст, то значение параметра matchAlgorithm_code указывает, каким образом определяется совпадение имени атрибута со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»,
- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные эле-

менты, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- BadlyFormedMatchText;
- UnknownMatchAlgorithm;
- TimeoutError;
- UnexpectedError.

Метод getSupportedVocabularyDomains возвращает список словарных доменов, известных службе и удовлетворяющих заданным критериям.

Входные параметры:

- matchText — если присутствует и не пуст, то возвращаются сведения только о тех словарных доменах, имена которых совпадают с текстом, переданным в этом параметре. Если параметр matchText отсутствует или пуст, то возвращаются сведения обо всех словарных доменах;
- matchAlgorithm_code — если параметр matchText присутствует и не пуст, то значение параметра matchAlgorithm_code указывает, каким образом определяется совпадение имени словарного домена со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»;
- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- BadlyFormedMatchText;
- UnknownMatchAlgorithm;
- TimeoutError;
- UnexpectedError.

Метод getSupportedValueSets возвращает список наборов значений, известных службе и удовлетворяющих заданным критериям.

Параметры:

- matchText — если присутствует и не пуст, то возвращаются сведения только о тех наборах значений, имена которых совпадают с текстом, переданным в этом параметре. Если параметр matchText отсутствует или пуст, то возвращаются сведения обо всех наборах значений;
- matchAlgorithm_code — если параметр matchText присутствует и не пуст, то значение параметра matchAlgorithm_code указывает, каким образом определяется совпадение имени набора значений со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»;
- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- BadlyFormedMatchText;
- UnknownMatchAlgorithm;
- TimeoutError;
- UnexpectedError.

Метод getSupportedCodeSystems возвращает список систем кодирования, известных службе и удовлетворяющих заданным критериям.

Параметры.

- matchText — если присутствует и не пуст, то возвращаются сведения только о тех системах кодирования, имена которых совпадают с текстом, переданным в этом параметре. Если параметр matchText отсутствует или пуст, то возвращаются сведения обо всех системах кодирования;
- matchAlgorithm_code — если параметр matchText присутствует и не пуст, то значение параметра matchAlgorithm_code указывает, каким образом определяется совпадение имени системы кодирования со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»;

- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;

- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- BadlyFormedMatchText;
- UnknownMatchAlgorithm;
- TimeoutError;
- UnexpectedError.

**8.4.3 Обозревание словарного домена**

8.4.3.1 Класс VocabularyDomainDescription

Класс VocabularyDomainValueSet содержит описание набора значений (см. 8.3.8 «Раскрытие словарного домена») и код контекста, в котором применяется этот набор значений (если таковой контекст имеется). Он имеет следующие поля:

- definedByValueSet — набор значений, определяющий словарный домен в конкретном прикладном контексте;
- applicationContext_code — код контекста, в котором применяется этот набор значений.

Класс VocabularyDomainDescription описывает структуру, возвращаемую методом lookupVocabularyDomain. Он имеет следующие поля:

- vocabularyDomain_name — уникальное имя словарного домена;
- description — описание домена;
- restrictsDomain_name — словарный домен, ограниченный данным доменом, если таковой имеется;
- basisOfDomains — список доменов, являющихся ограничениями данного домена, если таковые имеются;
- constrainsAttributes — список атрибутов модели RIM, использующих данный словарный домен, если таковые имеются;
- representedByValueSets — список наборов значений и контекст, которые могут представлять данный словарный домен, если таковые имеются.

8.4.3.2 Метод lookupVocabularyDomain

Метод lookupVocabularyDomain возвращает детальные сведения о словарном домене. Он имеет следующий входной параметр:

- vocabularyDomain_name — имя обозреваемого словарного домена.

Исключения:

- UnknownVocabularyDomain;
- UnexpectedError.

**8.4.4 Обозревание набора значений**

8.4.4.1 Класс FullValueSetDescriptionstructure

Класс FullValueSetDescription включает в себя полное описание набора значений, в том числе список включенных в него кодов понятий и других наборов значений.

Класс ValueSetConstructor имеет следующие поля:

- includedValueSet — идентификатор и имя набора значений, считающегося частью данного набора, если таковой имеется;
- includeHeadCode — значение TRUE указывает, что головной код набора значений, включенного в данный набор (если таковой имеется), входит в состав данного набора. Значение FALSE указывает, что не входит.

Класс ValueSetCodeReference имеет следующие поля:

- referenced_code — код понятия, на который ссылается данный набор;
- relationship_code — код отношения. Если присутствует, то все потомки ссылочного кода также считаются частью набора значений, с поправкой на ограничение, задаваемое описанным ниже параметром leafOnly. Если отношение транзитивно, то все потомки также включаются в набор значений. Если оно не транзитивно, то рассматриваются только прямые потомки ссылочного кода;
- includeReferencedCode — значение TRUE указывает, что сам ссылочный код является частью набора значений. Значение FALSE указывает, что в набор значений включаются только его дети или

потомки. Парметр includeReferencedCode обязательно имеет значение TRUE, если код отношения relationship_code не задан;

- leafOnly — значение TRUE указывает, что в набор значений включаются только конечные узлы (листья) отношения. Значение FALSE указывает, что в набор значений включаются все потомки, возможно, за исключением самого ссылочного кода. Параметр leafOnly обязательно должен иметь значение FALSE, если код отношения relationship_code не задан.

Класс ValueSetDescription имеет следующие поля:

- idAndName — идентификатор и имя (если таковое имеется) набора значений;
- description — текстовое описание набора значений и его применения;
- definingExpression — выражение, используемое для построения содержания набора значений (если таковое имеется);
- basedOnCodeSystem — идентификатор, имя и версия системы кодирования, используемой для построения набора значений;
- allCodes — значение TRUE указывает, что все коды данной системы кодирования включаются в набор значений. Если значение этого параметра не равно TRUE, то набор значений не должен ссылаться на какие-либо дополнительные коды;
- head_code — код понятия, представляющий весь набор значений (если таковой имеется).

Класс FullValueSetDescription имеет следующие поля:

- description — идентификатор, имя и другие атрибуты набора значений;
- constructedUsingValueSets — список дополнительных наборов значений и включений головных кодов, если таковые имеются, используемых для построения данного набора значений;
- usedToDefine — список других наборов значений, при построении которых используется данный набор значений,
- referencesCodes — список кодов понятий, на которые в данном наборе значений имеются прямые ссылки. Он будет пустым, если параметр allCodes имеет значение TRUE, и не будет содержать те коды, которые косвенно включены из других наборов значений и/или отношений между кодами понятий.

8.4.4.2 Метод lookupValueSet

Метод lookupValueSet извлекает полное описание набора значений по идентификатору или имени набора.

Входные параметры:

- valueSet_id — идентификатор обозреваемого набора значений;
- valueSet_name — имя обозреваемого набора значений.

Должны быть заданы идентификатор набора значений, либо имя, либо оба параметра. В последнем случае имя должно соответствовать идентификатору, иначе будет сгенерировано исключение.

Исключения:

- UnknownValueSet;
- ValueSetNameIdMismatch;
- UnexpectedError.

**8.4.5 Обозревание детальной информации о системе кодирования**

8.4.5.1 Структура данных, возвращаемая методом lookupCodeSystem

Класс CodeSystemRegistration имеет следующие поля:

- sponsor — член комитета HL7 или организация, выступающая спонсором регистрации системы кодирования;
- publisher — наименование официального издателя системы кодирования;
- versionReportingMethod — описание того, как и сколь часто создаются и распространяются новые версии;
- licensingInformation — описание требуемых лицензий, цены и способа приобретения;
- inUMLS — значение TRUE указывает, что кодированные термины, определенные в этой системе кодирования, включены в Унифицированную систему медицинского языка UMLS (Unified Medical Language System);
- systemSpecificLocatorInfo — информация, специфичная для издателя системы кодирования. Она служит для определения или идентификации конкретной системы кодирования. Комитет HL7 может иногда использовать атрибут systemSpecificLocatorInfo для идентификации соответствующей таблицы стандарта HL7 2.x, если таковая имеется;

- codeSystemType_code — код, имеющий значения I (система кодирования является внутренней, ведется и распространяется комитетом HL7), E (ведется и распространяется третьей стороной), EI (ведется третьей стороной, но ради удобства комитет HL7 поддерживает ее внутренний образ).

Примечание — Перечисленные выше регистрационные элементы предназначены только для описания системы кодирования. Многие из полей (спонсор, издатель, метод публикации версий и т. д.) могут иметь собственную структуру, но в целях настоящего документа описание их структуры не является необходимым.

Класс CodeSystemInfo имеет следующие поля:

- description — базовое описание системы кодирования. См. описание класса CodeSystemDescriptor в 8.4.2.1.2;
- registrationInfo — регистрационная информация о системе кодирования (если таковая имеется).

8.4.5.2 Метод lookupCodeSystem

Метод lookupCodeSystem возвращает детальную информацию о системе кодирования по ее идентификатору (ОИД) или имени.

Входные параметры:

- codeSystem_Id — уникальный идентификатор системы кодирования, которым обычно служит ИСО ОИД;
- codeSystem_name — имя системы кодирования.

Должны быть заданы идентификатор системы кодирования, либо имя, либо оба параметра. В последнем случае имя должно соответствовать идентификатору, иначе будет сгенерировано исключение.

Исключения:

- UnknownCodeSystem;
- CodeSystemNameIdMismatch;
- UnexpectedError.

**8.4.6 Обозревание наборов значений в словарном домене и контексте**

Метод lookupValueSetForDomain возвращает идентификацию набора значений (идентификатор и имя, если таковое имеется), использованного для заданного словарного домена в прикладном контексте (если таковой имеется).

Входные параметры:

- vocabularyDomain_name — имя обозреваемого словарного домена;
- applicationContext_code (не обязателен) — код прикладного контекста.

Исключения:

- UnknownVocabularyDomain;
- UnknownApplicationContextCode;
- NoApplicableValueSet;
- UnexpectedError.

**8.4.7 Установление принадлежности кода понятий к набору значений**

Метод isCodeInValueSet возвращает значение TRUE, если понятие с заданным идентификатором включено в набор значений и может быть выбрано из него. В противном случае возвращается значение FALSE.

Входные параметры:

- valueSet_id — идентификатор обозреваемого набора значений;
- valueSet_name — имя обозреваемого набора значений;
- includeHeadCode — значение TRUE указывает, что головной код (если таковой имеется) считается частью данного набора. Значение FALSE указывает, что головной код (если таковой имеется), исключается из набора;
- codeToValidate — код проверяемого понятия и идентификатор его системы кодирования.

Должны быть заданы идентификатор набора значений, либо имя, либо оба параметра. В последнем случае имя должно соответствовать идентификатору, иначе будет сгенерировано исключение.

Исключения:

- UnknownValueSet;
- ValueSetNameIdMismatch;
- UnknownConceptCode;
- UnknownCodeSystem;
- UnexpectedError.

# 9 Модель API словаря

## 9.1 Введение

В следующих подразделах описана модель, положенная в основу API словаря ОТС. В настоящем стандарте не представлена полная или глубокая модель всех возможных сущностей, образующих систему кодирования[1)]. Его основная цель — описать классы и отношения, имеющие прямое отношение к содержанию кодированных атрибутов HL7 с точки зрения словаря.

## 9.2 Система кодирования

Структура классов, образующих систему кодирования, представлена на рисунке 6.

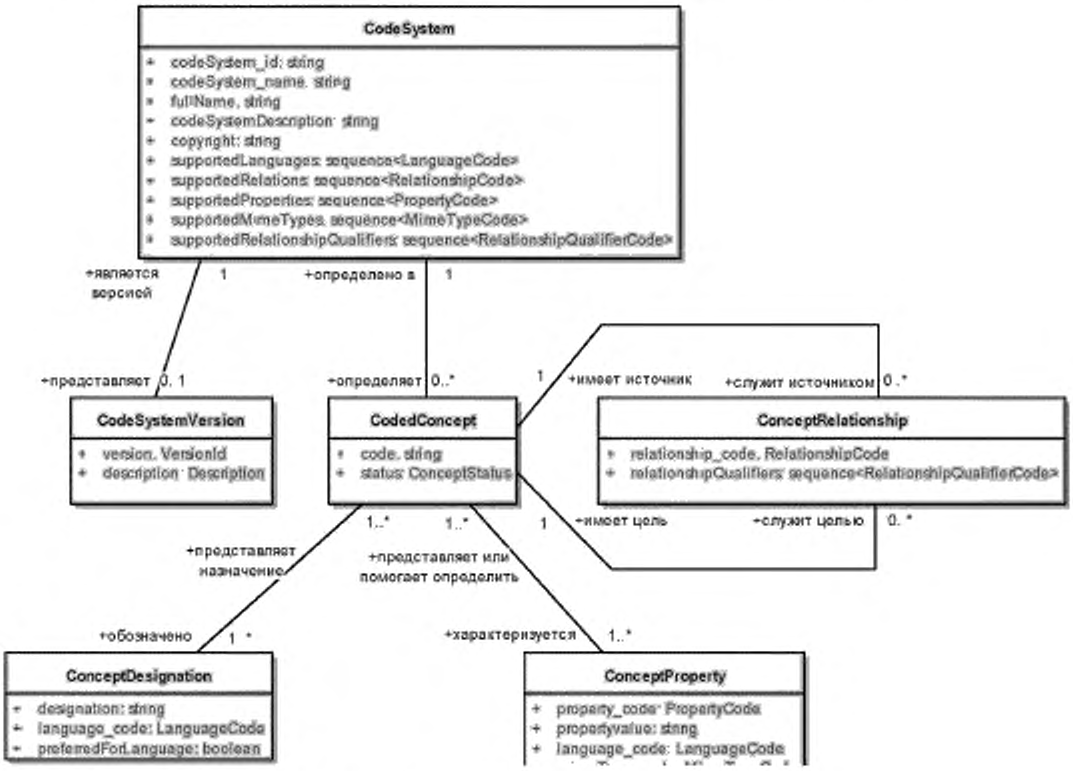


Рисунок 6 — Классы, образующие систему кодирования

Система кодирования **CodeSystem** может определять нуль или более кодированных понятий **CodedConcept**. Кодированное понятие представляет класс объектов или понятие в конкретной области суждений. Каждое кодированное понятие **CodedConcept** должно быть определено ровно в одной системе кодирования **CodeSystem**. Будучи однажды определенным, значение кодированного понятия не может изменяться. Существующие кодированные понятия могут отзываться, а новые — добавляться, но, будучи определенным, значение кодированного понятия должно оставаться статическим.

Класс **CodeSystem** имеет следующие атрибуты:

- *codeSystem_id* — глобально уникальный идентификатор системы кодирования. В контексте стандартов HL7 он должен принимать форму объектного идентификатора ИСО (ОИД). Комитет HL7

[1)] В настоящем стандарте термин «система кодирования» используется для обозначения системы кодов, описаний, обозначений, свойств и отношений. К другим общим именам этой сущности относятся «словарь», «терминология», «схема кодирования», «схема классификации» и «онтология».

ведет регистр идентификаторов ОИД систем кодирования, и пользователям стандарта настоятельно рекомендуется регистрировать в этом регистре все ОИД, используемые ими в этих службах;

- *codeSystem_name* — краткая символьная метка, уникально идентифицирующая систему кодирования в контексте модели HL7 RIM. Комитет HL7 включает имена систем кодирования в «систему кодирования систем кодирования», ассоциированную со словарным доменом **CodeSystem**. Атрибут *codeSystem_name* используется исключительно в целях коммуникации углеродных форм жизни, а при взаимодействии компьютеров должен использоваться атрибут *codeSystemId*;

- *fullName* — официальное имя системы кодирования **CodeSystem**. Для тех систем, что зарегистрированы в словарном домене HL7 **CodeSystem**, значением этого имени является предпочтительное английское обозначение **ConceptDesignation** кодированного понятия **CodedConcept**, совпадающего с атрибутом *codeSystem_name* системы кодирования **CodeSystem**;

- *codeSystemDescription* — описание назначения и содержания системы кодирования **CodeSystem**. Для тех систем, что зарегистрированы в словарном домене HL7 **CodeSystem**, значением этого имени является английское описание **ConceptDescription** кодированного понятия **CodedConcept**, совпадающего с атрибутом *codeSystem_name* системы кодирования **CodeSystem**;

- *copyright* — необязательная информация об авторских правах на систему кодирования. Если присутствует, то эта информация должна предоставляться всякий раз, когда система кодирования используется или к ней осуществляется доступ.

Следующие атрибуты содержат дополнительные метаданные системы кодирования:

- *supportedLanguages* — список всех языков, которые полностью или частично поддерживаются системой кодирования. «Поддерживаемый» язык распознается системой кодирования и на нем доступна хотя бы часть обозначений или свойств понятий. Все системы кодирования должны поддерживать хотя бы один язык. В то время как служба должна возвращать перечисление всех субтегов основного языка, который она поддерживает, она не обязана перечислять вторичные языки. (Например, если она поддерживает язык «en-UK», она должна возвращать тег «en», но может возвращать или не возвращать «en-UK»);

- *supportedRelations* — отношения (роли), представленные в системе кодирования. В данной модели отношение подтипа (subtype) трактуется как отношение первого класса (код отношения: hasSubtype). Код отношения имеет и идентификатор системы кодирования, и код понятия, что позволяет брать коды отношений из многих источников. Для целей интероперабельности коды отношений должны по возможности браться из системы кодирования HL7 ConceptRelationship (ОИД 2.16.840.1.113883.5.1088);

- *supportedProperties* — коды свойств, поддерживаемых системой кодирования. Однако по возможности коды свойств должны браться из системы кодирования HL7 ConceptProperty (ОИД 2.16.840.1.113883.5.1087);

- *supportedMimeTypes* — список типов среды MIME, используемых системой кодирования в обозначениях, описаниях или свойствах. Эти коды должны браться из системы кодирования типов среды MediaType, официально предназначенной для этих целей комитетом HL7 (в настоящее время — ОИД 2.16.840.1.113883.5.79). Тип среды text/plain (неформатированный текст) должен поддерживаться всеми системами кодирования;

- *supportedRelationshipQualifiers* — список квалификаторов отношений, распознаваемых системой кодирования.

Система кодирования **CodeSystem** может представлять нуль или одну версию **CodeSystemVersion** в каждый момент времени[1].

### 9.3 Кодированное понятие CodedConcept

Кодированное понятие **CodedConcept** уникально в системе кодирования **CodeSystem**, где оно определено. Кодированное понятие **CodedConcept** должно быть обозначено по меньшей мере одним экземпляром класса **ConceptDesignation**. Обозначение **ConceptDesignation** должно представлять назначение одного или более кодированных понятий **CodedConcept**[2]. Кодированное понятие **CodedConcept**

[1] Возможность доступа к предыдущим выпускам версий системы кодирования будет представлена в следующих редакциях настоящего стандарта.

[2] Обозначение показано как ассоциированное более чем с одним кодированным понятием, чтобы подчеркнуть тот факт, что несколько кодированных понятий могут иметь один и тот же текст обозначения. В модели не делается никаких утверждений о том, является ли обозначение «объектом первого класса», то есть обладает ли оно идентичностью помимо текстовой части.

может характеризоваться нулем или более свойств **ConceptProperty**. Свойство **ConceptProperty** может представлять или определять одно или более кодированных понятий **CodedConcept**.

Кодированное понятие **CodedConcept** может служить источником или служить целью нуля или более отношений **ConceptRelationship**. Отношения описаны более детально в следующем подпункте.

Класс **CodedConcept** имеет следующие атрибуты:

- *code* — идентификатор, однозначно задающий класс или «понятие» в контексте определяющей системы кодирования **CodeSystem**. Будучи присвоенным, «смысл» кода понятия никогда не должен меняться. В настоящем стандарте не делается никаких предположений о тех или других способах присвоения в системе кодирования одного и того же «смысла» нескольким кодированным понятиям;

- *status* — текущий статус кодированного понятия **CodedConcept** в системе кодирования **CodeSystem**. Значения статусов понятий должны браться из системы кодирования HL7 ConceptStatus (ОИД 2.16.840.1.113883.5.1086). К возможным значениям статуса относятся «proposed» (предложен), «active» (активный), «deleted» (удален) и «retired» (устарел). В настоящем стандарте проводится только различие между статусами «active» (активный) и «not active» (неактивный). Статусы и версионность понятий будут более полно рассмотрены в следующих версиях настоящего документа.

### 9.4 Обозначение понятия ConceptDesignation

Обозначением понятия **ConceptDesignation** является имя или иной текстовый символ, представляющий назначение нуля или более кодированных понятий **CodedConcept**. Обозначения понятий **ConceptDesignation** зависят от языка.

Класс **ConceptDesignation** имеет следующие атрибуты:

- *designation* — строка текста, являющаяся внешним представление кодированного понятия **CodedConcept**;

- *languageCode* — код языка, конструируемый по правилам, описанным в IETF RFC 3066 — Tag for Identification of Languages (теги идентификации языка). Этот код состоит из нескольких субтегов, разделенных дефисами («-»). Первый субтег идентифицирует основной код языка. По возможности он должен быть взят из ИСО 639-1 «Codes for the representation of names of languages — Part 1: Alpha-2 code» (Коды для представления названий языков. Часть 1. Двухбуквенный код). Если двухбуквенный код отсутствует, то надо взять код из ИСО 639-2 «Codes for the representation of names of languages — Part 2: Alpha-3 code» (Коды для представления названий языков. Часть 2. Трехбуквенный код). Существует также дополнительный механизм спецсимволов, который в настоящем стандарте не описан.

Второй субтег не обязателен. Если он присутствует, то должен иметь длину от 2 до 8 символов. Если его длина равна двум, то он должен содержать двухбуквенный код страны, взятый из ИСО 3166-1 «Codes for the representation of names of countries and their subdivisions — Part 1: Country codes» (Коды для представления названий стран и единиц их административно-территориального деления. Часть 1. Коды стран). Если длина субтега от 3 до 8 символов, то он должен содержать код, взятый из регистра тегов языков организации IANA. Дополнительные субтеги используются, если надо уточнить информацию о языке;

- *preferredForLanguage* — значение TRUE указывает, что это обозначение следует предпочесть для представления назначения кодированного понятия **CodedConcept** на заданном языке, если иная контекстуальная информация отсутствует. Для каждого заданного языка только одно обозначение может быть указано предпочтительным.

### 9.5 Свойство понятия ConceptProperty

Свойством понятия **ConceptProperty** является «атрибут», «фасет» или иная характеристика, которая может представить или помочь определить предназначение нуля или более кодированных понятий **CodedConcept**. Класс **ConceptProperty** имеет следующие атрибуты:

- *propertyCode* — сочетание идентификаторов системы кодирования и кода понятия, идентифицирующее тип свойства. По возможности коды свойств должны браться из системы кодирования HL7 ConceptProperty (ОИД 2.16.840.1.113883.5.1087);

- *propertyValue* — текстовое значение ассоциированного свойства;

- *language_code* — код языка, взятый из системы кодирования, официально предназначенной для этих целей комитетом HL7 (в настоящее время — ОИД 2.16.840.1.113883.6.84 — IETF RFC 3066 — Tag for Identification of Languages). Не у всех свойств понятий **ConceptProperty** есть коды языка. Если код языка опущен, то предполагается, что характер этого свойства таков, что понятие языка к нему неприменимо;

- *mimeType_code* — код, взятый из системы кодирования типов среды MediaType, официально предназначенной для этих целей комитетом HL7 (в настоящее время — ОИД 2.16.840.1.113883.5.79). Не у всех свойств **ConceptProperty** есть код среды. По умолчанию используется тип среды text/plain (неформатированный текст).

### 9.6 Отношение понятий ConceptRelationship

Класс ConceptRelationship представляет бинарные отношения на множестве кодированных понятий CodedConcept, принадлежащих одной системе кодирования CodeSystem[1)]. Каждое отношение ConceptRelationship должно иметь ровно один экземпляр класса CodedConcept в качестве источника и ровно один экземпляр класса CodedConcept в качестве цели. Атрибут relationship_code идентифицирует направленное отношение. Система кодирования HL7 ConceptCodeRelationship (ОИД 2.16.840.1.113883.5.1088) определяет типы отношений, используемых в словаре стандарта HL7 Версии 3. В этой системе кодирования описано, какой код понятия является источником, какой целью, характеризуются транзитивность, симметричность и рефлексивность отношения, а также термин, используемый для представления инверсии отношения.

Список кодов отношений приведен в таблице 21.

Таблица 21 — Основные коды отношений

| Код понятия | Описание | Транзитивность | Рефлексивность | Симметричность | Инверсия |
|---|---|---|---|---|---|
| hasSubtype | Отношение подтипа, не специфицированное иным образом | Да | Нет | Нет | isSubtypeOf |
| hasPart | Отношение части к целому, не специфицированное иным образом | Да | Да | Нет | isPartOf |
| smallerThan | Общее отношение упорядочения (меньше) | Да | Нет | Нет | greaterThan |

## 10 Спецификация API словаря

### 10.1 Введение

API словаря ОТС (CTS Vocabulary API — CTSVAPI) состоит из двух разделов — раздел времени исполнения, определяющий комплекс функций, необходимых для каждодневных операций программного обеспечения, обрабатывающего сообщения, соответствующие стандарту HL7 Версии 3, и раздел обозревателя, определяющего функции, предназначенные для разработки и конструирования словаря HL7 и содержания наборов значений/словарных доменов. Предполагается, что функции раздела обозревателя имеют доступ к API времени сообщений, поэтому функциональность этого API не воспроизводится в разделе обозревателя. В обоих разделах используется описанный ниже общий комплекс определений базовых типов данных.

### 10.2 Базовые типы данных

Ниже перечислены базовые типы данных, используемые в API времени исполнения на уровне словаря, API обозревателя словаря и API трансляции:

- OID — объектный идентификатор ИСО (ОИД);
- Description — текстовое описание объекта или сущности. При реализации следует иметь в виду, что некоторые описания могут быть довольно длинными;

[1)] Приведенная в настоящем стандарте модель не полна и не отражает всю семантику систем онтологий и терминологий. Для описания семантических последствий отношений ассоциированных кодированных терминов или их экземпляров, очевидно, недостаточно объявления отношения между двумя кодированными терминами. Важно также учесть, что в настоящем стандарте не делается никаких утверждений о последствиях отсутствия декларированных отношений между двумя кодами. Привнесение дополнительной семантики оставляется на усмотрение отдельного поставщика терминологии в соответствии с его собственной моделью и содержанием.

- CTSVersionId — идентификатор версии реализации Общих терминологических служб (ОТС). В версии 1.0 поле major имеет значение 1, а поле minor — значение 0;

- CodeSystemId — уникальный идентификатор системы кодирования. В контексте стандартов HL7 им должен быть объектный идентификатор ИСО (ОИД), присвоенный комитетом HL7, если таковой имеется. Другие виды идентификаторов, например UUID, определенный для среды распределенных вычислений DCE (Distributed Computing Environment), и т. д., могут использоваться в качестве идентификаторов систем кодирования вне среды применения стандартов HL7. В этих случаях разработчик должен предупреждать любые конфликты пространств имен, которые могут возникать между ОИД и другими идентификаторами;

- CodeSystemName — краткая мнемоника или имя системы кодирования. Как идентификатор системы кодирования, так и ее имя уникальны в пространстве имен, контролируемом стандартом HL7 Версии 3, но в общем случае уникальность не гарантируется;

- ConceptCode — код, уникально представляющий класс или понятие в контексте системы кодирования;

- ConceptId — сочетание идентификаторов системы кодирования и кода понятия, являющееся глобально уникальным именем понятия;

- VersionId — уникальный идентификатор версии. Обычно идентификаторы версий имеют определенную упорядоченность, позволяющую людям или программам определить, в какой последовательности идут эти идентификаторы;

- CodeSystemVersion — идентификатор версии системы кодирования;

- ExpansionContext — непрозрачный большой двоичный объект, используемый для передачи контекстной информации между сервером и клиентом.

**10.2.1 Кодируемые элементы данных**

Ниже описаны кодируемые элементы данных, используемые в API словаря:

- LanguageCode — код разговорного или письменного языка, конструируемый по правилам, описанным в документе IETF RFC 3066 — Tag for Identification of Languages (теги идентификации языка). Этот код состоит из нескольких субтегов, разделенных дефисами («-»). Первый субтег идентифицирует основной код языка. По возможности он должен быть взят из ИСО 639-1 «Codes for the representation of names of languages — Part 1: Alpha-2 code» (Коды для представления названий языков. Часть 1. Двухбуквенный код). Если двухбуквенный код отсутствует, то надо взять код из ИСО 639-2 «Codes for the representation of names of languages — Part 2: Alpha-3 code» (Коды для представления названий языков. Часть 2. Трехбуквенный код). Существует также дополнительный механизм спецсимволов, который в настоящем стандарте не описан.

Второй субтег не обязателен. Если он присутствует, то должен иметь длину от 2 до 8 символов. Если его длина равна двум, то он должен содержать двухбуквенный код страны, взятый из ИСО 3166-1 «Codes for the representation of names of countries and their subdivisions — Part 1: Country codes» (Коды для представления названий стран и единиц их административно-территориального деления. Часть 1. Коды стран). Если длина субтега от 3 до 8 символов, то он должен содержать код, взятый из регистра тегов языков организации IANA. Дополнительные субтеги используются, если надо уточнить информацию о языке;

- MimeTypeCode — тип среды MIME, взятый из документа организации IETF RFC 2045 — Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (Многоцелевые расширения почты Интернет (MIME). Часть первая: формат тел сообщений Интернет). Комитет HL7 ведет подмножество MediaType типов среды MIME, имеющее ОИД 2.16.840.1.113883.5.79;

- ConceptStatusCode — статус понятия в системе кодирования (активное, устарело и т. д.);

- PropertyCode — свойство, которое может быть ассоциировано с понятием в системе кодирования;

- RelationshipCode — идентифицирует конкретное отношение, имеющее место в системе кодирования. Коды отношений должны по возможности браться из системы кодирования HL7 ConceptRelationship (ОИД 2.16.840.1.113883.5.1088);

- MapQualityCode — общее «качество» отображения понятия (точное, шире, уже и т. д.);

- RelationQualifierCode — квалификатор отношения понятий;

- MatchAlgorithmCode — кодированное обозначение алгоритма, используемого для определения совпадения строк и сравнения.

В таблице 22 перечислены системы кодирования и их идентификаторы ОИД, используемые в сообщениях API словаря ОТС.

Таблица 22 — ОИД и имена кодируемых элементов данных

| Элемент данных API словаря | ОИД системы кодирования | Имя системы кодирования |
|---|---|---|
| LanguageCode | 2.16.840.1.113883.6.99 | ISO639-1 |
| LanguageCode | 2.16.840.1.113883.6.100 | ISO639-2 |
| MimeTypeCode | 2.16.840.1.113883.5.79 | MediaType |
| ConceptStatusCode | 2.16.840.1.113883.5.1086 | ConceptStatusCode |
| PropertyCode | 2.16.840.1.113883.5.1087 | ConceptProperty |
| RelationshipCode | 2.16.840.1.113883.5.1088 | ConceptCodeRelationship |
| MapQualityCode | 2.16.840.1.113883.5.1093 | TranslationQuality |
| RelationQualifierCode | — | — |
| MatchAlgorithmCode | 2.16.840.1.113883.5.1094 | MatchAlgorithm |

**10.2.2 Раздел идентификации службы**

API времени исполнения на уровне словаря и API обозревателя наследуют общий интерфейс идентификации.

Идентификация службы содержит сведения о ее реализации — имя, версию, описание, реализованную версию ОТС. При каждом вызове методов идентификации службы возможна генерация исключения UnexpectedError.

Общий интерфейс идентификации предоставляет следующие методы:

- getServiceName — возвращает имя, присвоенное службе ее поставщиком;
- getServiceVersion — возвращает идентификатор версии, специфичный для конкретной реализации службы;
- getServiceDescription — описание службы, сведения об авторе, назначении и т. д.;
- getCTSVersion — конкретная версия ОТС, реализованная службой (например, 1.0).

**10.2.3 Исключения**

Ниже перечислены исключения, которые могут генерироваться одним или несколькими методами, описанными в настоящем подразделе. Исключения вызываются аномальными событиями, препятствующими нормальному завершению выполнения метода. Те исключения, что связаны с ошибками коммуникации, с операционными системами, базами данных и т. д., не включены в этот список. Предполагается, что механизмы обработки такого типа ошибок уже предусмотрены в языке программирования и/или в подсистеме коммуникации, использованной при реализации службы.

В приведенном ниже перечислении исключений предполагается, что базовая информация об исключении содержит текстовое поле, описывающее специфичные детали исключения. Дополнительные атрибуты, указанные в этом списке, предоставляют информацию, дополняющую базовый текст.

Исключения, генерируемые API словаря:

- идентификатор системы кодирования не распознан службой;
- идентификатор системы кодирования распознан службой, но код понятия concept_code не определен в этой системе;
- значение кода свойства property_code не поддерживается системой кодирования;
- язык с кодом language_code не поддерживается системой кодирования;
- отношение с кодом relationship_code не поддерживается системой кодирования;
- значение квалификатора отношения с кодом relationQualifier_code не поддерживается системой кодирования;
- тип среды MIME с кодом mimeType_code не поддерживается системой кодирования;
- у понятия с кодом concept_id нет обозначения на языке с кодом language_code;
- значение параметра codeSystem_name не идентифицирует ту же систему кодирования, что и значение параметра codeSystem_id, либо вообще не именует систему кодирования;
- служба не может осуществить разбор текста, переданного в параметре matchText.

**10.3 API времени исполнения на уровне словаря**

В настоящем подразделе описаны атрибуты и методы раздела времени исполнения API словаря ОТС.

**10.3.1 Системы кодирования, поддерживаемые API**

10.3.1.1 Структура класса CodeSystem

Класс CodeSystem имеет следующие поля:

- codeSystem_id — уникальный идентификатор системы кодирования, которым обычно служит объектный идентификатор ИСО (ОИД);
- codeSystem_name — офицальное имя системы кодирования;
- copyright — информация об авторских правах на систему кодирования, если таковые имеются;
- codeSystem_versions — версия или версии системы кодирования, поддерживаемые службой. Этот список может быть пуст, если система кодирования не имеет отдельных идентификаторов версий.

10.3.1.2 Метод getSupportedCodeSystems

Метод getSupportedCodeSystems возвращает список всех систем кодирования и их версий, поддерживаемых службой.

Входные параметры:

- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- TimeoutError;
- UnexpectedError.

**10.3.2 Обозревание идентифицирующей информации системы кодирования**

10.3.2.1 Структура, возвращаемая методом lookupCodeSystemInfo

Структура, возвращаемая методом lookupCodeSystemInfo, имеет следующие поля:

- codeSystem — идентификатор системы кодирования, ее имя и поддерживаемая версия (версии);
- fullName — полное имя системы кодирования;
- codeSystemDescription — описание содержания системы кодирования;
- supportedLanguages — список всех языков, которые полностью или частично поддерживаются системой кодирования. «Поддерживаемый» язык распознается системой кодирования и на нем доступна хотя бы часть обозначений или свойств понятий. Все системы кодирования должны поддерживать хотя бы один язык. В то время как служба должна возвращать перечисление всех субтегов основного языка, который она поддерживает, она не обязана перечислять вторичные языки. (Например, если она поддерживает язык «en-UK», она должна возвращать тег «en», но может возвращать или не возвращать «en-UK»);
- supportedRelations — список всех отношений, поддерживаемых службой в системе кодирования. «Поддерживаемое» отношение распознается системой кодирования, которая способна определить, являются ли два принадлежащих ей кода понятий связанными этим отношением. Неиерархическая система кодирования не нуждается в поддержке каких-либо отношений. Отношение подтипа представлено кодом отношения hasSubtype;
- supportedProperties — список всех свойств, поддерживаемых системой кодирования. Свойство считается «поддерживаемым», если система кодирования распознает его и по меньшей мере одно кодированное понятие ассоциировано с этим свойством и необязательным значением;
- supportedMimeTypes — список типов среды MIME, поддерживаемых системой кодирования. Все системы кодирования должны поддерживать тип среды text/plain (неформатированный текст), даже если в ней нет ни одного свойства с этим типом;
- supportedRelationshipQualifiers — список квалификаторов отношений, распознаваемых системой кодирования (если таковые имеются).

10.3.2.2 Метод lookupCodeSystemInfo

Метод lookupCodeSystemInfo получает идентификатор системы кодирования (ОИД) и/или ее имя и возвращает детальное описание системы кодирования и элементов, поддерживаемых службой.

Входные параметры:

- codeSystem_Id — идентификатор ИСО ОИД, присвоеный системе кодирования;
- codeSystem_name — уникальное имя системы кодирования.

Исключения:
- UnknownCodeSystem;
- CodeSystemNameIdMismatch;
- UnexpectedError.

**10.3.3 Проверка идентификатора понятия**

Метод isConceptIdValid определяет, является ли переданный ему идентификатор понятия действительным.

Входные параметры:
- concept_id — проверяемый код понятия и его система кодирования;
- activeConceptsOnly — значение TRUE указывает, что действительными считаются только активные понятия. Значение FALSE указывает, что действительным считается любой код понятия, известный системе кодирования.

Исключения:
- UnknownCodeSystem;
- UnexpectedError.

**10.3.4 Обозревание обозначения идентификатора понятия**

10.3.4.1 Класс StringAndLanguage

Класс StringAndLanguage содержит поле текста и поле кода языка, ассоциированного с текстом. Такая структура, состоящая из двух частей, существует по той причине, что код языка, возвращаемый методом lookupDesignation и другими операциями, может отличаться от кода языка, указанного во входных параметрах. Например, клиент мог запросить обозначение на языке с кодом «en-scouse» (ливерпульский диалект английского языка), а служба может вернуть обозначение на английском языке без указания диалекта (с кодом «en»). Программному обеспечению клиента может потребоваться информация о том, что возвращенное значение не точно совпадает с запрошенным.

В некоторых языках программирования связь с языком может являться частью возвращаемого значения (например, при использовании протокола XML/SOAP). Для единообразия поле кода языка language_code должно всегда передаваться в таких значениях, даже если оно избыточно.

10.3.4.2 Метод lookupDesignation

Метод lookupDesignation возвращает обозначение полученного им идентификатора понятия, наиболее подходящее для заданного языка и контекста. Метод lookupDesignation сперва пытается найти обозначение, которое в точности соответствует заданному языку. Если такое обозначение не найдено, то метод удаляет из кода языка крайний правый субтег, если таковой имеется, и повторяет процедуру поиска. Этот процесс повторяется до тех пор, пока будет найдено требуемое обозначение либо останется только основной тег языка. Например, для кода языка «en-UK-south» будут сделаны попытки поиска для языков «en-UK-south», «en-UK» и «en» в указанном порядке. Обозначения на языках с кодами «en-US» и «fr» будут игнорироваться при поиске.

Входные параметры:
- concept_id — код понятия и его система кодирования;
- language_code — желательный код языка для возвращаемого обозначения.

Исключения:
- UnknownLanguageCode;
- UnknownCodeSystem;
- UnknownConceptCode;
- NoApplicableDesignationFound;
- UnexpectedError.

10.3.4.3 Правила совпадения в методе lookupDesignation

В документе RFC 3066 «Tag for Identification of Languages» (теги идентификации языка) описаны коды языков, состоящие из нескольких частей. Первая часть берется из списков двух- или трехсимвольных кодов, определенных в ИСО 639, при этом двухсимвольный код имеет приоритет по сравнению с трехсимвольных, если у языка есть оба кода. Вторая часть кода языка, субтег, идентифицирует страну, регион или другой вариант.

Метод lookupDesignation использует следующие правила совпадения с заданным кодом языка:

1) Определить, поддерживает ли система кодирования основной код языка. Если нет, сгенерировать исключение UnknownLanguageCode;

2) Если обнаружено обозначение, у которого код языка точно совпадает с заданным, а признак предпочтительности preferredForLanguage имеет значение TRUE, возвратить его;

3) Если существуют обозначения, язык которых точно совпадает с заданным, а признак предпочтительности preferredForLanguage имеет значение FALSE, возвратить обозначение, самое раннее в алфавитном порядке;

4) Если код языка имеет один или более вторичных субтегов, удалить крайний правый тег, и повторить шаги 1 и 2;

5) Если у кода языка вторичный субтег отсутствует, сгенерировать исключение NoApplicable DesignationFound.

**10.3.5 Определение наличия отношения между двумя кодами понятий**

Метод areCodesRelated определяет наличие отношения между двумя переданными ему кодами понятий.

Входные параметры:

- codeSystem_id — система кодирования родительского и дочернего кодов;
- source_code — код понятия, предполагаемый источником отношения;
- target_code — код понятия, предполагаемый целью отношения;
- relationship_code — код понятия, идентифицирующий отношение;
- relationQualifiers — необязательный список кодов квалификаторов отношения. Если список relationQualifiers содержит один или несколько кодов квалификаторов, то рассматриваются только сведения о тех отношениях, квалификаторы которых совпадают со всеми квалификаторами, включенными в этот список;
- directRelationsOnly — значение TRUE указывает, что должны проверяться только непосредственные отношения. Значение FALSE указывает, что должно проверяться транзитивное замыкание отношения, если оно является транзитивным. Если отношение не транзитивно, то значение признака directRelationsOnly никакого влияния на результат не оказывает.

Метод areCodesRelated возвращает значение TRUE, если выполняется одно из следующих условий:

1) В заданной системе кодирования существует прямое отношение типа relationship_code между кодами понятий source_code и target_code;

2) В заданной системе кодирования существует прямое отношение типа relationship_code между кодами понятий target_code и source_code, при этом отношение является симметричным;

3) Коды понятий source_code и target_code эквивалентны и отношение рефлексивно;

4) Параметр directRelationsOnly имеет значение FALSE, отношение relationship_code транзитивно и в прямом транзитивном замыкании отношения relationship_code, начинающегося с кода понятия source_code, существует отношение типа relationship_code между кодами понятий source_code и target_code;

5) Параметр directRelationsOnly имеет значение FALSE, отношение relationship_code транзитивно и симметрично, и в обратном транзитивном замыкании отношения relationship_code, начинающегося с кода понятия source_code, существует отношение типа relationship_code между кодами понятий target_code и source_code.

Исключения:

- UnknownConceptCode;
- UnknownCodeSystem;
- UnknownRelationshipCode;
- UnknownRelationQualifier;
- UnexpectedError.

### 10.4 API обозревателя на уровне словаря

Модуль обозревателя на уровне словаря может быть реализован отдельно или в сочетании с модулем сообщений на уровне словаря. Функции обозревателя предназначены для обеспечения дополнительных возможностей, которые наиболее пригодны для среды разработки и просмотра словаря, и их производительность не является абсолютно критичным требованием.

API обозревателя на уровне словаря использует те же самые базовые типы данных, что и API времени исполнения. Они определены в подразделе 10.2 «Базовые типы данных». В настоящем подразделе описаны атрибуты и методы раздела обозревателя API на уровне словаря.

Обозреватель словаря наследует идентифицирующую информацию от интерфейса, описанного в 10.2.2 «Раздел идентификации службы».

**10.4.1 Системы кодирования, поддерживаемые API**

Метод getSupportedCodeSystems предоставляет список всех систем кодирования и их версий, поддерживаемых службой, в форме класса CodeSystemIdAndVersionsList, описанной в 10.3.1 «Системы кодирования, поддерживаемые API».

Входные параметры:

- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- TimeoutError;
- UnexpectedError.

**10.4.2 Поиск кодов понятий по тексту обозначения**

Метод lookupConceptCodesByDesignation возвращает список идентификаторов понятий, обозначения которых совпадают с заданной строкой текста и удовлетворяют заданным критериям.

Входные параметры:

- codeSystem_id — идентификатор системы кодирования, в которой осуществляется поиск;
- matchText — если присутствует и не пуст, то возвращаются сведения только о тех кодах понятий, обозначения которых совпадают с текстом, переданным в этом параметре. Если параметр matchText отсутствует или пуст, то возвращаются сведения обо всех обозначениях;
- matchAlgorithm_code — если параметр matchText присутствует и не пуст, то значение параметра matchAlgorithm_code указывает, каким образом определяется совпадение обозначения со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»;
- language_code — если этот параметр указан, то ограничивает поиск обозначений только на этом языке. (По умолчанию — поиск на всех языках.) Совпадающий код языка должен быть не более общим, чем значение, заданное этим параметром. Например, если этот параметр имеет значение «en», то должны считаться совпадающими обозначения, имеющие коды языка «en», «en-UK», «en-UK-south» и т. д. Если же параметр language_code имеет значение «en-UK-south», то будут возвращены идентификаторы только тех понятий, у которых язык обозначений совпадает с заданным;
- activeConceptsOnly — значение TRUE (используемое по умолчанию) указывает, что допустимыми считаются только коды понятий, которые в настоящее время активны. Значение FALSE указывает, что допустимыми считаются все коды понятий, содержащиеся в системе кодирования;
- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- UnknownCodeSystem;
- BadlyFormedMatchText;
- UnknownMatchCode;
- UnknownLanguageCode;
- TimeoutError;
- UnexpectedError.

**10.4.3 Поиск кодов понятий по их свойствам**

Метод lookupConceptCodesByProperty возвращает список идентификаторов понятий, имеющих одно или несколько свойств, совпадающих с заданной строкой текста и удовлетворяющих заданным критериям.

Входные параметры:

- codeSystem_id — идентификатор системы кодирования, в которой осуществляется поиск;
- matchText — искомый текст. Формат и синтаксис этого текста зависит от значения параметра matchAlgorithmCode;

- matchAlgorithm_code — указывает алгоритм, с помощью которого определяется совпадение свойства со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»;

- language_code — если этот параметр указан, то ограничивает поиск свойств только на этом языке. (По умолчанию — поиск на всех языках.) Совпадающий код языка должен быть не более общим, чем значение, заданное этим параметром. Например, если этот параметр имеет значение «en», то должны считаться совпадающими обозначения, имеющие коды языка «en», «en-UK», «en-UK-south» и т. д. Если же параметр language_code имеет значение «en-UK-south», то будут возвращены идентификаторы только тех понятий, у которых язык свойств совпадает с заданным;

- activeConceptsOnly — значение TRUE (используемое по умолчанию) указывает, что допустимыми считаются только коды понятий, которые в настоящее время активны. Значение FALSE указывает, что допустимыми считаются все коды понятий, содержащиеся в системе кодирования;

- properties — список кодов свойств, участвующих в поиске (по умолчанию искать по всем свойствам);

- mimeTypes — список типов среды MIME, участвующих в поиске (по умолчанию искать по всем типам среды MIME);

- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;

- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Исключения:

- UnknownCodeSystem;
- BadlyFormedMatchText;
- UnknownMatchCode;
- UnknownLanguageCode;
- UnknownPropertyCode;
- UnknownMimeTypeCode,
- TimeoutError;
- UnexpectedError.

**10.4.4 Возвращение полного описания кодированного понятия**

Возвращаемые полные сведения о кодируемом понятии образованы из всех описанных ниже классов.

10.4.4.1 Класс ConceptDesignation

Класс ConceptDesignation имеет следующие поля:

- designation — обозначение кодированного понятия;
- language_code — язык обозначения;
- preferredForLanguage — значение TRUE указывает, что это обозначение следует предпочесть для представления назначения кодированного понятия CodedConcept на заданном языке, если иные критерии отсутствуют. Для каждого заданного языка и заданного понятия только одно обозначение может быть указано предпочтительным.

10.4.4.2 Класс ConceptProperty

Класс ConceptProperty имеет следующие поля:

- property_code — код понятия, идентифицирующий конкретное свойство;
- propertyValue — значение этого свойства у данного понятия;
- language_code — язык значения propertyValue (необязательный);
- mimeTypeCode — тип среды MIME значения propertyValue (по умолчанию text/plain).

10.4.4.3 Класс CompleteCodedConceptDescription

10.4.4.3.1 Класс ConceptRelationship

Класс ConceptRelationship имеет следующие поля:

- sourceConcept_id — идентификатор понятия (идентификатор системы кодирования и код понятия), являющегося источником отношения;
- relationship_code — код отношения;
- relationQualifiers — необязательный список кодов квалификаторов, уточняющих или иным образом характеризующих отношение;

- targetConcept_id — идентификатор понятия (идентификатор системы кодирования и код понятия), являющегося источником отношения.

10.4.4.3.2 Класс CompleteCodedConceptDescription

Класс CompleteCodedConceptDescription имеет следующие поля:

- concept_id — идентификатор системы кодирования и код понятия, информация о котором должна быть извлечена;
- conceptStatus_code — статус понятия в заданной системе кодирования;
- codeSystem_version — версия системы кодирования, из которой берется описание понятия;
- designatedBy — список всех обозначений кодированного понятия;
- hasProperties — список всех свойств кодированного понятия за исключением свойств обозначения ConceptDesignation;
- sourceFor — список всех отношений, в которых данное понятие выступает в роли «источника»;
- targetOf — список всех отношений, в которых данное понятие выступает в роли «цели».

10.4.4.3.3 Метод lookupCompleteCodedConcept

Метод lookupCompleteCodedConcept возвращает структурированную информацию, содержащую все известные сведения о заданном коде понятия (с точки зрения CTS).

Входные параметры:

- concept_id — идентификатор системы кодирования и код понятия.

Исключения:

- UnknownCodeSystem;
- UnknownConceptCode;
- UnexpectedError.

**10.4.5 Просмотр обозначений конкретного кода понятия**

Метод lookupDesignations возвращает избранные обозначения заданного понятия. Правила совпадения, используемые методом lookupDesignations, идентичны тем, что определены в 10.4.2 «Поиск кодов понятий по тексту обозначения». Возвращаемые значения имеют тип данных списка ConceptDesignationList, элементы которого описаны в 10.4.4.1 «Класс ConceptDesignation».

Входные параметры:

- codeSystem_id — идентификатор системы кодирования, в которой осуществляется поиск;
- matchText — искомый текст. Формат и синтаксис этого текста зависит от значения параметра matchAlgorithmCode;
- matchAlgorithm_code — указывает алгоритм, с помощью которого определяется совпадение обозначения со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»;
- language_code — если этот параметр указан, то ограничивает поиск свойств только на этом языке. (По умолчанию — поиск на всех языках.)

Исключения:

- UnknownCodeSystem;
- UnknownConceptCode;
- BadlyFormedMatchText;
- UnknownMatchAlgorithm;
- UnknownLanguageCode;
- UnexpectedError.

**10.4.6 Просмотр свойств кода понятия**

Метод lookupProperties возвращает избранные свойства заданного понятия. Обработка параметров properties, matchText language_code и mimeTypes следует тем же правилам, что используются методом lookupConceptCodesByProperty. Возвращаемые значения имеют тип данных списка ConceptPropertyList, элементы которого описаны в 10.4.4.2 «Класс ConceptProperty».

Входные параметры:

- codeSystem_id — идентификатор системы кодирования, в которой осуществляется поиск;
- properties — список кодов свойств, участвующих в поиске (по умолчанию искать по всем свойствам);
- matchText — искомый текст. Формат и синтаксис этого текста зависит от значения параметра matchAlgorithmCode,
- matchAlgorithm_code — указывает алгоритм, с помощью которого определяется совпадение свойства со значением параметра matchText. Детальные сведения см. в 8.2.2.1 «Алгоритмы совпадения строк»;

- language_code — если этот параметр указан, то ограничивает поиск свойств только на этом языке. (По умолчанию — поиск на всех языках.);
- mimeTypes — список типов среды MIME, участвующих в поиске (по умолчанию искать по всем типам среды MIME).

Исключения:

- UnknownCodeSystem;
- UnknownConceptCode;
- BadlyFormedMatchText;
- UnknownMatchAlgorithm;
- UnknownLanguageCode;
- UnknownPropertyCode;
- UnknownMimeTypeCode,
- UnexpectedError.

**10.4.7 Возвращение списка связанных кодов понятий**

10.4.7.1 Структура данных, возвращаемая методом lookupCodeExpansion

Структура данных, возвращаемая методом lookupCodeExpansion, имеет следующие поля:

- pathLength — целое значение, определяющее расстояние в уровнях от раскрываемого кода, для которого это поле всегда имеет значение 0;
- concept_code — связанный код понятия;
- designation — предпочтительное обозначение кода на подходящем языке в прикладном контексте либо обозначение по умолчанию, если ни одно из обозначений явно не объявлено предпочтительным;
- relationQualifiers — список кодов квалификаторов отношения, применяемых к данному узлу;
- canExpand — значение TRUE указывает, что существуют дополнительные коды понятий, непосредственно связанные с кодом понятия concept_code, которые могут быть далее раскрыты;
- expansionContext — если параметр canExpand имеет значение TRUE, то параметр expansionContext содержит непрозрачный контекст, который может использоваться для дальнейшего раскрытия кода в этом узле.

10.4.7.2 Просмотр иерархического раскрытия кода

Метод lookupCodeExpansion возвращает линеаризованный список кодов, имеющих отношение типа relationship_code с понятием, имеющим код expandConcept_id.concept_ code в системе кодирования с идентификатором expandConcept_id.codeSystem_id.

Входные параметры:

- expandConcept_id — идентификатор системы кодирования и раскрываемый код понятия. Если параметр relationship_code имеет значение «hasSubtype» и признак souceToTarget имеет значение TRUE, то код понятия может быть опущен. В этом случае должны возвращаться сведения обо всех «корневых» понятиях отношения подтипа, то есть всех понятиях, которые не являются целью одного или нескольких отношений типа «hasSubtype»;
- relationship_code — код раскрываемого понятия;
- sourceToTarget — значение TRUE указывает, что раскрытие осуществляется от источника к цели. Значение FALSE указывает, что раскрытие осуществляется от цели к источнику;
- directRelationsOnly — если этот параметр имеет значение TRUE или отношение, заданное параметром relationship_code, не является транзитивным, то возвращаются только прямые цели (или источники, если параметр sourceToTarget имеет значение FALSE) раскрываемого понятия с идентификатором expandConcept_id. Если он имеет значение FALSE и отношение, заданное параметром relationship_code, является транзитивным, то возвращаются также потомки или предшественники этого понятия;
- designationLanguage_code — код языка, который должен был использоваться для возвращаемых обозначений понятий;
- timeout — время в миллисекундах, в течение которого клиент готов ждать завершения операции. Значение 0 параметра timeout указывает, что на время ее выполнения ограничения не накладываются;
- sizeLimit — максимальное число элементов, которое служба может вернуть. Если число возвращенных элементов совпадает с sizeLimit, то клиент предполагает, что существуют дополнительные элементы, которые не были возвращены. Значение 0 параметра sizeLimit указывает, что число элементов, которые могут быть возвращены, не ограничивается.

Для возвращения потомков или предшественников узла используется обход дерева в глубину. Если узел встречается более чем в одной ветви, то в возвращаемой структуре он реплицируется. Если параметр directRelationsOnly имеет значение FALSE, отношение транзитивно и в структуре отношения встречается цикл, то раскрытие будет останавливаться на последнем неповторяющемся узле. В этом случае параметр canExpand будет иметь значение TRUE, что позволит клиенту при необходимости пошагово пройти по этой циклической структуре.

Параметры designationLanguage_code и usageContext_code используются при определении того, какое именно обозначение должно быть возвращено в структуре с типом данных RelatedCode. Для определения подходящего обозначения используются те же правила, что и в методе lookupDesignation, только данный метод не генерирует исключение NoApplicableDesignationFound. Если у конкретного узла нет подходящих обозначений, то поле обозначения должно содержать пустую строку.

Исключения:

- UnknownCodeSystem;
- UnknownConceptCode;
- UnknownRelationshipCode;
- UnknownLanguageCode;
- TimeoutError;
- UnexpectedError.

10.4.7.2.1 Детальные сведения о методе lookupCodeExpansion

На диаграмме, приведенной на рисунке 7, вершины представляют произвольное транзитивное отношение, а стрелки исходят из понятий-источников и указывают на целевые понятия.

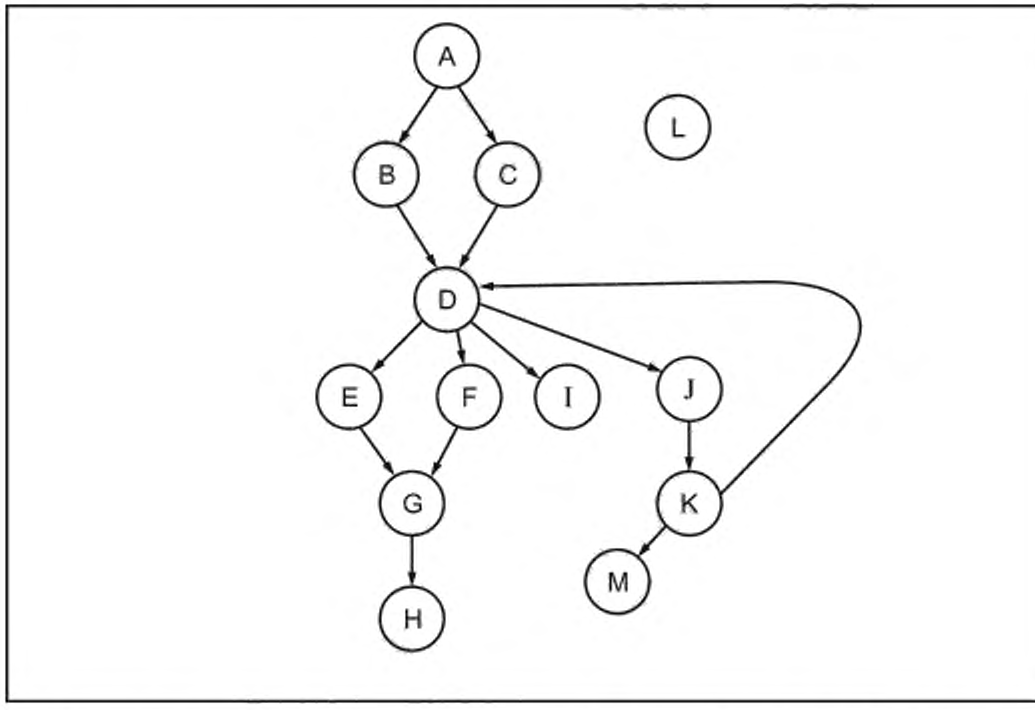


Рисунок 7 — Пример графа отношений для демонстрации применения метода lookupCodeExpansion

В таблицах 23—29 показаны сведения, возвращаемые методом lookupCodeExpansion при использовании информации, показанной на рисунке 7, и параметрах вызова, указанных в заголовках этих таблиц.

Таблица 23 — Результат вызова метода lookupCodeExpansion с параметрами lookupCodeExpansion (expandConcept_id=D, sourceToTarget=TRUE, directRelationsOnly=TRUE)

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | E | TRUE | (ecE) |
| 1 | F | TRUE | (ecF) |
| 1 | I | FALSE | — |
| 1 | J | TRUE | (ecJ) |

Таблица 24 — Результат вызова метода lookupCodeExpaionsion (expandConcept_id=D, sourceToTarget=TRUE, directRelationsOnly=FALSE)

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | E | FALSE | — |
| 2 | G | FALSE | — |
| 3 | H | FALSE | — |
| 1 | F | FALSE | — |
| 2 | G | FALSE | — |
| 3 | H | FALSE | — |
| 1 | I | FALSE | — |
| 1 | J | FALSE | — |
| 2 | K | TRUE | (ecK) |
| 3 | M | FALSE | — |

Примечание — В таблице 24 узел 2 K возвращен как раскрываемый, поскольку был обнаружен цикл.

Таблица 25 — Результат вызова метода lookupCodeExpansion (expandConcept_id=D, sourceToTarget=FALSE, directRelationsOnly=FALSE)

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | B | FALSE | — |
| 2 | A | FALSE | — |
| 1 | C | FALSE | — |
| 2 | K | FALSE | — |
| 1 | J | FALSE | — |
| 2 | D | TRUE | (ecD) |

Таблица 26 — Результат вызова метода lookupCodeExpansion (expandConcept_id=, sourceToTarget=TRUE, directRelationsOnly=TRUE)

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | A | TRUE | (ecA) |
| 1 | L | FALSE | — |

Таблица 27 — Результат вызова метода lookupCodeExpansion (expandConcept_id=, sourceToTarget=TRUE, directRelationsOnly=FALSE)

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | L | FALSE | — |
| 1 | A | FALSE | — |
| 2 | B | FALSE | — |
| 3 | D | FALSE | — |
| 4 | E | FALSE | — |
| 5 | G | FALSE | — |
| 6 | H | FALSE | — |
| 4 | F | FALSE | — |
| 5 | G | FALSE | — |
| 6 | H | FALSE | — |
| 4 | I | FALSE | — |
| 4 | J | FALSE | — |
| 5 | K | TRUE | (ecK) |
| 6 | M | FALSE | — |
| 2 | C | FALSE | — |
| 3 | D | FALSE | — |
| 4 | E | FALSE | — |
| 5 | G | FALSE | — |
| 6 | H | FALSE | — |
| 4 | F | FALSE | — |
| 5 | G | FALSE | — |
| 6 | H | FALSE | — |
| 4 | I | FALSE | — |
| 4 | J | FALSE | — |
| 5 | K | TRUE | (ecK) |
| 6 | M | FALSE | — |
| 2 | C | FALSE | — |

Таблица 28 — Результат вызова метода lookupCodeExpansion (expandConcept_id=, sourceToTarget=FALSE, directRelationsOnly=FALSE)

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | H | TRUE | (ecH) |
| 1 | I | TRUE | (ecI) |
| 1 | M | TRUE | (ecM) |
| 1 | L | FALSE | — |

Таблица 29 — Результат вызова метода lookupCodeExpansion (expandConcept_id=L, sourceToTarget=TRUE, directRelationsOnly=FALSE)

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| — | — | — | — |

**10.4.8 Дальнейшее раскрытие возвращенного узла**

Метод expandCodeExpansionContext обеспечивает дальнейшее раскрытие элемента списка RelatedCodeList, возвращенного предшествующим вызовом метода lookupCodeExpansion или expandCodeExpansionContext.

Входные параметры:

- contextToExpand — непрозрачный идентификатор, возвращенный в поле expansionContext элемента списка RelatedCodeList, возвращенного предшествующим вызовом метода lookupCodeExpansion или expandCodeExpansionContext.

Исключения:

- InvalidExpansionContext.

Примечание — Контексты раскрытия не обязательно постоянны. Обеспечение действительности контекста раскрытия в определенном интервале времени является функцией конкретной реализации службы;

- NoApplicableDesignationFound;
- TimeoutError;
- UnexpectedError.

10.4.8.1 Детальные сведения о методе expandCodeExpansionContext

В таблицах 30—32 показаны сведения, возвращаемые методом expandCodeExpansionContext при использовании информации, описанной в 10.4.7 «Возвращение списка связанных кодов понятий».

Таблица 30 — Результат вызова метода expandCodeExpansionContext (expandContext=ecE) для раскрытия контекста, возвращенного в примере, показанном в таблице 23

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | G | TRUE | (ecG) |

Примечание — Длина пути продолжает ту, что была получена при предшествующем вызове.

Таблица 31 — Результат вызова метода expandCodeExpansionContext (expandContext=ecG) для раскрытия контекста, возвращенного в примере, показанном в таблице 30

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 1 | H | FALSE | — |

Таблица 32 — Результат вызова метода expandCodeExpansionContext (expandContext=ecG) для раскрытия контекста, возвращенного в примере, показанном в таблице 31

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 3 | D | FALSE | — |
| 4 | E | FALSE | — |
| 5 | G | FALSE | — |
| 6 | H | FALSE | — |
| 4 | F | FALSE | — |
| 5 | G | FALSE | — |
| 6 | H | FALSE | — |

*Окончание таблицы 32*

| pathLength | concept_code | canExpand | expansionContext |
|---|---|---|---|
| 4 | I | FALSE | — |
| 4 | J | FALSE | — |
| 5 | K | TRUE | (ecK) |
| 6 | M | FALSE | — |

## 11 Модель отображения кодов

### 11.1 Введение

Программному обеспечению, обрабатывающему сообщения HL7, может потребоваться отображение местных понятий на стандартизованные коды, используемые в среде сообщений, соответствующих стандарту HL7, или отображение из одного стандартизованного набора данных в другой. Для выполнения таких преобразований API отображения кодов предоставляет необходимый интерфейс. Модель отображения, представленная в настоящем стандарте (рисунок 8), весьма ограничена и не предусматривает отображение одного кода-источника на несколько целевых кодов. Предполагается, что в следующих версиях настоящего документа функциональность отображения будет расширена.

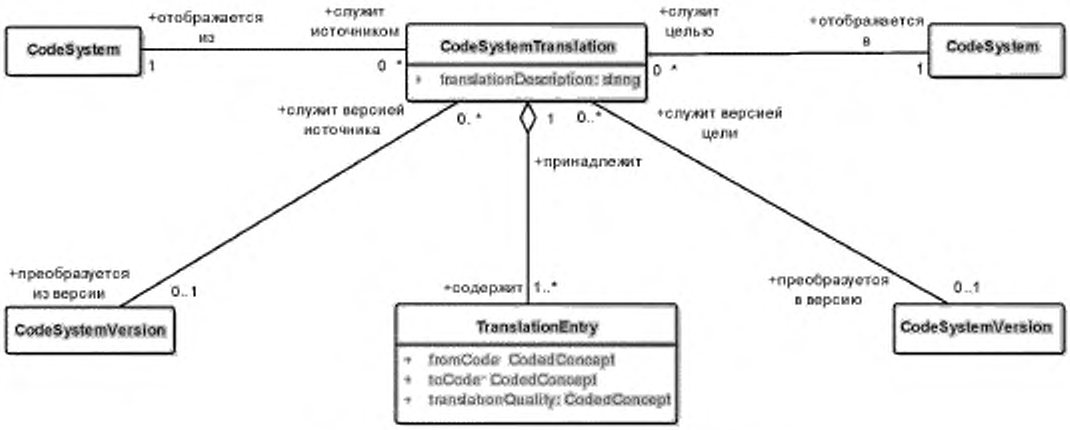


Рисунок 8 — Модель отображения кодов в ОТС

### 11.2 Отображение кодов

#### 11.2.1 Класс CodeMap

Экземпляры класса **CodeMap** описывают отображение некоторых или всех кодов понятий из системы кодирования-источника в наиболее близкие коды понятий в целевой системе кодирования. Отображение **CodeMap** является однонаправленным — из наличия отображения системы кодирования-источника не вытекает возможность обратного отображения. В экземпляре класса **CodeMap** может быть указана необязательная идентификация версий системы кодирования-источника и целевой системы кодирования, используемых при отображении.

Класс **CodeMap** «содержит» один или несколько экземпляров класса **MapEntry**. В настоящем стандарте представление этого отображения является чисто символическим и не предназначено для описания способов реального отображения кодов.

Каждое отображение **CodeMap** идентифицируется уникальным именем отображения *map_name*. Между одной и той же парой систем кодирования может быть более одного отображения **CodeMap**.

Каждое отображение **CodeMap** имеет необязательное поле *mapDescription*, описывающее источник трансляции, момент ее создания, способ ее выполнения и т. д.

### 11.3 Класс MapEntry

Класс TranslationEntry представляет трансляцию кода понятия fromCode из системы кодирования-источника в соответствующий код понятия toCode, принадлежащий целевой системе кодирования. Поле mapQuality_code задает «качество» трансляции исходного кода в целевой и может принимать одно из значений «точная», «уже», «шире» или «частичное перекрытие».

## 12 Спецификация отображения кодов

### 12.1 Введение

Интерфейс отображения кодов поддерживает отображение кодов понятий, принадлежащих одной или нескольким системам кодирования-источников, в эквивалентные им коды, принадлежащие целевой системе кодирования.

### 12.2 Идентификация службы отображения

Служба отображения наследует общий интерфейс идентификации.

### 12.3 Отображение кода

#### 12.3.1 Класс CodeMap

Класс CodeMap описывает идентификацию конкретного отображения. Он имеет следующие поля:

- map_name — уникальное имя конкретного отображения;
- fromCodeSystem_id — идентификатор ИСО ОИД системы кодирования-источника;
- fromCodeSystem_name — имя системы кодирования-источника;
- fromCodeSystem_version — версия системы кодирования-источника, используемая при отображении (необязательная);
- toCodeSystem_id — идентификатор ИСО ОИД целевой системы кодирования;
- toCodeSystem_name — имя целевой системы кодирования;
- toCodeSystem_version — версия целевой системы кодирования, используемая при отображении (необязательная);
- description — описание отображения (источник, версия, дата, местонахождение и т. д.).

#### 12.3.2 Метод getSupportedMaps

Метод getSupportedMaps возвращает список отображений кодов, поддерживаемых данной конкретной службой. Отображения кодов не симметричны. Из того факта, что служба поддерживает отображение из системы кодирования А в систему кодирования В, не следует, что она также поддерживает отображение в обратном направлении.

### 12.4 Исключения

Ниже перечислены исключения, которые могут генерироваться одним или несколькими методами, описанными в настоящем разделе. Исключения вызываются аномальными событиями, препятствующими нормальному завершению выполнения метода. Те исключения, что связаны с ошибками коммуникации, с операционными системами, базами данных и т. д., не включены в этот список. Предполагается, что механизмы обработки такого типа ошибок уже предусмотрены в языке программирования и/или в подсистеме коммуникации, использованной при реализации службы.

В приведенном ниже перечислении исключений предполагается, что базовая информация об исключении содержит текстовое поле, описывающие специфичные детали исключения. Дополнительные атрибуты, указанные в этом списке, предоставляют информацию, дополняющую базовый текст.

Исключения, генерируемые при отображении кодов:

- служба не поддерживает трансляцию системы кодирования с идентификатором fromCodeSystem_id в систему кодирования с идентификатором toCodeSystem_id;
- служба не в состоянии выполнить требуемое отображение;
- заданное имя отображения не распознано службой;

- между заданной системой кодирования-источника и целевой системой кодирования существует несколько возможных отображений. В параметре possible_maps передается список имен этих отображений;
- идентификатор системы кодирования, указанный в параметре fromConcept_id, не совпадает с идентификатором системы кодирования, указанным в отображении с именем, переданным в параметре map_name;
- идентификатор системы кодирования, указанный в параметре toCodeSystem_id, не совпадает с идентификатором системы кодирования, указанным в отображении с именем, переданным в параметре map_name.

### 12.5 Отображение кода понятия

#### 12.5.1 Структура выходных данных метода mapConceptCode

Выходные данные метода mapConceptCode имеют следующие поля.

- mappedConcept_id — идентификатор системы кодирования и код понятия;
- mapQuality_code — код, характеризующий «качество» операции отображения, например, «точное», «шире», «уже» и т. д.

#### 12.5.2 Метод mapConceptCode

Метод mapConceptCode отображает заданный код понятия, принадлежащий заданной системе кодирования, в соответствующий код целевой системы кодирования (если таковой имеется). Параметру fromConcept_id должно быть присвоено значение. Если указано только имя отображения map_name, то значение параметра toCodeSystem_id выводится из полей этого отображений. Если оба эти параметра имеют значения, то они должны быть согласованными.

Входные параметры:

- fromConcept_id — имя системы кодирования и код отображаемого понятия;
- toCodeSystem_id — идентификатор целевой системы кодирования;
- map_name — имя используемого отображения.

Исключения:

- UnknownCodeSystem;
- UnknownConceptCode;
- MappingNotAvailable;
- UnknownMapName;
- AmbiguousMapRequest;
- MapNameSourceMismatch;
- MapNameTargetMismatch;
- UnableToMap;
- UnexpectedError.

## 13 Привязка ОТС к языкам программирования

### 13.1 Преобразование спецификации на языке IDL в программный код на языке Java

Спецификация интерфейсов ОТС написана на языке определения интерфейсов IDL (interface definition language), описанном в ИСО/МЭК 14750:1999 — Open Distributed Processing — Interface Definition Language (IDL) (Информационные технологии. Открытая распределенная обработка. Определение интерфейсов. Язык). Организация Object Management Group (OMG) описала отображение языка IDL на многие широко распространенные языки программирования, включая ADA, C, C++, COBOL, Java, Lisp, PL/1, Python и Smalltalk. Существует также привязка языка IDL к объектной модели Microsoft Common Object Model (COM).

К сожалению, эти отображения на языки программирования частично зависят от архитектуры CORBA (Common Object Request Broker Architecture — общая архитектура брокера объектных запросов). Например, отображение на язык программирования Java имеет следующие особенности:

- классы исключений расширяют объект org.omg.CORBA.UserException и вызывают его как суперкласс с аргументом [class]Helper.id();
- классы Struct реализуют интерфейс org.omg.CORBA.portable.IDLEntity;
- классы интерфейсов именуются [class]Operations.java;
- создаются многие дополнительные вспомогательные файлы [class]Holder.java, [class]Helper.java, _[class]Stub.java, [class]POA.java, [class]POATie.java.

Однако не очень сложно удалить эти пережитки платформы CORBA и тем самым получить спецификацию интерфейса, нейтральную по отношению к реализации.

Для перехода от языка IDL к целевым языкам SOAP и Java используются следующие шаги:

1. Преобразовать IDL в Java:

java com.sun.tools.corba.se.idl.toJavaPortable.Compile -fallTIE -pkgPrefix types org.hl7 -pkgPrefix CTSMAPI org.hl7 -pkgPrefix CTSVAPI org.hl7 CTSVAPI.idl

2. Удалить базовые интерфейсные файлы — те, у которых есть соответствующие файлы «xxxOperations.java». Например, файл «Browser.java» удаляется, поскольку существует соответствующий ему файл «BrowserOperations.java».

3. Удалить все выходные файлы, имена которых заканчиваются на «Holder.java», «Helper.java», «Stub.java», «POA.java» and «POATie.java».

4. Заменить «extends org.omg.CORBA.UserException» на «extends java.lang.Exception» и удалить все вызовы с ключевым словом «super» из конструкторов класса исключений.

5. Изменить обозначения комментариев с «/*» на «/**», чтобы они включались в документацию, создаваемую с помощью утилиты javadoc.

Ссылки на класс org.omg.CORBA.portable.IDLEntity не удаляются, поскольку они указывают на пустой класс, который тем не менее может быть полезен для различения разных типов.

Ниже показаны примеры этих преобразований:

***Примеры***

***1 Объявление структуры на языке IDL.***

```
/* Идентификатор версии спецификации CTC */
        struct CTSVersionId {
                short   major;
                short   minor;
        };
```

***2 Объявление структуры на языке Java.***

```
package org.hl7.CTSVAPI;

/**
* org/hl7/CTSVAPI/CTSVersionId.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from idl/CTSVAPI.idl
* Monday, March 8, 2004 11:17:26 PM CST
*/


/**
 *$lt;PRE> Идентификатор версии спецификации CTC </PRE>
 */
public final class CTSVersionId implements org.omg.CORBA.portable.IDLEntity
{
  public short major = (short)0;
  public short minor = (short)0;

  public CTSVersionId ()
  {
  } // ctor

  public CTSVersionId (short _major, short _minor)
  {
    major = _major;
    minor = _minor;
  } // ctor

} // class CTSVersionId
```

***3 Объявление исключения на языке IDL.***

```
/*
 * Использован код свойства, не действительный для системы кодирования
 */
        exception UnknownPropertyCode {
                PropertyCode            property_code;
        };
```

***4 Объявление исключения на языке Java***

```
package org.hl7.CTSVAPI;

/**
* org/hl7/CTSVAPI/UnknownPropertyCode.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from idl/CTSVAPI.idl
* Monday, March 8, 2004 11:17:26 PM CST
*/

public final class UnknownPropertyCode extends java.lang.Exception
{
  public String property_code = null;

  public UnknownPropertyCode ()
  {
    // super(UnknownPropertyCodeHelper.id());
  } // ctor

  public UnknownPropertyCode (String _property_code)
  {
    // super(UnknownPropertyCodeHelper.id());
    property_code = _property_code;
  } // ctor


  public UnknownPropertyCode (String $reason, String _property_code)
  {
    // super(UnknownPropertyCodeHelper.id() + "  " + $reason);
    property_code = _property_code;
  } // ctor

} // class UnknownPropertyCode
```

***5 Объявление интерфейса на языке IDL.***

```
/***********************************************
 *      Интерфейс отображения кодов             *
 *                                              *
 * Интерфейс отображения кодов предоставляет одно*
 * или несколько отображений систем кодирования  *
 ***********************************************/
        interface CodeMapping : Identification {

/* Список отображений, поддерживаемых службой */
                CodeMapList getSupportedMaps() raises (UnexpectedError);

/* Отображает код понятия, включенный в систему кодирования-источник, на его
 * ближайший эквивалент из целевой системы кодирования, если таковой имеется
 *      fromConcept_id      - Система кодирования / код отображаемого понятия
 *      toCodeSystem_id     - Целевая система кодирования
```

```
*      map_name                  - Имя используемого отображения. Может быть
*                                опущено, если отображение из системы
*                                кодирования, являющейся компонентом fromConcept_id,
*                                в систему кодирования toCodeSystem_id является
*                                единственным
*
*      Возвращаемое значение     - Соответствующее понятие в целевой системе
*      кодирования
*
*      Исключения:
*    UnknownCodeSystem     - Система кодирования-источник или целевая
*                          система кодирования не поддерживается
*                          данной службой отображения
*    UnknownConceptCode    - Код отображаемого понятия не является частью
*                          системы кодирования
*    MappingNotAvailable   - Переданный код понятия не отображается на
*                          целевую систему кодирования
*
*    UnableToMap           - Переданный код понятия не может быть отображен
*    UnknownMapName        - Служба не распознала имя отображения
*    MapSourceMismatch     - Идентификатор системы кодирования-источника
*                          fromConcept_id не совпадает с тем, что указан в
*                          отображении
*    MapTargetMismatch     - Идентификатор целевой системы кодирования
*                          targetCodeSystem_id не совпадает с тем, что
*                          указан в отображении
*    AmbiguousMapRequest   - Существует несколько отображений понятия системы
*                          кодирования-источника в целевую систему кодирования
*    UnexpectedError       - Не специфичная ошибка, препятствующая успешному
*                          завершению вызова
*/
            MappedConceptCode mapConceptCode(
                    in ConceptId                fromConcept_id,
                    in CodeSystemId         toCodeSystem_id,
                    in string               map_name
                    )
                    raises ( UnknownCodeSystem,
                             UnknownConceptCode,
                             MappingNotAvailable,
                             UnknownMapName,
                             AmbiguousMapRequest,
                             MapNameSourceMismatch,
                             MapNameTargetMismatch,
                             UnableToMap,
                             UnexpectedError);
    };
```

***6 Объявление интерфейса на языке Java.***

```
package org.hl7.CTSVAPI;


/**
* org/hl7/CTSVAPI/CodeMappingOperations.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from idl/CTSVAPI.idl
* Monday, March 8, 2004 11:17:27 PM CST
*/
```

```
/**********************************************
 *      Интерфейс отображения кодов           *
 *                                            *
 * Интерфейс отображения кодов предоставляет одно*
 * или несколько отображений систем кодирования  *
 **********************************************/
public interface CodeMappingOperations  extends org.hl7.CTSVAPI.IdentificationOperations
{

  /**
 *<PRE> Список отображений, поддерживаемых службой </PRE>
 */
  org.hl7.CTSVAPI.CodeMap[] getSupportedMaps () throws org.hl7.CTSVAPI.UnexpectedError;

  /**
 *<PRE> Отображает код понятия, включенный в систему кодирования-источник,
 *        на его ближайший эквивалент из целевой системы кодирования,если
 *        таковой имеется
 *      fromConcept_id      - Система кодирования / код отображаемого понятия
 *      toCodeSystem_id     - Целевая система кодирования
 *      map_name            - Имя используемого отображения. Может быть
 *                          опущено, если отображение из системы
 *                          кодирования, являющейся компонентом fromConcept_id,
 *                          в систему кодирования toCodeSystem_id является
 *                          единственным
 *
 *      Возвращаемое значение     - Соответствующее понятие в целевой системе
 *      кодирования
 *
 *      Исключения:
 *    UnknownCodeSystem     - Система кодирования-источник или целевая
 *                          система кодирования не поддерживается
 *                          данной службой отображения
 *    UnknownConceptCode - Код отображаемого понятия не является частью
 *                          системы кодирования
 *    MappingNotAvailable   - Переданный код понятия не отображается на
 *                          целевую систему кодирования
 *
 *    UnableToMap           - Переданный код понятия не может быть отображен
 *    UnknownMapName        - Служба не распознала имя отображения
 *    MapSourceMismatch     - Идентификатор системы кодирования-источника
 *                          fromConcept_id не совпадает с тем, что указан в
 *                          отображении
 *    MapTargetMismatch     - Идентификатор целевой системы кодирования
 *                          targetCodeSystem_id не совпадает с тем, что
 *                          указан в отображении
 *    AmbiguousMapRequest   - Существует несколько отображений понятия системы
 *                          кодирования-источника в целевую систему кодирования
 *    UnexpectedError       - Не специфичная ошибка, препятствующая успешному
 *                          завершению вызова
    </PRE>
 */
  org.hl7.CTSVAPI.MappedConceptCode mapConceptCode (org.hl7.CTSVAPI.ConceptId
fromConcept_id, String toCodeSystem_id, String map_name)
     throws org.hl7.CTSVAPI.UnknownCodeSystem, org.hl7.CTSVAPI.UnknownConceptCode, org.
hl7.CTSVAPI.MappingNotAvailable,
            org.hl7.CTSVAPI.UnknownMapName, org.hl7.CTSVAPI.AmbiguousMapRequest, org.hl7.
CTSVAPI.MapNameSourceMismatch,
            org.hl7.CTSVAPI.MapNameTargetMismatch, org.hl7.CTSVAPI.UnableToMap, org.hl7.
CTSVAPI.UnexpectedError;
} // interface CodeMappingOperations
```

### 13.2 Преобразование спецификации на языке IDL в описание на языке WSDL

Для преобразования объявлений на языке Java в описание на языке WSDL используется компилятор Apache Axis 1.1 java2wsdl. Ниже приведен пример его вызова для преобразования CTSVAPI:

```
java org.apache.axis.wsdl.Java2WSDL -n urn://HL7.org/CTSVAPI -i
org.hl7.refImpl.CTSVAPI -lhttp://localhost:8080/axis/services/CTSVAPIService
org.hl7.CTSVAPI
```

***Пример — Объявления интерфейса на языке WSDL:***

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn://hl7.org/CTSVAPI"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/xml-
soap"
                  xmlns:impl="urn://hl7.org/CTSVAPI"
xmlns:intf="urn://hl7.org/CTSVAPI"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:types>
      <schema targetNamespace="urn://hl7.org/CTSVAPI"
xmlns="http://www.w3.org/2001/XMLSchema">
         <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
         <complexType name="SupportedMap">
            <sequence>
               <element name="map_name" nillable="true" type="xsd:string"/>
               <element name="mapDescription" nillable="true" type="xsd:string"/>
               <element name="fromCodeSystem_id" nillable="true" type="xsd:string"/>
               <element name="fromCodeSystem_name" nillable="true" type="xsd:string"/>
               <element name="fromCodeSystem_version" nillable="true"
type="xsd:string"/>
               <element name="toCodeSystem_id" nillable="true" type="xsd:string"/>
               <element name="toCodeSystem_name" nillable="true" type="xsd:string"/>
               <element name="toCodeSystem_version" nillable="true" type="xsd:string"/>
            </sequence>
         </complexType>
         <complexType name="ArrayOfSupportedMap">
            <complexContent>
               <restriction base="soapenc:Array">
                  <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:SupportedMap[]"/>
               </restriction>
            </complexContent>
         </complexType>
         <complexType name="UnexpectedError">
            <sequence>
               <element name="possible_cause" nillable="true" type="xsd:string"/>
            </sequence>
         </complexType>
         <complexType name="ConceptId">
            <sequence>
               <element name="codeSystem_id" nillable="true" type="xsd:string"/>
               <element name="concept_code" nillable="true" type="xsd:string"/>
            </sequence>
         </complexType>
         <complexType name="MappedConceptCode">
            <sequence>
               <element name="mappedConcept_id" nillable="true" type="impl:ConceptId"/>
               <element name="mapQuality_code" nillable="true" type="xsd:string"/>
            </sequence>
         </complexType>
```

```
        <complexType name="MappingNotAvailable">
           <sequence>
              <element name="fromCodeSystem_id" nillable="true" type="xsd:string"/>
              <element name="toCodeSystem_id" nillable="true" type="xsd:string"/>
           </sequence>
        </complexType>
        <complexType name="UnableToMap">
           <sequence/>
        </complexType>
        <complexType name="UnknownConceptCode">
           <sequence>
              <element name="concept_code" nillable="true" type="xsd:string"/>
           </sequence>
        </complexType>
        <complexType name="UnknownCodeSystem">
           <sequence>
              <element name="codeSystem_id" nillable="true" type="xsd:string"/>
           </sequence>
        </complexType>
        <complexType name="UnknownMapName">
           <sequence>
              <element name="map_name" nillable="true" type="xsd:string"/>
           </sequence>
        </complexType>
        <complexType name="AmbiguousMapRequest">
           <sequence>
              <element maxOccurs="unbounded" name="possible_maps" nillable="true"
type="xsd:string"/>
           </sequence>
        </complexType>
        <complexType name="CTSVersionId">
           <sequence>
              <element name="major" type="xsd:short"/>
              <element name="minor" type="xsd:short"/>
           </sequence>
        </complexType>
     </schema>
  </wsdl:types>
  <wsdl:message name="UnknownMapName">
     <wsdl:part name="fault" type="impl:UnknownMapName"/>
  </wsdl:message>
  <wsdl:message name="AmbiguousMapRequest">
     <wsdl:part name="fault" type="impl:AmbiguousMapRequest"/>
  </wsdl:message>
  <wsdl:message name="getServiceDescriptionResponse">
     <wsdl:part name="getServiceDescriptionReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="MappingNotAvailable">
     <wsdl:part name="fault" type="impl:MappingNotAvailable"/>
  </wsdl:message>
  <wsdl:message name="getServiceVersionRequest">

  </wsdl:message>
  <wsdl:message name="getSupportedMapsResponse">
     <wsdl:part name="getSupportedMapsReturn" type="impl:ArrayOfSupportedMap"/>
  </wsdl:message>
  <wsdl:message name="UnknownConceptCode">
     <wsdl:part name="fault" type="impl:UnknownConceptCode"/>
  </wsdl:message>
  <wsdl:message name="UnexpectedError">
     <wsdl:part name="fault" type="impl:UnexpectedError"/>
  </wsdl:message>
```

```
   <wsdl:message name="UnknownCodeSystem">
      <wsdl:part name="fault" type="impl:UnknownCodeSystem"/>
   </wsdl:message>
   <wsdl:message name="UnableToMap">
      <wsdl:part name="fault" type="impl:UnableToMap"/>
   </wsdl:message>
   <wsdl:message name="getCTSVersionRequest">

   </wsdl:message>
   <wsdl:message name="getServiceNameResponse">
      <wsdl:part name="getServiceNameReturn" type="xsd:string"/>
   </wsdl:message>
   <wsdl:message name="getServiceNameRequest">

   </wsdl:message>
   <wsdl:message name="getServiceDescriptionRequest">

   </wsdl:message>
   <wsdl:message name="mapConceptCodeRequest">
      <wsdl:part name="fromConcept_id" type="impl:ConceptId"/>
      <wsdl:part name="toCodeSystem_id" type="xsd:string"/>
      <wsdl:part name="map_name" type="xsd:string"/>
   </wsdl:message>
   <wsdl:message name="mapConceptCodeResponse">
      <wsdl:part name="mapConceptCodeReturn" type="impl:MappedConceptCode"/>
   </wsdl:message>
   <wsdl:message name="getSupportedMapsRequest">

   </wsdl:message>
   <wsdl:message name="getCTSVersionResponse">
      <wsdl:part name="getCTSVersionReturn" type="impl:CTSVersionId"/>
   </wsdl:message>
   <wsdl:message name="getServiceVersionResponse">
      <wsdl:part name="getServiceVersionReturn" type="xsd:string"/>
   </wsdl:message>
   <wsdl:portType name="CodeMappingOperations">
      <wsdl:operation name="getSupportedMaps">
         <wsdl:input message="impl:getSupportedMapsRequest"
name="getSupportedMapsRequest"/>
         <wsdl:output message="impl:getSupportedMapsResponse"
name="getSupportedMapsResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="mapConceptCode" parameterOrder="fromConcept_id
toCodeSystem_id map_name">
         <wsdl:input message="impl:mapConceptCodeRequest"
name="mapConceptCodeRequest"/>
         <wsdl:output message="impl:mapConceptCodeResponse"
name="mapConceptCodeResponse"/>
         <wsdl:fault message="impl:AmbiguousMapRequest" name="AmbiguousMapRequest"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:UnableToMap" name="UnableToMap"/>
         <wsdl:fault message="impl:MappingNotAvailable" name="MappingNotAvailable"/>
         <wsdl:fault message="impl:UnknownMapName" name="UnknownMapName"/>
         <wsdl:fault message="impl:UnknownConceptCode" name="UnknownConceptCode"/>
         <wsdl:fault message="impl:UnknownCodeSystem" name="UnknownCodeSystem"/>
      </wsdl:operation>
      <wsdl:operation name="getCTSVersion">
         <wsdl:input message="impl:getCTSVersionRequest" name="getCTSVersionRequest"/>
         <wsdl:output message="impl:getCTSVersionResponse"
name="getCTSVersionResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
```

```
      </wsdl:operation>
      <wsdl:operation name="getServiceDescription">
         <wsdl:input message="impl:getServiceDescriptionRequest"
name="getServiceDescriptionRequest"/>
         <wsdl:output message="impl:getServiceDescriptionResponse"
name="getServiceDescriptionResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceName">
         <wsdl:input message="impl:getServiceNameRequest"
name="getServiceNameRequest"/>
         <wsdl:output message="impl:getServiceNameResponse"
name="getServiceNameResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceVersion">
         <wsdl:input message="impl:getServiceVersionRequest"
name="getServiceVersionRequest"/>
         <wsdl:output message="impl:getServiceVersionResponse"
name="getServiceVersionResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="CodeMappingServiceSoapBinding"
type="impl:CodeMappingOperations">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getSupportedMaps">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getSupportedMapsRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="getSupportedMapsResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
         </wsdl:output>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="mapConceptCode">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="mapConceptCodeRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="mapConceptCodeResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
         </wsdl:output>
         <wsdl:fault name="AmbiguousMapRequest">
            <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
         </wsdl:fault>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
         </wsdl:fault>
         <wsdl:fault name="MapNameSourceMismatch">
            <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
```

```
        </wsdl:fault>
        <wsdl:fault name="MapNameTargetMismatch">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
        <wsdl:fault name="UnableToMap">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
        <wsdl:fault name="MappingNotAvailable">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
        <wsdl:fault name="UnknownMapName">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
        <wsdl:fault name="UnknownConceptCode">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
        <wsdl:fault name="UnknownCodeSystem">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
     </wsdl:operation>
     <wsdl:operation name="getCTSVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getCTSVersionRequest">
           <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:input>
        <wsdl:output name="getCTSVersionResponse">
           <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:output>
        <wsdl:fault name="UnexpectedError">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
     </wsdl:operation>
     <wsdl:operation name="getServiceDescription">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getServiceDescriptionRequest">
           <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:input>
        <wsdl:output name="getServiceDescriptionResponse">
           <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:output>
        <wsdl:fault name="UnexpectedError">
           <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
     </wsdl:operation>
     <wsdl:operation name="getServiceName">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getServiceNameRequest">
           <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:input>
```

```
        <wsdl:output name="getServiceNameResponse">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:output>
        <wsdl:fault name="UnexpectedError">
          <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getServiceVersion">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getServiceVersionRequest">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:input>
        <wsdl:output name="getServiceVersionResponse">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:output>
        <wsdl:fault name="UnexpectedError">
          <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn://hl7.org/CTSVAPI" use="encoded"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CodeMappingOperationsService">
      <wsdl:port binding="impl:CodeMappingServiceSoapBinding"
name="CodeMappingService">
        <wsdlsoap:address
location="http://localhost:8080/axis/services/CodeMappingService"/>
      </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

## 14 Сводные требования к службам

### 14.1 Введение

В настоящем разделе приведены сводные требования к широкому спектру терминологических служб, взятых из материалов и опыта применения других релевантных работ, указанных в разделе 2.

Детализированные предложения, приведенные в настоящем стандарте, относятся к ограниченному подмножеству таких служб, а именно к тем службам, которые имеют наиболее прямое отношение к поддержке реализаций стандарта HL7 Версии 3.

### 14.2 Информационные службы сервера

#### 14.2.1 Введение

Данный подраздел посвящен службам, предоставляющим информацию о сервере терминологий и терминологиях, к которым он обеспечивает доступ.

**14.2.2 Предоставление информации о версии сервера и его возможностях**

Вариант использования: клиентскому приложению требуется соответствующим образом взаимодействовать с серверами, реализующими разные версии спецификации терминологического сервера.

Все спецификации подвержены эволюции и при их конкретных реализациях могут предоставляться дополнительные возможности. Поэтому существенно необходимы не зависящие от версии средства определения возможностей сервера.

**14.2.3 Предоставление информации о терминологиях, доступных с помощью сервера**

Вариант использования: клиентскому приложению требуется определить, можно ли пользоваться серверам для доступа к определенной терминологии.

Отдельный экземпляр сервера может предоставлять доступ к одной или нескольким терминологиям, и клиентское приложение может иметь доступ к нескольким альтернативным серверам терминологий.

Эта услуга может предоставляться в форме проверки доступа к конкретной терминологии (например, поддерживает ли сервер терминологию X) или в форме запроса списка поддерживаемых терминологий.

### 14.3 Службы, предоставляющие метаданные терминологий

#### 14.3.1 Введение

Данный подраздел посвящен службам, предоставляющим информацию о терминологии, поддерживаемой сервером.

Во всех вариантах использования, обсуждаемых в настоящем подразделе, при каждом вызове службы требуется доступ к одной заданной терминологии. В каждом обращении к службе указана одна идентифицированная терминология, доступная с помощью сервера.

#### 14.3.2 Получение информации о терминологии, доступной с помощью сервера

Вариант использования: служба позволяет клиентскому приложению определить, предоставляет ли сервер достаточно полные, актуальные и аутентичные данные.

Клиентское приложение должно быть способно иметь доступ к различным свойствам терминологий, поддерживаемых серверов (например, номера версий, даты выпусков, авторизованный источник, доступность в полном или ограниченном объеме).

#### 14.3.3 Получение детальных сведений о свойствах понятий или терминов, содержащихся в данной терминологии

Вариант использования: служба позволяет клиентскому приложению получить сведения о доступных свойствах каждого понятия или термина, содержащегося в данной терминологии, чтобы подходящим образом обработать или визуализировать их.

Некоторые терминологии имеют более широкие совокупности свойств, ассоциированных с каждым понятием. Сервер может предоставлять доступ ко всем этим свойствам или только к какой-то их части. Клиентскому приложению необходимо знать, какие именно свойства доступны с помощью этого сервера.

#### 14.3.4 Получение сведений о типах отношений, используемых в данной терминологии

Вариант использования: служба позволяет клиентскому приложению получить сведения о типах отношений, поддерживаемых данной терминологией, чтобы оно могло обратиться с правильными запросами сведений об отношениях.

Некоторые терминологии поддерживают разнообразные типы отношений между терминами и/или понятиями. Например:

- некоторые термины могут быть синонимами, представляющими одно и то же понятие;
- понятие может быть связано с другим, более общим понятием (например, «пневмония» является подтипом «заболевания легких» и подтипом «инфекционного заболевания»);
- понятие может быть определено в терминах нескольких других понятий;
- понятие может быт ассоциировано с другими понятиями, квалифицирующими его.

#### 14.3.5 Получение сведений о критериях поиска, поддерживаемых данной терминологией

Вариант использования: служба позволяет клиентскому приложению получить сведения о разрешенных критериях поиска, которые поддерживаются данным сервером по отношению к конкретной доступной терминологии.

Существуют разнообразные потенциальные критерии поиска, которые могут быть применены для нахождения понятия или термина. Различные серверы могут использовать разные типы поддерживаемых ими критериев. Критерии, подходящие для поиска в конкретной доступной терминологии, также могут варьироваться.

### 14.4 Службы, предоставляющие доступ к содержанию терминологии

#### 14.4.1 Введение

Настоящий подраздел посвящен службам, обеспечивающим доступ к информационному наполнению терминологии.

В большинстве вариантов использования, обсуждаемых в настоящем подразделе, при каждом вызове службы требуется доступ к одной заданной терминологии. В каждом обращении к службе задается одна идентифицированная терминология, доступная с помощью сервера. Можно предложить альтернативный подход, при котором инициируется сеанс доступа к конкретной терминологии, в рамках которого выполняются все последовательные обращения к службам.

Некоторые варианты использования могли бы значительно выиграть при наличии возможности применения одного и того же запроса службы ко всем или избранному подмножеству терминологий, поддерживаемых сервером. Примером может служить поиск заданной фразы во всех терминах всех терминологий, поддерживаемых сервером. Однако это может привести к ряду существенных осложнений:

- серверу может понадобиться интеграция результатов поиска, проведенного в различных терминологиях-источниках;
- терминологии могут не обладать общей структурой, и поиск в них может быть не однотипным;
- для каждого извлеченного термина необходимо идентифицировать терминологии, в которых он найден.

**14.4.2 Получение свойств понятия или термина**

Требуется служба, возвращающая доступные свойства отдельного понятия или термина, извлекаемого с помощью уникального идентификатора. Результат возвращается в форме совокупности свойств заданного элемента. Могут возвращаться все свойства или только те, что были перечислены в запросе.

Вариант использования: служба позволяет клиентскому приложению изобразить или проверить текстовое значение понятия или термина, представленного уникальным идентификатором.

Если идентификатор хранится или передается без текстового описания, ассоциированного с ним, то такая служба необходима, чтобы клиентское приложение могло представить информацию в человекочитаемой форме. Если текстовое описание хранится или передается вместе с идентификатором, служба предоставляет возможность проверить, что идентификатору (являющемуся предпочтительным объектом для машинной обработки) действительно присвоено то текстовое значение, которое было передано.

Варианты использования: служба позволяет клиентскому приложению изобразить альтернативное текстовое значение идентифицированного понятия.

В некоторых терминологиях с одним понятием связано несколько терминов, являющихся синонимами и переводами на другие языки/диалекты. Сервер должен обеспечить клиентскому приложению возможность узнать, существуют ли такие варианты термина, и обеспечить к ним доступ.

Вариант использования: служба позволяет клиентскому приложению получить доступ к дополнительным свойствам, ассоциированным с понятием или термином, чтобы оно могло соответствующим образом обработать их или визуализировать.

В некоторых терминологиях с каждым понятием или термином могут быть ассоциированы дополнительные свойства. Например, понятие может иметь свойство, указывающее, что оно предназначено для текущего использования или оставлено для обеспечения обработки унаследованных данных. Термин может иметь свойство, указывающее, какому языку он принадлежит и является ли он предпочтительным термином или синонимом.

**14.4.3 Получение совокупности понятий или терминов, совпадающих с заданными текстовыми критериями**

14.4.3.1 Общий вариант использования

Вариант использования: служба позволяет клиентскому приложению найти понятие или термин по одному или нескольким введенным словам. Режимы ввода могут включать в себя клавиатурный ввод, распознавание речи или разбор текста электронного документа. Результатом должна быть совокупность совпадений, которая может использоваться клиентским приложением для формирования меню, списка или иного визуального компонента управления.

Спектр полезных критериев поиска может варьироваться в зависимости от размера, структуры и богатства терминологии. Ниже приведены некоторые критерии для разных вариантов использования.

Производительность службы поиска критична, поскольку она может возвращать много терминов и понятий для визуализации в режиме реального времени. Возвращение полных сведений о понятии или термине, включая все ассоциированные с ним свойства и отношения, не является необходимым и может существенно повлиять на производительность. Поэтому по умолчанию такая служба должна возвращать только текст совпадающего термина и ассоциированный с ним уникальный идентификатор.

14.4.3.2 Критерии текстового поиска

В зависимости от размера терминологии и режима ввода данных могут оказаться полезными различные критерии текстового поиска (чем объемнее терминология, тем более вариабельными и избирательными должны быть критерии). Ниже перечислены основные типы критериев текстового поиска, которые могут оказаться востребованными.

Совпадение с фразой:

- полное совпадение с фразой — полное словесное совпадение фразы и термина;
- совпадение фразы с началом термина — совпадение фразы с началом термина;
- совпадение фразы с термином — совпадение фразы с какой-либо частью термина.

Совпадение со словом:

- полное совпадение со словом — сравниваемое слово является частью термина;
- совпадение слова с началом слова в термине — слово (или его часть) совпадает с началом слова в термине;

- совпадение слова, дополненное шаблоном — совпадение с шаблоном проверяется только в контексте слова;
- совпадение словоформ (например, «туберкулез» и «туберкулезный»).

Совпадение с несколькими словами:
- применение одного из критерия совпадения со словом к каждому слову.

Вариации совпадения с несколькими словами:
- совпадение с несколькими словами без учета порядка — все набранные слова встречаются где-либо в термине;
- совпадение с несколькими словами с учетом порядка — все набранные слова встречаются где-либо в термине в том же порядке.

Совпадение с шаблоном:
- совпадение с шаблоном фразы — термин совпадает с регулярным выражением;
- совпадение с шаблоном слов — то же, что и совпадение слов, только искомое слово представлено регулярным выражением.

Модификации, применимые к большинству типов текстового поиска:
- поиск, зависящий/не зависящий от регистра;
- обработка диакритических и специальных знаков;
- определение разрывов слова при поиске по словам;
- фонетический поиск;
- совпадение слов/фраз/сокращений с аналогичным значением (например, «ренальный» и «почечный», «ИМ» и «инфаркт миокарда»).

Текстовый поиск может дополнительно модифицироваться критериями, приведенными в следующем пункте.

14.4.3.3 Дополнительные критерии, модифицирующие текстовый поиск

Для некоторых вариантов использования текстовый поиск должен модифицироваться с помощью дополнительных критериев. Следующие три типа критериев могут быть применимы к некоторым или всем терминологиям, поддерживаемым сервером.

Критерии статуса

Вариант использования: служба позволяет клиентскому приложению получать информацию только о понятиях или терминах, используемых в настоящее время.

Терминология может содержать понятия или термины, не предназначенные для текущего использования и оставленные в ней для обеспечения обработки унаследованных данных.

Критерии словарного домена

Вариант использования: служба позволяет клиентскому приложению ограничить запрашиваемые термины или понятия конкретным словарным доменом. Например, ограничить поиск теми значениями, которые допустимы для конкретного атрибута сообщения, соответствующего стандарту HL7.

Терминология может содержать значения, применимые в разном контексте и допустимые для использования в разных атрибутах конкретных сообщений, соответствующих стандарту HL7.

Для реализации таких критериев сервер должен иметь доступ к соответствующим таблицам словарных доменов HL7.

Критерии отношений

Вариант использования: служба позволяет клиентскому приложению ограничить поиск только теми терминами или понятиями, которые имеют конкретное отношение с другим понятием. Например, ограничить поиск слова «аппендикс» понятиями, которые имеют отношение подтипа с понятием «хирургическая процедура».

Терминология может содержать отношения, обеспечивающие полезный механизм уточнения поиска. Обеспечение поиска такого типа возможно только для некоторых терминологий, и степень интеллектуальности поиска может существенно варьироваться в зависимости от поддерживаемых отношений.

**14.4.4 Получение совокупности понятий или терминов, заданной отношениями**

14.4.4.1 Получение всех отношений заданного понятия или термина

Вариант использования: служба позволяет клиентскому приложению визуализировать отношения, делать семантические выводы или предлагать выбор только тех квалификаторов, которые применимы к данному понятию.

Функция выборки полной совокупности отношений может быть эффективной, если клиентское приложение должно просматривать много отношений. Однако следующая пара служб, которые ищут

отношения более селективно, может оказаться полезнее для специфичного просмотра отношений и навигации по отношениям.

14.4.4.2 Получение типов отношений, применяемых к идентифицированному понятию или термину

Вариант использования: служба позволяет клиентскому приложению определить типы доступных отношений до того, как выполнить выборку понятий с заданным отношением (см. ниже).

Такая служба не обеспечивает дополнительную функциональность по сравнению с получением всех отношений (см. выше). Однако она может быть более эффективной для идентификации типов отношений, которые могут быть обработаны индивидуально (см. ниже).

Некоторые типы отношений могут быть специфичными для конкретной терминологии. Однако служба должна обеспечивать однотипную обработку ряда более общих типов отношений, если они поддерживаются терминологией, например, отношения подтипа, то есть отношения между понятием и его синонимами.

14.4.4.3 Получение понятий/терминов, имеющих заданный тип отношения с понятием/термином

Вариант использования: служба позволяет клиентскому приложению предложить пользователю возможность навигации по заданным семантическим связям, например, для уточнения или для обобщения понятия.

Вариант использования: служба позволяет клиентскому приложению изобразить синонимы, ассоциированные с понятием.

Вариант использования: служба позволяет клиентскому приложению делать семантические выводы, используя отношение подтипа, например, подсчитывать экземпляры понятия «пневмония» в качестве экземпляров понятий «заболевание легких» и «инфекционное заболевание».

Такая служба не обеспечивает дополнительную функциональность по сравнению с получением всех отношений (см. выше). Однако такая избирательная служба может быть более эффективной для специфичного просмотра и навигации.

По умолчанию такая служба должна возвращать уникальный идентификатор и термин для каждого понятия или термина.

**14.4.5 Получение всех понятий или терминов, принадлежащих заданному словарному домену**

Вариант использования: служба позволяет клиентскому приложению получить полный набор значений, допустимых для конкретного атрибута сообщения, соответствующего стандарту HL7.

Эта служба близка к текстовому поиску с дополнительными ограничениями словарным доменом. Однако в данном случае возвращаются все понятия, принадлежащие словарному домену, без попытки выполнить текстовый поиск.

По умолчанию такая служба должна возвращать уникальный идентификатор и термин для каждого понятия или термина.

**14.4.6 Проверка уникального идентификатора**

14.4.6.1 Проверка действительности уникального идентификатора в терминологии

Вариант использования: служба позволяет клиентскому приложению проверить, что идентификаторы (или коды), входящие в состав сообщения, действительны в справочной терминологии.

Этот вариант использования может встретиться также при попытке извлечь из терминологии требуемое понятие или термин. Однако по сравнению с простой проверкой запрос на извлечение создает избыточную нагрузку и избыточные данные. Проверка может требоваться для многих сообщений, каждое из которых содержит много идентификаторов, поэтому нужна достаточно простая и эффективная служба проверки.

14.4.6.2 Дополнительные критерии, применимые при проверке идентификаторов

Критерии статуса

Вариант использования: служба позволяет клиентскому приложению проверить, используется ли в настоящее время идентификатор, содержащийся в сообщении или записи.

Терминология может содержать понятия или термины, не предназначенные для текущего использования и оставленные в ней для обеспечения обработки унаследованных данных.

Критерии словарного домена

Вариант использования: служба позволяет клиентскому приложению удостовериться, что идентификатор, переданный в сообщении, принадлежит соответствующему словарному домену. Например, ограничить поиск значениями, допустимыми для конкретного атрибута сообщения, соответствующего стандарту HL7.

Терминология может содержать значения, применимые в разном контексте и допустимые для использования в разных атрибутах конкретных сообщений, соответствующих стандарту HL7.

Для реализации таких критериев сервер должен иметь доступ к соответствующим таблицам словарных доменов HL7.

Критерии отношений

Вариант использования: служба позволяет клиентскому приложению проверить, должно ли понятие считаться подтипом другого понятия (например, понятие используется в алгоритме поддержки решений или в качестве критериев анализа).

Терминология может содержать отношения, которые можно использовать для проверки того факта, что одно понятие является подтипом другого понятия.

**14.4.7 Отображение и преобразование кодов и кодовых фраз**

14.4.7.1 Получение идентификатора с тем самым значением в другой терминологии

Вариант использования: служба позволяет клиентскому приложению транслировать идентификатор, используемый в его собственной среде, в терминологию, требуемую при формировании сообщения (и обратно).

Во многих случаях такая трансляция представляет собой нетривиальную задачу, поскольку коды из разных терминологий могут быть связаны отношением «один ко многим», и для выбора конкретного значения может потребоваться дополнительная контекстная информация. Но если словарные домены содержат относительно небольшое число значений и перечислены отображения этих значений, имеющие кратность «один к одному» (или «многие к одному»), такая служба может оказаться достаточно полезной.

14.4.7.2 Декомпозиция понятия в посткоординируемое выражение

Вариант использования. служба позволяет клиентскому приложению разбить код комплексного понятия на составные коды, отражающие разные аспекты значения этого понятия. Например, представить «аппендэктомию» как «хирургическое удаление» «червеобразного отростка».

В некоторых терминологиях, включающих в себя как простые, так и комплексные понятия, предусмотрены отношения, позволяющие провести такое преобразование. Такая служба может быть полезной при формировании сообщений, представляющих различные аспекты комплексного понятия в виде отдельных кодированных элементов. Конечным результатом вызова такой службы будет выражение (называемое кодовой фразой), состоящее из пар «атрибут — значение» (возможно, вложенных). Клиентское приложение должно затем распределить эти элементы по атрибутам сообщения.

Реализация этой службы не обязательно должна строго соответствовать такой форме. Можно реализовать альтернативный подход, при котором для каждого формируемого атрибута сообщения запрашиваются специфичные типы отношений.

14.4.7.3 Преобразование посткоординированного выражения в прекоординированное понятие

Вариант использования: служба позволяет клиентскому приложению преобразовать ряд идентификаторов, представляющих различные компоненты более сложного понятия, в простой идентификатор понятия.

Это преобразование является обратным по отношению к декомпозиции. Преобразование посткоординированного выражения в код простого понятия может оказаться невозможным, поскольку может не существовать простого понятия, представляющего полное значение этого выражения. Поэтому результатом вызова службы может оказаться то же самое выражение и другое посткоординированное выражение, состоящее из меньшего числа компонентов. Например, выражение «экстренное» «хирургическое удаление» «червеобразного отростка» может быть преобразовано в выражение «экстренная» «аппендэктомия».

## 15 Системы кодирования, используемые в API ОТС

В таблицах 33 и 34 приведены системы кодирования, используемые в API ОТС.

Таблица 33 — Перечень систем кодирования

| Имя системы кодирования | ОИД | Описание | Примеры значений |
|---|---|---|---|
| CodeSystem | 2.16.840.1.113883.5.22 | Совокупность систем кодирования, известных HL7 | CAS (химические абстрактные коды)<br>IETF1766 (идентификаторы языков)<br>CodeSystem (система кодирования) |

*Окончание таблицы 33*

| Имя системы кодирования | ОИД | Описание | Примеры значений |
|---|---|---|---|
| ConceptStatus | 2.16.840.1.113883.5.1086 | Статус записи словарного домена, характеризующий состояние ее включения в базу данных спецификаций доменов HL7 или пересмотра | A (Active — активная)<br>D (Deleted — удаленная)<br>P (Proposed — предложенная)<br>R (Retired — отмененная) |
| VocabularyDomainQualifier | 2.16.840.1.113883.5.147 | Расширяемость кодирования указывает, разрешаются ли исключения в домене кодированного атрибута | CWE (coded with exceptions — кодированный с исключениями)<br>CNE (coded no exceptions — кодированный без исключений) |
| Language | 2.16.840.1.113883.6.99 | Человеческий язык, используемый в текстовых описаниях или при коммуникации. Коды берутся из ИСО 639 | EN (English — английский)<br>DE (German — немецкий)<br>EN-US (English-US dialect — американский диалект английского)<br>EN-GB (English-GB dialect — британский диалект английского) |
| TranslationQuality | 2.16.840.1.113883.5.1093 | Отношения между понятиями между двумя системами кода | Exact (точное совпадение)<br>Broader Than (шире)<br>Narrower Than (уже)<br>Different (различны) |
| ConceptProperty | 2.16.840.1.113883.5.1087 | Идентификаторы свойств понятий | openIssue (открытый вопрос)<br>appliesTo (применяется к)<br>howApplies (как применяется)<br>OID (ОИД) |
| ConceptCodeRelationship | 2.16.840.1.113883.5.1088 | Отношения между двумя понятиями одной и той же системы кодирования | hasSubtype<br>hasPart<br>smallerThan |
| MatchAlgorithm | 2.16.840.1.113883.5.1094 | Алгоритм совпадения, используемый при поиске в тексте | Identical<br>IdenticalIgnoreCase<br>StartsWith<br>StartsWithIgnoreCase<br>EndsWith<br>EndsWithIgnoreCase<br>ContainsPhrase<br>ContainsPhraseIgnoreCase<br>WordsAnyOrder<br>WordsAnyOrderIgnoreCase<br>WildCards<br>WildCardsIgnoreCase<br>RegularExpression |

Таблица 34 — Основные коды отношений

| Код | Обозначение | Описание | Транзитивность | Инверсия |
|---|---|---|---|---|
| hasSubtype | Имеет подтип | Отношение подтипа, не специфицированное иным образом | Да | isSubtypeOf |
| hasPart | Является частью | Отношение части к целому, не специфицированное иным образом | Да | isPartOf |
| smallerThan | Меньше | Общее отношение упорядочения (меньше) | Да | greaterThan |

## 16 Спецификация API ОТС на языке IDL

### 16.1 Базовые типы данных HL7

Ниже приведено определение на языке IDL базовых типов данных HL7, используемых в ОТС.

```
/* Исходный файл: types.idl
 */

#ifndef  __types_IDL_
#define  __types_IDL_

/*
 * Определение на языке IDL подмножества типов данных HL7 Версии 3.
 * Версия: V3 Ballot Cycle 5
 *
 *   В настоящее время этот модуль специально предназначен для
 *   использования в API Централизованных терминологических служб,
 *   хотя это формальное отображение спецификации типов
 *   может найти более широкое применение.
 *
 *   Примечание - Некоторые целевые языки не чувствительны к регистру.
 *   Как следствие, язык IDL не позволяет записывать несколько представлений
 *   одного и того же имени в разных регистрах. Чтобы максимально следовать
 *   этому правилу, к именам в нижнем регистре, конфликтующим с именами в
 *   верхнем регистре, добавляется суффикс "_value".
 *
 * Примечание - Из базовых типов данных удалены аспекты причины пустоты
 * nullFlavor.
 */
module types {

        typedef boolean bl_value;
        struct BL {
          bl_value v;
        };

        typedef long int_value;
        struct INT {
          int_value v;
        };

        typedef string uid_value;
        struct UID {
          uid_value v;
        };
        typedef sequence <UID> LIST_UID;

        typedef string st_value;
        struct ST {
          st_value v;
        };

        typedef string ts_value;
        struct TS {
```

```
        ts_value TS;
      };

  typedef sequence<octet> bin_value;

      enum cs_BinaryDataEncoding {
            B64,
            TXT
      };

      typedef string cs_value;

  /* Следующий фрагмент в действительности является объединением,
но для простоты вместо него дано явное представление
      union binary_or_text switch(cs_BinaryDataEncoding) {
            case B64: bin_value binaryValue;
            case TXT: st_value textualValue;
      }; */
  struct binary_or_text {
    cs_BinaryDataEncoding itemType;
    bin_value binaryValue;
    st_value  textualValue;
  };

/* Примечание - Неуклюжее объявление sequence<ED> позволяет компилятору
выполнить рекурсию. Кратность 0..1 */
      struct ED {
            binary_or_text this;
            cs_value encoding;
            cs_value mediaType;
            cs_value compression;
            bin_value integrityCheck;
            cs_value reference;
            cs_value ingegrityCheckAlgorithm;
            cs_value charset;
            cs_value language;
            sequence<ED> thumbnail;
      };

      struct CS {
    cs_value code;
            ED originalText;
            cs_value codingRationale;
      };


      struct CV {
            cs_value code;
            uid_value codeSystem;
            st_value codeSystemName;
            st_value codeSystemVersion;
            st_value displayName;
            ED originalText;
            CS codingRationale;
      };
```

```
        typedef sequence <CV> SET_CV;

/* Примечание - Фактически значение имеет тип данных CD, но из-за рекурсии
такое объявление здесь не используется */
        struct CR {
          CV name;
          CV value;
          BL inverted;
        };
        typedef sequence <CR> LIST_CR;

        struct CD {
                cs_value code;
                uid_value codeSystem;
                st_value codeSystemName;
                st_value codeSystemVersion;
                st_value displayName;
                LIST_CR qualifiers;
                ED originalText;
                sequence <CD> translation;
                CS codingRationale;
        };
        typedef sequence <CD> SET_CD;

        struct CE {
                cs_value code;
                uid_value codeSystem;
                st_value codeSystemName;
                st_value codeSystemVersion;
                st_value displayName;
                ED originalText;
                SET_CD translation;
                CS codingRationale;
        };


};

#endif
```

### 16.2 Определение API сообщений ОТС на языке IDL

Ниже приведено определение API сообщений ОТС на языке IDL (файл CTSMAPI.idl).

```
/*  Source file: CTSMAPI.idl
 */

#ifndef   __CTSMAPI_DEFINED
#define   __CTSMAPI_DEFINED


#include "types.idl"


/*   API Централизованных терминологических служб на уровне сообщений
 *   Этот модуль определяет интерфейс между приложением, обрабатывающим
```

```
 * сообщения HL7, и терминологическими службами.
 */

module CTSMAPI {
/*********************************************
 * Соглашение о суффиксах:
 * Xxx    - суффикс типа
 * _xxx   - суффикс атрибута
 *
  Суффиксы
 * Id   - уникальный идентификатор сущности
 * Name   - уникальное имя
 * Code   - код понятия
 */

/*********************************************
 *   Базовые идентификаторы сущностей                *
 *********************************************/
/* HL7SpecBlock:basicData */
/* Спецификация идентификатора версии ОТС */
  struct CTSVersionId {
    types::INT  major;
    types::INT  minor;
  };

/* ИСО ОИД, уникально обозначающий систему кодирования */
  typedef types::UID    CodeSystemId;

/* Уникальное имя системы кодирования */
  typedef types::ST  CodeSystemName;

/* Код понятия, уникальный в системе кодирования */
  typedef types::ST  ConceptCode;

/* Уникальное имя словарного домена */
        typedef types::ST     VocabularyDomainName;
        typedef sequence<VocabularyDomainName> VocabularyDomainNameList;

/* Идентификатор набора значений */
        typedef types::UID    ValueSetId;

/* Имя набора значений */
        typedef types::ST     ValueSetName;


/* Уникальный идентификатор понятия, состоящий из идентификатора системы
   кодирования и кода понятия */
  struct ConceptId {
    CodeSystemId  codeSystem_id;
    ConceptCode   concept_code;
  };

/* Конкретный выпуск системы кодирования */
  typedef types::ST ReleaseVersionId;
  typedef sequence<ReleaseVersionId> ReleaseVersionIdList;
```

```
/* Непрозрачный контекст для итерационного раскрытия кода
 */
  typedef types::bin_value ExpansionContext;

/* HL7SpecBlockEnd */


/*********************************************
 *      Кодированные элементы                *
 *********************************************/

/* Идентификатор языка
 *   Синтаксис основан на IETF RFC 3066 - теги для идентификации языков
 *   Состоит из основного субтега <primary subtag>, за которым следуют
 *   нуль или более вторичных субтегов <secondary subtag>
 *   <primary subtag>  - по возможности берется из ИСО 639 часть 1
 *                       (2 символа), иначе берется трехсимвольный код
 *                     - в этой спецификации коды "i-" и "x-" не разрешены
 *
 *   <secondary subtag 1> - 2-8 символов
 *             - двухсимвольные коды берутся из ИСО 3166 alpha-2 коды стран
 *               коды длиной 3-8 символов берутся из регистра IANA - см.
 *   http://www.iana.org/assignments/language-tags
 *   <secondary subtag 2-n> - общих правил нет, некоторые субтеги
 *                            зарегистрированы в IANA
 */
 /* HL7SpecBlock:conceptCodes */
  typedef ConceptCode LanguageCode;


/*   Идентификатор отношения
 *   Система кодирования: отношения, определенные в стандарте HL7,
 *   берутся из системы кодирования ConceptCodeRelationship
 *   (2.16.840.1.113883.5.1088)
 *   Если в ней нет нужного эквивалента, можно использовать внешнюю
 *   систему.
 */
      typedef ConceptCode RelationshipCode;

/*   Область применения словарного домена (сфера, геополитическая единица,
 *   специальность и т. д.
 *   Система кодирования : VocabularyDomainQualifier (2.16.840.1.113883.5.147)
 *   (подтипы словарного домена RealmOfUse)
 */
    typedef ConceptCode ApplicationContextCode;

/* Data type code (CD, CE, CS, etc)
 *Code System:  DataType (2.16.840.1.113883.5.1007)
 */
  typedef ConceptCode DataTypeCode;

/*   Сила кодирования (CNE, CWE)
 *   Система кодирования: VocabularyDomainQualifier (2.16.840.1.113883.5.147)
 */
  typedef ConceptCode CodingStrengthCode;
```

```
/*   Тип узла набора значений (A, L, S)
 *   Система кодирования: ConceptGenerality (2.16.840.1.113883.5.24)
 */
  typedef ConceptCode ValueSetNodeTypeCode;

 /*
  *  Тип системы кодирования (E, L, I)
  *  Система кодирования:  CodeSystemType (2.16.840.1.113883.5.1085)
  */
  typedef ConceptCode CodeSystemTypeCode;

 /*
  *  Алгоритм совпадения
  *  Система кодирования:  MatchAlgorithm (2.16.840.1.113883.5.1094)
  */
  typedef ConceptCode MatchAlgorithmCode;
  typedef sequence<MatchAlgorithmCode> MatchAlgorithmCodeList;
/* HL7SpecBlockEnd */

/*************************************
 * Сущности, представленные в UML-модели*
 *************************************/


/* HL7SpecBlock:codeSystemDescriptor */

/* CodeSystemDescriptor - описатель системы кодирования
 * codeSystem_id          - OID системы кодирования
 * codeSystem_name        - мнемоническое имя системы кодирования
 * copyright              - авторские права на систему кодирования
 * availableReleases      - выпуск представленной системы кодирования
 */
  struct CodeSystemDescriptor {
    CodeSystemId     codeSystem_id;
    CodeSystemName   codeSystem_name;
    types::ST   copyright;
    ReleaseVersionIdList      availableReleases;
  };
  typedef sequence<CodeSystemDescriptor> CodeSystemDescriptorList;

/* HL7SpecBlockEnd */

/* HL7SpecBlock:RIMCodedAttribute */

/* RIMAttributeId - идентификатор кодированного атрибута модели RIM.
 *                  Для его уникальности необходимы все три элемента:
 *      model_id              - идентификатор модели RIM
 *      class_name            - имя класса
 *      attribute_name        - имя атрибута в модели
 */
        struct RIMAttributeId {
                types::ST   model_id;
                types::ST   class_name;
                types::ST   attribute_name;
        };
```

```
 /* RIMCodedAttribute          - кодированный атрибут модели RIM
  *  RIMAttribute_id           - уникальный идентификатор атрибута (модель,
  *                              класс, атрибут)
  *  dataType_code             - специфичный тип данных атрибута (CD, CE, CS, ...)
  *  codingStrength_code       - кодированное значение, представляющее силу
  *                              кодирования атрибута (CWE и т. д.)
  *  vocabularyDomain_name     - словарный домен атрибута (если есть)
  */
  struct RIMCodedAttribute {
    RIMAttributeId     RIMAttribute_id;
    DataTypeCode     dataType_code;
    CodingStrengthCode codingStrength_code;
    VocabularyDomainName     vocabularyDomain_name;
  };
  typedef sequence<RIMCodedAttribute> RIMCodedAttributeList;

/* HL7SpecBlockEnd */

/* HL7SpecBlock:valueSetDescriptor */
/* ValueSetDescriptor    - описатель набора значений. Набор значений может быть
 *                         уникально идентифицирован своим идентификатором
 *                         или именем. Имена присвоены не всем наборам значений.
 * valueSet_id           - уникальный идентификатор набора значений
 * valueSet_name         - уникальное имя набора значений (есть не у всех
 *                         наборов значений)
 */
  struct ValueSetDescriptor {
    ValueSetId  valueSet_id;
    ValueSetName  valueSet_name;
  };
  typedef sequence<ValueSetDescriptor> ValueSetDescriptorList;


/* HL7SpecBlockEnd */

/* HL7SpecBlock:vocabularyDomainDescription */
/* VocabularyDomainValueSet    - ассоциация словарного домена с набором
 *                               значений в необязательном контексте
 *  definedByValueSet          - идентификатор набора значений и необязательно имя,
 *                               идентифицирующие словарный домен
 * appliesInContext            - контекст, в котором применим набор значений
 */

       struct VocabularyDomainValueSet {
              ValueSetDescriptor        definedByValueSet;
              ApplicationContextCode          applicationContext_code;
       };
       typedef sequence<VocabularyDomainValueSet> VocabularyDomainValueSetList;


/* VocabularyDomainDescription  - полное описание словарного домена
 * для целей просмотра
 * vocabularyDomain_name        - уникальное имя домена
 * description                  - описание цели и назначения домена
 * restrictsDomain_name         - словарный домен, являющийся суперклассом
 *                                данного домена (если имеется)
```

```
 * basisOfDomains          - словарные домены, являющиеся подклассами данного домена
 * (если имеется)
 * constrainsAttributes    - атрибуты, непосредственно ограничивающие данный
 * домен
 * representedByValueSets - список ссылок на наборы значений
 */
        struct VocabularyDomainDescription {
                VocabularyDomainName            vocabularyDomain_name;
                types::ST                  description;
                VocabularyDomainName            restrictsDomain_name;
                sequence<VocabularyDomainName>  basisOfDomains;
                sequence<RIMCodedAttribute>     constrainsAttributes;
                VocabularyDomainValueSetList    representedByValueSets;
        };

/* HL7SpecBlockEnd */
/* HL7SpecBlock:valueSetDescription */
/* ValueSetConstructor - набор значений, используемый как часть другого
 *                       набора значений
 * includeHeadCode     - TRUE указывает, что головной код head_code является
 *                       частью конструируемого набора значений
 *                       FALSE указывает, что не является
 * includedValueSet    - набор значений, используемый для определения
 *                       конструируемого набора
 */
  struct ValueSetConstructor {
    ValueSetDescriptor includedValueSet;
    types::BL   includeHeadCode;
  };
  typedef sequence<ValueSetConstructor> ValueSetConstructorList;

/* ValueSetCodeReference   - ссылка на код понятия, включаемого в набор
 *                           значений
 * referenced_code         - код из системы кодирования, содержащей включаемое
 *                           значение
 * includeReferencedCode   - TRUE указывает, что ссылочный код является частью
 *                           набора значений. FALSE указывает, что не сам
 *                           является, а только связанные с ним понятия
 * leafOnly     - TRUE указывает, что включаются только конечные узлы (листья).
 *                FALSE указывает, что промежуточные узлы тоже включаются
 * relationship - код, идентифицирующий используемое отношение. Если указан,
 *                то все потомки ссылочного кода referencedCode включаются
 *                в набор (если не отфильтрованы ограничениями, задаваемыми
 *                полями includeReferencedCode и leafOnly)
 */
  struct ValueSetCodeReference {
    ConceptCode      referenced_code;
    RelationshipCode   relationship_code;
    types::BL   includeReferencedCode;
    types::BL   leafOnly;

  };
  typedef sequence<ValueSetCodeReference> ValueSetCodeReferenceList;


/* ValueSetDescription - описание набора значений
```

```
 * idAndName    - идентификатор и имя набора значений (если таковое имеется)
 *      description           - описание набора значений и его применения
 *      definingExpression    - полуформализованное выражение, определяющее
 *                              содержание набора значений (если таковое
 *                              имеется)
 *      basedOnCodeSystem - идентификатор, имя и версия базовой системы
 *                          кодирования, используемой для конструирования
 *                          этого набора данных

 *      allCodes          - TRUE указывает, что набор включает в себя все
 *                          коды названной системы кодирования
 *                          FALSE указывает, что берется избранное
 *                          подмножество
 * head_code    - код понятия из системы кодирования, представляющий весь
 *                набор значений (если таковой код имеется)
 */
  struct ValueSetDescription {
    ValueSetDescriptor idAndName;
    types::ST   description;
    types::ST   definingExpression;
    CodeSystemDescriptor     basedOnCodeSystem;
    types::BL   allCodes;
    ConceptCode      head_code;
  };


/* FullValueSetDescription - полное описание набора значений, используемое
 *                           для просмотра
 * description      - описание набора значений
 * constructedUsing - наборы значений, используемые для определения
 *                    данного набора
 * usedToDefine     - наборы значений, в определении которых участвует
 *                    данный набор
 * referencesCodes  - конкретные коды, которые включаются в данный набор
 *                    или исключаются из него
 */
  struct FullValueSetDescription {
    ValueSetDescription            description;
    ValueSetConstructorList  constructedUsingValueSets;
    ValueSetDescriptorList         usedToDefine;
    ValueSetCodeReferenceList      referencesCodes;
  };

/* HL7SpecBlockEnd */

/* HL7SpecBlock:codeSystemRegistration */

/*
 * CodeSystemRegistration - регистрационная запись системы кодирования
 * sponsor     - член комитета HL7 или организация, выступающая спонсором
 *               системы кодирования
 * publisher   - лицо или организация, публикующая систему кодирования
 * versionReportingMethod - метод обновления системы кодирования
 * licensingInformation   - описание требуемых лицензий и способа приобретения
*  inUMLS    - TRUE указывает, что система кодирования включена в
*              Унифицированную систему медицинского языка UMLS
```

```
 * systemSpecificLocatorInfo - сведения, используемые для дальнейшей
 *                             идентификации системы кодирования в контексте
 *                             издателя. Могут представлять собой номер
 *                             таблицы, подраздела и т. д.
 * codeSystemType_code - одно из значений E (внешняя), I (внутренняя),
 *                       EI (внешняя, но управляется комитетом HL7)
 */
  struct CodeSystemRegistration {
    types::ST   sponsor;
    types::ST   publisher;
    types::ST   versionReportingMethod;
    types::ST   licensingInformation;
    types::BL   inUMLS;
    types::ST   systemSpecificLocatorInfo;
    CodeSystemTypeCode codeSystemType_code;
  };

/*
 * CodeSystemInfo     - полная информация о системе кодирования
 * description        - имя, идентификатор и поддерживаемые версии
 *                      системы кодирования
 * registrationInfo   - дополнительная информация о регистрации,
 *                      предоставленная комитетом HL7
 */

  struct CodeSystemInfo {
    CodeSystemDescriptor     description;
    CodeSystemRegistration   registrationInfo;
  };

/* HL7SpecBlockEnd */

/*********************************************
 *    Конструкции, специфичные для сообщений    *
 *********************************************/

/* HL7SpecBlock:validateCodeBlock */

/* ValidationDetail - сведения об ошибке проверки кода
 * codeInError       - часть кодированного атрибута, в которой обнаружена ошибка
 *                     (базовый код, трансляция, имя или значение квалификатора)
 *      isError         - TRUE указывает, что эта ошибка считается достаточно
 *                        существенной, чтобы признать сообщение, содержащее
 *                        этот атрибут, неправильным
 *                        FALSE указывает, что ошибка не является фатальной,
 *                        но представляет потенциальную угрозу для построения
 *                        сообщения
 *      error_id        - уникальный идентификатор ошибки
 *      errorText       - текст, ассоциированный с идентификатором ошибки
 */
       struct ValidationDetail {
         types::CD codeInError;
              types::BL isError;
              types::ST error_id;
              types::ST errorText;
       };
```

```
        typedef sequence<ValidationDetail> ValidationDetailList;

/* ValidateCodeReturn   - структура возвращаемого значения
 * nErrors              - число обнаруженных ошибок
 * nWarnings            - число обнаруженных предупреждений
 * detail               - список отдельных ошибок и/или предупреждений
 */
  struct ValidateCodeReturn {
    types::INT nErrors;
    types::INT nWarnings;
    ValidationDetailList detail;
  };

/* HL7SpecBlockEnd */

/* HL7SpecBlock:valueSetExpansion */
/* ValueSetExpansion    - раскрытие набора значений в режиме реального времени
 *      pathLength      - целое значение, определяющее расстояние от
 *                        корневого набора значений
 * nodeType_code - тип узла набора значений (A, L или S)
 *      valueSet - идентификатор и имя (если оно имеется) набора значений,
 *                 ассоциированного с данным узлом
 *      concept_id          - система кодирования и код понятия, используемые
 *                            для этого узла (если имеются)
 *      displayName         - текст, ассоциированный с записью (если имеется)
 *      isExpandable        - TRUE, если этот узел можно раскрыть дальше
 *      expansionContext    - (непрозрачный) контекст, используемый для
 *                            дальнейшего раскрытия, если параметр isExpandable
 *                            имеет значение TRUE
 */
        struct ValueSetExpansion {
                types::INT          pathLength;
                ValueSetNodeTypeCode      nodeType_code;
                ValueSetDescriptor  valueSet;
                ConceptId           concept_id;
                types::ST           displayName;
                types::BL           isExpandable;
                ExpansionContext    expansionContext;
        };
        typedef sequence<ValueSetExpansion> ValueSetExpansionList;

/* HL7SpecBlockEnd */

/*****************************************************************
*       Исключения                                               *
*****************************************************************/
 /* HL7SpecBlock:exceptions */

/*
 * Не специфичная ошибка, воспрепятствовавшая успешному завершению вызова
 */
  exception UnexpectedError {
    types::ST possible_cause;
  };

/*
```

```
 * Имя словарного домена не распознано службой
 * vocabularyDomain_name - переданное имя, которое не распознано
 */
  exception UnknownVocabularyDomain {
                VocabularyDomainName vocabularyDomain_name;
        };

/*
 * Код контекста не распознан службой
 * context_code   - не распознанный код
 */
  exception UnknownApplicationContextCode {
    ApplicationContextCode    applicationContext_code;
  };


/*
 * Переданная идентификация набора значений не распознана
 *   valueSet  - идентификатор и/или имя набора значений
 */
        exception UnknownValueSet {
                ValueSetDescriptor valueSet;
        };

/*
 * В запросе переданы и имя, и идентификатор набора значений, но имя
 * не соответствует идентификатору

 */
        exception ValueSetNameIdMismatch {
          ValueSetId   valueSet_id;
          ValueSetName    valueSet_name;
        };

  exception UnknownConceptCode {
    ConceptId concept_id;
  };

  exception SubsumptionNotSupported {
    CodeSystemId codeSystem_id;
  };

  exception UnrecognizedQualifier {
    types::CR qualifier;
  };

  exception UnableToTranslate {
  };

  exception UnknownCodeSystem {
    CodeSystemId codeSystem_id;
  };

  exception UnknownLanguage {
    LanguageCode language_code;
  };
```

```
  exception InvalidExpansionContext {
  };

  exception QualifiersNotSupported {
  };

  exception NoApplicableValueSet {
    VocabularyDomainName      vocabularyDomain_name;
    ApplicationContextCode    applicationContext_code;
  };

  exception CodeSystemName!dMismatch {
    CodeSystemId      codeSystem_id;
    CodeSystemName    codeSystem_name;
  };

/*
 * Время выполнения операции превысило заданный лимит
 */
  exception TimeoutError {
  };

  /*
   * Формат искомого образца не может быть разобран
   */
  exception BadlyFormedMatchText {
    types::ST matchText;
  };


 /*
  * Код алгоритма совпадения не поддерживается службой
  */
  exception UnknownMatchAlgorithm {
    MatchAlgorithmCode matchAlgorithm_code;
  };


 /*
  * Для заданного кода и критериев языка не может быть найдено
  * подходящее обозначение
  */
  exception NoApplicableDesignationFound {
    types::CD codeToFillIn;
    types::ST displayLanguage_code;
  };

/* HL7SpecBlockEnd */


/************************************
 *  Раздел модуля идентификации   *
 ************************************/
/* HL7SpecBlock:identification */

  interface Identification {
```

```
        types::ST getServiceName() raises (UnexpectedError);
        types::ST getServiceVersion() raises (UnexpectedError);
        types::ST getServiceDescription() raises (UnexpectedError);
        types::ST getHL7ReleaseVersion() raises (UnexpectedError);
        CTSVersionId  getCTSVersion() raises (UnexpectedError);
      };

 /* HL7SpecBlockEnd */

/**********************************
 *  API времени исполнения        *
 **********************************/
 /* HL7SpecBlock:runtime */

      interface Runtime : Identification {

/* getSupportedMatchAlgorithms - возвратить список поддерживаемых алгоритмов
 * совпадения
 */
    MatchAlgorithmCodeList   getSupportedMatchAlgorithms()
       raises (UnexpectedError);

/* getSupportedVocabularyDomains - возвратить список словарных доменов,
 *                                 известных модулю времени исполнения
 *
 * matchText     - искомый образец. Формат зависит от алгоритма совпадения.
 *                 Пустая строка означает возвращение всех имен
 * matchAlgorithm_code   - код алгоритма совпадения.
 * timeout       - время ожидания завершения операции в миллисекундах
 *                 (0 - не ограничено)
 * sizeLimit     - максимальное число возвращаемых элементов
 *                 (0 - не ограничено)
 *
 * Возвращаемое значение  - список словарных доменов
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон.
 *
 * Исключения
 *   TimeoutError              - превышен лимит времени
 *   BadlyFormedMatchText      - искомый образец не может быть разобран для
 *                               заданного алгоритма
 *   UnknownMatchAlgorithm     - алгоритм совпадения не распознан службой
 */
    VocabularyDomainNameList getSupportedVocabularyDomains(
      in types::ST     matchText,
      in MatchAlgorithmCode  matchAlgorithm_code,
      in long     timeout,
           in long     sizeLimit
           )
           raises (BadlyFormedMatchText,
            UnknownMatchAlgorithm,
            TimeoutError,
            UnexpectedError
           );
```

```
/* HL7SpecBlockEndEllipsis */
/* HL7SpecBlock:validateCode */

/*  validateCode - проверить значение элемента с типом данных CD на
 *                 соответствие заданному словарному домену
 *
 *   vocabularyDomain_name   - словарный домен атрибута
 *   codeToValidate          - проверяемый код
 *   applicationContext_code - прикладной контекст (не обязателен)
 *   activeConceptsOnly      - TRUE указывает, что код должен быть активным
 *                             в целевой системе кодирования
 *                             FALSE указывает, что неактивные коды также
 *                             учитываются (Примечание - Если код неактивен,
 *                             то в этом случае должно генерироваться
 *                             предупреждение)
 *      errorCheckOnly       - TRUE указывает ограниченную проверку (только
 *                             на ошибки)
 *                             FALSE указывает расширенную проверку (на
 *                             ошибки и предупреждения)
 *
 *     Возвращаемое значение - структура, содержащая счетчик ошибок nErrors,
 *                             счетчик предупреждений nWarnings, за которыми
 *                             следует детальная информация
 *
 * Примечание - Этот метод обеспечивает проверку текущего кода и
 *              квалификаторов. Для проверки трансляций кода используйте
 *              метод validateTranslations
 *
 *Исключения:
 *   UnknownVocabularyDomain         - словарный домен не распознан службой
 *   UnknownApplicationContextCode   - прикладной контекст не распознан
 *   NoApplicableValueSet            - не существует набора значений, относящегося
 *                                     к этому домену в данном контексте
 *
 */
               ValidateCodeReturn validateCode(
                       in VocabularyDomainName       vocabularyDomain_name,
                       in types::CD                  codeToValidate,
                       in ApplicationContextCode     applicationContext_code,
                       in types::BL                  activeConceptsOnly,
                       in types::BL                  errorCheckOnly
                       )
                       raises (UnknownVocabularyDomain,
                             UnexpectedError,
                             UnknownApplicationContextCode,
                             NoApplicableValueSet);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:validateTranslation */

/* validateTranslation - проверить совокупность трансляций значения типа CD
 *vocabularyDomain_name         - словарный домен атрибута
 *      codeToValidate          - проверяемый код
 *      applicationContext_code - прикладной контекст (не обязателен)
*   activeConceptsOnly      - TRUE указывает, что код должен быть активным
```

```
*                           в целевой системе кодирования
*                           FALSE указывает, что неактивные коды также
*                           учитываются (Примечание - Если код неактивен,
*                           то в этом случае должно генерироваться
*                           предупреждение)
*      errorCheckOnly       - TRUE указывает ограниченную проверку (только
*                           на ошибки)
*                           FALSE указывает расширенную проверку (на
*                           ошибки и предупреждения)
*
*     Возвращаемое значение - структура, содержащая счетчик ошибок nErrors,
*                           счетчик предупреждений nWarnings, за которыми
*                           следует детальная информация
*
*Исключения:
*   UnknownVocabularyDomain         - словарный домен не распознан службой
*   UnknownApplicationContextCode   - прикладной контекст не распознан службой
*/
                ValidateCodeReturn validateTranslation(
                        in VocabularyDomainName       vocabularyDomain_name,
                        in types::CD                  codeToValidate,
                        in ApplicationContextCode     applicationContext_code,
                        in types::BL                  activeConceptsOnly,
                        in types::BL                  errorCheckOnly
                        )
                        raises (UnknownVocabularyDomain,
                              UnknownApplicationContextCode,
                              UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:translateCode */

/* translateCode - транслировать кодированный атрибут. Если указана целевая
 * система кодирования, то переданный код транслируется в эту систему. Если
 * она не указана, то код транслируется в форму, применимую в целевом
 * контексте
 *
*      vocabularyDomain_name       - словарный домен кода и трансляции
*      fromCode                    - транслируемый код
*      toCodeSystemId              - идентификатор целевой системы кодирования
*                                    (не обязателен)
*      toApplicationContext_code   - контекст, идентифицирующий целевую среду
*
*Возвращаемое значение - новый элемент типа CD, содержащий трансляцию
*
*Исключения:
*   UnknownVocabularyDomain - словарный домен не распознан службой
*   UnknownCodeSystem       - система кодирования в параметре fromCode или
*                             toCodeSystemId не распознана службой
*   UnknownApplicationContextCode  - прикладной контекст не распознан службой
*   UnableToTranslate  - трансляция не состоялась по указанной причине
*
*/
                types::CD translateCode(
                        in VocabularyDomainName       vocabularyDomain_name,
                        in types::CD                  fromCode,
```

```
                        in CodeSystemId                toCodeSystemId,
                        in ApplicationContextCode      toApplicationContext_code
                        )
                        raises (UnknownVocabularyDomain,
                                UnknownCodeSystem,
                                UnknownApplicationContextCode,
                                UnableToTranslate,
                                UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:fillInDetails */


/* fillInDetails - заполнить необязательные поля в значении типа CD,
 *                 включая имя системы кодирования, выпуск и изображаемое
 *                 имя кода
 *      codeToFillIn          - заполняемый код
 *      displayLanguage_code  - язык, используемый для имени системы
 *                              кодирования и обозначения кода
 *
 * Возвращаемое значение    - код с заполненными полями
 *
 * Исключения:
 * UnknownCodeSystem     - часть кода, означающая систему кодирования,
 *                         не распознана службой
 * UnknownConceptCode    - код понятия не действителен в заданной системе
 *                         кодирования либо код не действителен
 * UnknownLanguage       - код языка не распознан
 * NoApplicableDesignationFound - для заданного значения типа CD не найдено
 *                                обозначение на заданном языке
 *
 */
      types::CD fillInDetails(
                        in types::CD                  codeToFillIn,
                        in LanguageCode         displayLanguage_code
              )
                        raises (UnknownCodeSystem,
                              UnknownConceptCode,
                                UnknownLanguage,
                                UnexpectedError,
                                NoApplicableDesignationFound);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:subsumes */

/* subsumes            - проверить, является первый код категорией второго
 *      parentCode     - код родителя
 *      childCode      - код потомка
 *
 *      Возвращаемое значение  - TRUE, если первый код является родительским
 *                               для второго
 *                             - FALSE, если не является
 *
 * Примечания
 *   1 Трансляции игнорируются этим методом
 *   2 Если оба кода эквивалентны, метод возвращает значение TRUE
```

```
 *   3 Если один из кодов или оба кода являются исключениями из
 *     кодирования (CWE) и при этом не эквивалентны, то метод
 *     возвращает значение FALSE
 *
 * Исключения:
 * UnknownCodeSystem  - система кодирования родителя или потомка не распознана
 * UnknownConceptCode - код parentCode или childCode не распознан службой
 * SubsumptionNotSupported - проверка родительского отношения не
 *                           поддерживается службой или системой кодирования
 * UnrecognizedQualifier    - один из квалификаторов понятия не распознан
 *                           службой
 * QualifiersNotSupported   - квалифицированные понятия не поддерживаются
 *                           службой
 */
               types::BL subsumes(
                       in types::CD  parentCode,
                       in types::CD  childCode
                       )
                       raises (UnknownCodeSystem,
                             UnknownConceptCode,
                               SubsumptionNotSupported,
                               UnrecognizedQualifier,
                               QualifiersNotSupported,
                               UnexpectedError);


/* HL7SpecBlockEnd */
/* HL7SpecBlock:areEquivalent */


/* areEquivalent  - определить, являются ли два кода эквивалентными
 * code1          - первый код
 * code2          - второй код
 *
 * Возвращаемое значение   - TRUE, если определено, что коды эквивалентны,
 *                           FALSE в противном случае
 *
 * Исключения:
 * UnknownCodeSystem          - система кодирования кодов code1 или code2 не
 *                              распознана службой
 * UnknownConceptCode         - код code1 или code2 не распознан службой
 * SubsumptionNotSupported    - проверка родительского отношения не
 *                              поддерживается службой или системой кодирования
 * UnrecognizedQualifier      - один из квалификаторов понятия не распознан
 *                              службой
 * QualifiersNotSupported     - квалифицированные понятия не поддерживаются
 *                              службой
 */
    types::BL areEquivalent (
      in types::CD  code1,
      in types::CD  code2
      )
                       raises (UnknownCodeSystem,
                             UnknownConceptCode,
                               SubsumptionNotSupported,
                               UnrecognizedQualifier,
```

```
                                QualifiersNotSupported,
                                UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupValueSetExpansion */

/* ValueSetExpansionList - возвратить иерархический список значений,
 *                         допустимых для заданного словарного домена и
 *                         контекста
 *      vocabularyDomain_name   - раскрываемый словарный домен
 *      applicationContext_code - контекст, используемый для выборки
 *                                набора значений (не обязателен)
 *      language_code         - язык, используемый для визуализации списка
 *      expandAll           - TRUE указывает, что все узлы раскрываются
 *                            на полную глубину, FALSE - на один уровень
 * timeout     - время ожидания завершения операции в миллисекундах
 *               (0 - не ограничено)
 * sizeLimit   - максимальное число возвращаемых элементов
 *               (0 - не ограничено)
 *
 *      Возвращаемое значение   - список допустимых кодов
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 * Исключения
 *   UnknownVocabularyDomain         - домен не распознан службой
 *   UnknownApplicationContextCode   - прикладной контекст не распознан службой
 *   UnknownLanguage           - код языка не распознан службой
 *   NoApplicableValueSet      - не существует набора значений, относящегося
 *                             к этому домену в данном контексте

 *   TimeoutError        - превышен лимит времени
 */

              ValueSetExpansionList lookupValueSetExpansion(
                      in VocabularyDomainName       vocabularyDomain_name,
                      in ApplicationContextCode     applicationContext_code,
                      in LanguageCode               language_code,
                      in types::BL                  expandAll,
                      in long                 timeout,
                      in long                       sizeLimit
                      )
                      raises (UnknownVocabularyDomain,
                            UnknownApplicationContextCode,
                              UnknownLanguage,
                              NoApplicableValueSet,
                              TimeoutError,
                              UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:expandValueSetExpansionContext */

/* expandValueSetExpansionContext - раскрыть узел, который не до конца
```

```
 *                                раскрыт предыдущим вызовом описанного
 *                                выше метода lookupValueSetExpansion
 * expandContext          - начальный контекст, используемый для раскрытия
 * Возвращаемое значение  - список допустимых кодов для данного контекста
 *
 */
               ValueSetExpansionList expandValueSetExpansionContext(
                in ExpansionContext      expansionContext
                )
                raises (InvalidExpansionContext,
                  TimeoutError,
                  UnexpectedError);

/* HL7SpecBlockEnd */
       };



/**********************************
 *     API обозревателя           *
 **********************************/
 /* HL7SpecBlock:browser */

       interface Browser : Identification {

/* HL7SpecBlockEndEllipsis */
/* HL7SpecBlock:browserDescription */

/* getSupportedMatchAlgorithms - возвратить список поддерживаемых алгоритмов
 * совпадения
 */
   MatchAlgorithmCodeList  getSupportedMatchAlgorithms() raises
(UnexpectedError);

/* getSupportedAttributes - возвратить кодированные атрибуты, известные
 *                          обозревателю, чьи имена соответствуют заданным
 *                          критериям
 *
 *  matchText           - искомый образец. Формат зависит от алгоритма
 *                        совпадения. Пустая строка означает возвращение
 *                        всех имен
 *  matchAlgorithm_code - код алгоритма совпадения
 *  timeout             - время ожидания завершения операции в миллисекундах
 *                        (0 - не ограничено)
 *  sizeLimit           - максимальное число возвращаемых элементов
 *                        (0 - не ограничено)
 *  Возвращаемое значение     - список кодированных атрибутов
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 * Исключения:
 *  TimeoutError         - превышен лимит времени
```

```
 *  BadlyFormedMatchText - искомый образец не может быть разобран
 *                         указанным алгоритмом
 *  UnknownMatchAlgorithm - алгоритм совпадения не распознан службой
 */
        RIMCodedAttributeList     getSupportedAttributes(
          in types::ST      matchText,
      in MatchAlgorithmCode  matchAlgorithm_code,
          in long     timeout,
           in long    sizeLimit
           )
           raises (BadlyFormedMatchText,
                   UnknownMatchAlgorithm,
                   TimeoutError,
                   UnexpectedError);


/* getSupportedVocabularyDomains - возвратить словарные домены, известные
 *                                 обозревателю, чьи имена соответствуют
 *                                 заданным критериям
 *
 *  matchText           - искомый образец. Формат зависит от алгоритма
 *                        совпадения. Пустая строка означает возвращение
 *                        всех имен
 *  matchAlgorithm_code - код алгоритма совпадения
 *  timeout             - время ожидания завершения операции в миллисекундах
 *                        (0 - не ограничено)
 *  sizeLimit           - максимальное число возвращаемых элементов
 *                        (0 - не ограничено)
 *
 *  Возвращаемое значение   - список словарных доменов
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 * Исключения:
 *  TimeoutError          - превышен лимит времени
 *  BadlyFormedMatchText  - искомый образец не может быть разобран
 *                          указанным алгоритмом
 *  UnknownMatchAlgorithm - алгоритм совпадения не распознан службой
 */
    VocabularyDomainNameList getSupportedVocabularyDomains(
      in types::ST     matchText,
      in MatchAlgorithmCode  matchAlgorithm_code,
          in long     timeout,
           in long    sizeLimit
           )
           raises (BadlyFormedMatchText,
                   UnknownMatchAlgorithm,
                   TimeoutError,
                   UnexpectedError);

/* getSupportedValueSets - возвратить список наборов значений, известных
 *                         обозревателю, чьи имена совпадают с переданным
 *                         искомым образцом
```

```
*
*  matchText            - искомый образец. Формат зависит от алгоритма
*                         совпадения. Пустая строка означает возвращение
*                         всех имен
*  matchAlgorithm_code - код алгоритма совладения
*  timeout              - время ожидания завершения операции в миллисекундах
*                         (0 - не ограничено)
*  sizeLimit            - максимальное число возвращаемых элементов
*                         (0 - не ограничено)
*
* Возвращаемое значение       - список наборов значений
*
* Примечание - Если указан параметр sizeLimit и число записей в возвращенном
* списке совпадает с его значением, то клиент должен предполагать, что
* список не полон
*
* Исключения:
*  TimeoutError           - превышен лимит времени
*  BadlyFormedMatchText   - искомый образец не может быть разобран
*                           указанным алгоритмом
*  UnknownMatchAlgorithm - алгоритм совпадения не распознан службой
*/

    ValueSetDescriptorList   getSupportedValueSets(
      in types::ST     matchText,
      in MatchAlgorithmCode  matchAlgorithm_code,
           in long     timeout,
            in long    sizeLimit
            )
            raises (BadlyFormedMatchText,
              UnknownMatchAlgorithm,
              TimeoutError,
                    UnexpectedError);

/* getSupportedCodeSystems - возвратить список систем кодирования, известных
*                           обозревателю, чьи имена совпадают с заданными
*                           критериями
*
*
*  matchText             - искомый образец. Формат зависит от алгоритма
*                          совпадения. Пустая строка означает возвращение
*                          всех имен
*  matchAlgorithm_code   - код алгоритма совпадения
*  timeout               - время ожидания завершения операции в миллисекундах
*                          (0 - не ограничено)
*  sizeLimit             - максимальное число возвращаемых элементов
*                          (0 - не ограничено)
*
*      Возвращаемое значение      - список систем кодирования
*
* Примечание - Если указан параметр sizeLimit и число записей в возвращенном
*              списке совпадает с его значением, то клиент должен
*              предполагать, что список не полон
*
```

```
 * Исключения
 *  TimeoutError          - превышен лимит времени
 *  BadlyFormedMatchText  - искомый образец не может быть разобран
 *                          заданным алгоритмом
 *  UnknownMatchAlgorithm - алгоритм совпадения не распознан службой
 */
    CodeSystemDescriptorList getSupportedCodeSystems(
      in types::ST     matchText,
      in MatchAlgorithmCode  matchAlgorithm_code,
           in long     timeout,
            in long    sizeLimit
            )
            raises (BadlyFormedMatchText,
             UnknownMatchAlgorithm,
             TimeoutError,
                   UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupVocabularyDomain */

/* lookupVocabularyDomain - просмотреть полное описание словарного домена
 *  domain     - идентификатор просматриваемого словарного домена
 *
 *  Возвращаемое значение - полное описание словарного домена
 */
    VocabularyDomainDescription lookupVocabularyDomain(
      in VocabularyDomainName vocabularyDomain_name
      )
      raises (UnknownVocabularyDomain,
        UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupValueSet */

/* lookupValueSet - возвратить полное описание набора значений
 *  valueSet      - уникальный идентификатор или имя набора значений
 *
 * Возвращаемое значение - полное описание набора значений
 *
 * Исключения:
 *  UnknownValueSet    - идентификатор набора значений задан, но не распознан
 *                       службой, либо задано только имя набора значений, но
 *                       не распознано службой, либо не заданы ни
 *                       идентификатор, ни имя набора значений
 *  ValueSetNameIdMismatch  - заданы и идентификатор набора значений, и его
 *                            имя, но они не соответствуют друг другу
 */
    FullValueSetDescription lookupValueSet(
           in ValueSetId           valueSet_id,
           in ValueSetName         valueSet_name
                )
             raises (UnknownValueSet,
               ValueSetNameIdMismatch,
               UnexpectedError);

/* HL7SpecBlockEnd */
```

```
/* HL7SpecBlock:lookupCodeSystem */

/* lookupCodeSystem - возвратить описание и имя системы кодирования
 *  codeSystem     - уникальный идентификатор системы кодирования
 *
 * Возвращаемое значение - описание системы кодирования
 *
 * Исключения:
 *  UnknownCodeSystem      - идентификатор системы кодирования задан, но не
 *                           распознан службой, либо задано только имя системы
 *                           кодирования и не распознано службой, либо не
 *                           заданы ни идентификатор, ни имя системы
 *                           кодирования
 *  CodeSystemNameIdMismatch - заданы и идентификатор системы кодирования, и
 *                             ее имя, но они не соответствуют друг другу

 */
    CodeSystemInfo lookupCodeSystem (
      in CodeSystemId        codeSystem_id,
      in CodeSystemName      codeSystem_name
      )
      raises (UnknownCodeSystem,
        CodeSystemNameIdMismatch,
        UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupValueSetForDomain */

/* lookupValueSetForDomain - возвратить описатель набора значений, который
 *                           может использоваться для словарного домена в
 *                           заданном контексте (если такой набор имеется)
 *
 *  vocabularyDomain_name    - имя словарного домена
 *  applicationContext_code  - код прикладного контекста (не обязателен)
 *
 *  Возвращаемое значение    - идентификатор и имя (если оно имеется)
 *                             набора значений, который можно использовать
 *                             в данном контексте
 *
 *Исключения:
 *  UnknownVocabularyDomain       - словарный домен не распознан службой
 *  UnknownApplicationContextCode - прикладной контекст не распознан
 *                                  службой
 *  NoApplicableValueSet          - не существует набора значений,
 *                                  относящегося к этому домену в данном
 *                                  контексте
 *
 */
                ValueSetDescriptor lookupValueSetForDomain(
                      in VocabularyDomainName       vocabularyDomain_name,
      in ApplicationContextCode    applicationContext_code
                      )
                      raises (UnknownVocabularyDomain,
                            UnknownApplicationContextCode,
                            NoApplicableValueSet,
                            UnexpectedError);
```

```
/* HL7SpecBlockEnd */
/* HL7SpecBlock:isCodeInValueSet */


/* isCodeInValueSet - определить, входит ли данный код понятия в набор
 *                    значений
 *  valueSet_id      - идентификатор набора значений (не обязателен, но или
 *                     идентификатор, или имя набора должны быть указаны)
 *  valueSet_name    - имя набора значений (не обязательное)
 *  includeHeadCode  - TRUE указывает, что "головной код" набора
 *                     значений, если таковой имеется, должен считаться
 *                     частью набора
 *  codeToValidate   - система кодирования и проверяемый код понятия
 *
 *  Возвращаемое значение  - TRUE, если понятие входит в набор значений
 *                           FALSE, если не входит
 *
 * Исключения:
 *  UnknownValueSet        - идентификатор набора значений (если указан) не
 *                           распознан службой или, если идентификатор
 *                           отсутствует, имя набора не распознано службой
 *  ValueSetNameIdMismatch - идентификатор набора значений распознан
 *                           службой, но имя набора не соответствует этому
 *                           идентификатору
 *  UnknownCodeSystem      - система кодирования не распознана службой
 *  UnknownConceptCode     - код понятия не действителен в системе
 *                           кодирования
 *
 */
                types::BL isCodeInValueSet(
                       in ValueSetId          valueSet_id,
                       in ValueSetName        valueSet_name,
                       in types::BL           includeHeadCode,
                       in ConceptId           codeToValidate
                       )
                       raises (UnknownValueSet,
                             ValueSetNameIdMismatch,
                             UnknownConceptCode,
                             UnknownCodeSystem,
                             UnexpectedError);

/* HL7SpecBlockEnd */
        };

};

#endif
```

### 16.3 Определение API словаря ОТС на языке IDL

Ниже приведено определение API словаря ОТС на языке IDL (файл CTSVAPI.idl).

```
/*
 * Исходный файл: CTSVAPI.idl
 * Версия:  1.0
 * Дата:    10/31/03
```

```
 */
#ifndef   __CTSVAPI_DEFINED
#define   __CTSVAPI_DEFINED


/*
 *  API уровня словаря Централизованных терминологических служб
 *  Этот модуль определяет интерфейс между приложением, обрабатывающим
 *  сообщения, соответствующие стандарту HL7, и терминологическими службами
 */
module CTSVAPI {

/*********************************************
 * Соглашения о суффиксах:
 *Xxx     - суффикс типа
 *_xxx    - суффикс атрибута
 *
 * Суффиксы
 *Id      - уникальный идентификатор сущности
 *Name    - уникальное имя
 *Code    - код понятия
 *Version   - идентификатор версии
 */

/*********************************************
 *          Основные типы данных             *
 *********************************************/
/* HL7SpecBlock:basicData */

/* Объектный идентификатор ИСО (ОИД) */
        typedef string  OID;

/* Описание или определение сущности. Может представлять собой текст большого
 * объема, несколько строк и т. д.
 */
        typedef string  Description;

/*********************************************
 *   Идентификаторы основных сущностей       *
 *********************************************/

/* Идентификатор версии спецификации ОТС */
        struct CTSVersionId {
          short  major;
          short  minor;
        };

/* Уникальный идентификатор системы кодирования */
        typedef OID     CodeSystemId;
        typedef sequence<CodeSystemId> CodeSystemIdList;

/* Имя системы кодирования */
        typedef string  CodeSystemName;

/* Код понятия в системе кодирования */
        typedef string  ConceptCode;
```

```
        typedef sequence<ConceptCode> ConceptCodeList;

/* Уникальный идентификатор понятия, состоящий из идентификатора системы
 * кодирования и кода понятия
 */
        struct ConceptId {
                CodeSystemId codeSystem_id;
                ConceptCode  concept_code;
        };
        typedef sequence<ConceptId> ConceptIdList;

/* Идентификатор версии */
        typedef string  VersionId;

/* Версия системы кодирования */
        typedef VersionId       CodeSystemVersion;
        typedef sequence<VersionId> CodeSystemVersionList;

/* Непрозрачный контекст, передаваемый службой клиенту */
  typedef sequence<octet> ExpansionContext;

/* HL7SpecBlockEnd */
/*********************************************
 *      Кодированные элементы                *
 *********************************************/
 /* HL7SpecBlock:conceptCodes */
/* Идентификатор языка
 * Синтаксис основан на спецификации IETF RFC 3066 - Теги идентификации
 * языков. Состоит из основного тега primary-subtag, за которым следуют нуль
 * или более вторичных субтегов <secondary subtag>
 * <primary subtag>  - по возможности берется из ИСО 639 часть 1
 *                     (2 символа),  иначе берется трехсимвольный код
 *                   - коды "i-" и "x-" не разрешены в данной спецификации
 *
 * <secondary subtag 1> - 2-8 символов
 *             - коды из 2 символов берутся из стандарта кодов стран ИСО 3166
 *               коды из 3-8 символов берутся из регистра IANA
 *             - см. http://www.iana.org/assignments/language-tags
 * <secondary subtag 2-n> - правила не определены, хотя некоторые теги
 * зарегистрированы в IANA
 *
 * Системы кодирования:  ISO639-1          (2.16.840.1.113883.6.99)
 *      ISO639-2    (2.16.840.1.113883.6.100)
 *      ISO 3166-1     (2.16.1)
 */
        typedef ConceptCode     LanguageCode;
        typedef sequence<LanguageCode>  LanguageCodeList;


/* MimeTypeCode identifier - идентификатор типа среды Mime.
 * Система кодирования:  MediaType (2.16.840.1.113883.5.79)
 */
        typedef ConceptCode     MimeTypeCode;
        typedef sequence<MimeTypeCode> MimeTypeCodeList;

/* ConceptStatusCode - указывает, является ли код понятия активным,
```

```
 *                    устаревшим и т. д.
 * Система кодирования: ConceptStatusCode (2.16.840.1.113883.5.1086)
 */
  typedef ConceptCode   ConceptStatusCode;


/* PropertyCode - идентификатор свойства в системе кодирования.
 *      Включает в себя и идентификатор системы кодирования, и код понятия.
 *      Система кодирования: ConceptProperty (2.16.840.1.113883.5.1087)
 */
       typedef ConceptCode       PropertyCode;
       typedef sequence<PropertyCode> PropertyCodeList;


/* RelationshipCode - идентификатор отношения в системе кодирования.
 *                    Содержит только код понятия.
 *                    Отношения, определенные в стандарте HL7, должны браться
 *                    из системы кодирования ConceptCodeRelationship
 *                    (2.16.840.1.113883.5.1088)
 */
       typedef ConceptCode       RelationshipCode;
       typedef sequence<RelationshipCode> RelationshipCodeList;

/* MapQualityCode - код понятия, идентифицирующий общее качество отображения
 *                  (например шире, уже,...)
 *                  По возможности должен быть взят из системы кодирования
 *                  (2.16.840.1.113883.5.1093)
 */
       typedef ConceptCode     MapQualityCode;

/*
 * RelationQualifierCode - квалификатор, уточняющий или разъясняющий отношение
 *                         между двумя понятиями. В настоящее время комитетом
 *                         HL7 не используется
 */
  typedef ConceptCode   RelationQualifierCode;
  typedef sequence<RelationQualifierCode> RelationQualifierCodeList;

 /*
  * MatchAlgorithmCode - код, идентифицирующий алгоритм совпадения,
  *                      используемый при поиске. По возможности должен быть
  *                      взят из системы кодирования алгоритмов совпадения
  *                      (2.16.840.1.113883.5.1094)
  */
  typedef ConceptCode   MatchAlgorithmCode;
  typedef sequence<MatchAlgorithmCode> MatchAlgorithmCodeList;

/* HL7SpecBlockEnd */

/*************************************
 * Сущности, представляющие UML-модель  *
 *************************************/
 /* HL7SpecBlock:codeSystemIdAndVersions */
/* CodeSystemIdAndVersions - идентификатор, имя и версия системы кодирования
 *  codeSystem_id          - идентификатор системы кодирования (ОИД)
 *  codeSystem_name        - имя системы кодирования
```

```
 *  copyright              - сведения об авторских правах, если таковые имеются
 *  codeSystem_VersionList - версия выпуска системы кодирования (если
 *                           имеется)
 */
       struct CodeSystemIdAndVersions {
              CodeSystemId                    codeSystem_id;
              CodeSystemName                  codeSystem_name;
              string                    copyright;
              CodeSystemVersionList           codeSystem_versions;
       };
       typedef sequence<CodeSystemIdAndVersions> CodeSystemIdAndVersionsList;

/* HL7SpecBlockEnd */
/* HL7SpecBlock:codeSystemInfo */

/* CodeSystemInfo - детальное описание системы кодирования
 *  codeSystem            - идентификатор, имя и версия (версии) системы
 *                          кодирования
 *  fullName              - официальное наименование системы кодирования
 *  codeSystemDescription - текстовое описание системы кодирования
 *  supportedLanguages    - список языков, поддерживаемых системой
 *                          кодирования
 *  supportedRelations    - список отношений, поддерживаемых системой
 *                          кодирования
 *  supportedProperties   - список свойств, поддерживаемых системой
 *                          кодирования
 *  representsVersion     - версия системы кодирования, если применима
 *  supportedMimeTypes    - список типов среды MIME, поддерживаемых системой
 *                          кодирования
 *  supportedRelationQualifiers - квалификаторы отношений, поддерживаемых
 *                                системой кодирования
 */
       struct  CodeSystemInfo {
              CodeSystemIdAndVersions         codeSystem;
              string                          fullName;
              Description                     codeSystemDescription;
              LanguageCodeList                supportedLanguages;
              RelationshipCodeList            supportedRelations;
              PropertyCodeList                supportedProperties;
              MimeTypeCodeList          supportedMimeTypes;
              RelationQualifierCodeList  supportedRelationQualifiers;
       };
/* HL7SpecBlockEnd */
/* HL7SpecBlock:conceptDesignation */

/* ConceptDesignation     - имя или представление, данное понятию,
 *                          представленному кодом
 *  designation           - имя или обозначение понятия
 *  language              - язык обозначения
 *  preferredForLanguage  - TRUE указывает, что обозначение предпочтительно
 *                          для языка, указанного в поле ED.language
 *  contextsOfUse         - список контекстов,в которых применяется
 *                          обозначение
 */
       struct ConceptDesignation {
              string                          designation;
```

```
                LanguageCode                    language_code;
                boolean                         preferredForLanguage;
        };
        typedef sequence<ConceptDesignation> ConceptDesignationList;
/* HL7SpecBlockEnd */
/* HL7SpecBlock:conceptProperty */

/* ConceptProperty - характеристика или атрибут понятия, представленного
 *                   кодом
 *  property_code     - пара система кодирования/код свойства,
 *                      идентифицирующая свойство
 *  propertyValue     - значение свойства
 *  language_code     - язык свойства (не обязателен)
 *  mimeType_code     - тип среды MIME свойства (по умолчанию text/plain)
 */
        struct ConceptProperty {
                PropertyCode                    property_code;
                string                          propertyValue;
                LanguageCode                    language_code;
                MimeTypeCode                    mimeType_code;
        };
        typedef sequence<ConceptProperty> ConceptPropertyList;

/* HL7SpecBlockEnd */
/* HL7SpecBlock:conceptRelationship */

/* ConceptRelationship - определение отношения */
  struct ConceptRelationship {
    ConceptId     sourceConcept_id;
    RelationshipCode   relationship_code;
    RelationQualifierCodeList relationQualifiers;
    ConceptId     targetConcept_id;
  };
  typedef sequence<ConceptRelationship> ConceptRelationshipList;
/* HL7SpecBlockEnd */
/* HL7SpecBlock:completeCodedConceptDescription */

/* CompleteCodedConceptDescription - вся известная информация о коде понятия
 *  concept_id          - идентификатор системы кодирования и код понятия
 *  conceptStatus_code  - статус понятия в версии системы кодирования
 *  codeSystem_version  - версия системы кодирования, из которой взята
 *                        информация
 *  designatedBy        - обозначения понятия
 *  hasProperties       - свойства понятия
 *  sourceFor           - коды квалифицированных отношений и коды
 *                        непосредственных целевых понятий
 *  targetOf            - коды квалифицированных отношений и коды
 *                        непосредственных понятий-источников
 */
        struct CompleteCodedConceptDescription {
                ConceptId               concept_id;
                ConceptStatusCode         conceptStatus_code;
                CodeSystemVersion   codeSystem_version;
                ConceptDesignationList  designatedBy;
                ConceptPropertyList     hasProperties;
                ConceptRelationshipList sourceFor;
```

```
                ConceptRelationshipList targetOf;
        };
/* HL7SpecBlockEnd */

/* HL7SpecBlock:relatedCode */
/* RelatedCode - информация о связанном коде
 *  pathLength       - расстояние от стартового узла
 *  concept_code     - код понятия
 *  designation      - подходящее обозначение понятия
 *  canExpand        - TRUE указывает, что возможно дальнейшее раскрытие
 *  expansionContext - контекст (непрозрачный), используемый для дальнейшего
 *                     раскрытия
 */
        struct RelatedCode {
                short               pathLength;
                ConceptCode         concept_code;
                string              designation;
                RelationQualifierCodeList relationQualifiers;
                boolean             canExpand;
                ExpansionContext    expansionContext;
        };
        typedef sequence<RelatedCode> RelatedCodeList;
/* HL7SpecBlockEnd */
/* HL7SpecBlock:supportedMap */

/* CodeMap - отображение кода
 *  map_name        - уникальный идентификатор конкретного отображения
 *  fromCodeSystem_id       - система кодирования-источник отображения
 *  fromCodeSystem_name     - имя системы кодирования-источника
 *  fromCodeSystem_version  - версия системы кодирования-источника (не
 *                            обязательна)
 *  toCodeSystem_id         - идентификатор целевой системы кодирования
 *                            отображения
 *  toCodeSystem_name       - имя целевой системы кодирования
 *  toCodeSystem_version    - версия целевой системы кодирования (не обязательна)
 *  mapDescription          - описание отображения (источник, версия, дата,
 *                            местонахождение и т. д.)
 *
 * Примечание - Отображения не симметричны
 */
        struct CodeMap {
          string        map_name;
          Description                   mapDescription;
                CodeSystemId                    fromCodeSystem_id;
                CodeSystemName                  fromCodeSystem_name;
                CodeSystemVersion               fromCodeSystem_version;
                CodeSystemId                    toCodeSystem_id;
                CodeSystemName                  toCodeSystem_name;
                CodeSystemVersion               toCodeSystem_version;

        };
        typedef sequence<CodeMap> CodeMapList;
/* HL7SpecBlockEnd */
/* HL7SpecBlock:mappedConceptCode */
```

```
/* MappedConceptCode - результат отображения кода понятия и признак качества
 *                     отображения
 *  mappedConcept_id     - отображаемый код и его система кодирования
 *  mapQuality_code      - грубая мера точности отображения
 *
 */
        struct MappedConceptCode {
                ConceptId               mappedConcept_id;
                MapQualityCode          mapQuality_code;
        };

/* HL7SpecBlockEnd */
/* HL7SpecBlock:stringAndLanguage */
/* StringAndLanguage - пара, образованная текстовой строкой и ассоциированным
 *                     кодом языка
*/
  struct StringAndLanguage {
    string    text;
    LanguageCode  language_code;
  };
/* HL7SpecBlockEnd */

/*********************************************
 *      Исключения                           *
 *********************************************/
 /* HL7SpecBlock:exceptions */
/*
 * Не специфичная ошибка, воспрепятствовавшая успешному завершению вызова
 */
  exception UnexpectedError {
    string possible_cause;
  };

/*
 * Передан идентификатор системы кодирования, не распознанный службой
 */
        exception UnknownCodeSystem {
                CodeSystemId            codeSystem_id;
        };


/*
 * Передан код понятия, не действительный в заданной системе кодирования
 */
        exception UnknownConceptCode {
                ConceptCode               concept_code;
        };



/*
 * Использован код свойства, не действительный для системы кодирования
 */
        exception UnknownPropertyCode {
                PropertyCode            property_code;
        };
```

```
/*
 * Передан код языка, не действительный для системы кодирования
 */
  exception UnknownLanguageCode {
    LanguageCode     language_code;
  };

/*
 * Передан код отношения, не действительный для системы кодирования
 */
  exception UnknownRelationshipCode {
    RelationshipCode relationship_code;
  };

/*
 * Передан код квалификатора отношения, не действительный для системы
 * кодирования
 */
  exception UnknownRelationQualifier {
    RelationQualifierCode      relationQualifier_code;
  };

/*
 * Передан не распознанный код типа среды MIME
 */
  exception UnknownMimeTypeCode {
    MimeTypeCode     mimeType_code;
  };

/*
 * Контекст, переданный в параметре expandCodeContext, не действителен или
 * каким-то образом поврежден
 */
  exception InvalidExpansionContext {
  };

/*
 * Для переданного кода понятия и критериев выборки не найдено подходящее
 * обозначение
 */
  exception NoApplicableDesignationFound {
    ConceptId concept_id;
    LanguageCode  language_code;
  };


/*
 * Идентификатор системы кодирования не соответствует ее имени
 */
  exception CodeSystemNameIdMismatch {
    CodeSystemId  codeSystem_id;
    CodeSystemName codeSystem_name;
  };

/*
 * Формат текста для поиска совпадения не может быть разобран
```

```
 */
  exception BadlyFormedMatchText {
    string matchText;
  };

/*
 * Время выполнения операции превысило заданный лимит времени
 */
  exception TimeoutError {
  };

 /*
  * Переданный код алгоритма совпадения не поддерживается службой
  */
  exception UnknownMatchAlgorithm {
    MatchAlgorithmCode matchAlgorithm_code;
  };

/* HL7SpecBlockEnd */
/* HL7SpecBlock:mapExceptions */

/*
 * Отсутствует доступное отображение между двумя системами кодирования
 */
       exception MappingNotAvailable  {
              CodeSystemId            fromCodeSystem_id;
              CodeSystemId            toCodeSystem_id;
       };

/*
 * Служба не может создать отображение между понятием-источником и целевым
 * понятием
 */
  exception UnableToMap {
  };


 /*
  * Служба отображения не распознала переданное имя отображения
  */
  exception UnknownMapName {
    string      map_name;
  };

  /*
   * Имя отображения не было передано и существует несколько отображений
   * понятия системы кодирования-источника в целевую систему кодирования
   * Возвращаемые сведения содержат список возможных отображений
   */
    exception AmbiguousMapRequest {
      sequence<string> possible_maps;
    };

 /*
  * Передано имя отображения, но идентификатор системы кодирования-источника,
  * содержащийся в параметре fromConcept_id, не совпадает с тем, что указан
```

```
   * в отображении
   */
   exception MapNameSourceMismatch {
     CodeSystemId     fromCodeSystem_id;
     CodeSystemId     mapSourceCodeSystem_id;
   };

  /*
   * Передано имя отображения и идентификатор целевой системы кодирования
   * toCodeSystem_id не совпадает с тем, что указан в отображении
   */
   exception MapNameTargetMismatch {
     CodeSystemId     toCodeSystem_id;
     CodeSystemId     mapTargetCodeSystem_id;
   };

/* HL7SpecBlockEnd */


/**********************************
 *  Раздел модуля идентификации   *
 **********************************/
  /* HL7SpecBlock:identification */
  interface Identification {
/* Имя реализованной службы */
         string    getServiceName() raises (UnexpectedError);

/* Версия реализованной службы */
         string    getServiceVersion() raises (UnexpectedError);

/* Описание службы (источник, дата и т. д.) */
         string    getServiceDescription() raises (UnexpectedError);

/* Версия ОТС, представленная данной службой */
         CTSVersionId  getCTSVersion() raises (UnexpectedError);
        };
/* HL7SpecBlockEnd */

/* HL7SpecBlock:mapping */
/***********************************************
 *      Интерфейс отображения кодов             *
 *                                              *
 * Интерфейс отображения кодов представляет одно *
 * или несколько отображений систем кодирования  *
 ***********************************************/
        interface CodeMapping : Identification {
/* HL7SpecBlockEndElipsis */
/* HL7SpecBlock:supportedMaps */

/* getSupportedMaps - возвратить список отображений, поддерживаемых службой
 */
               CodeMapList getSupportedMaps() raises (UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:mapConceptCode */
```

```
/* mapConceptCode - отобразить код понятия из системы кодирования-источника
 * на ближайший эквивалент (если таковой имеется) в целевой системе
 * кодирования
 *  fromConcept_id    - система кодирования / отображаемый код понятия
 *  toCodeSystem_id   - целевая система кодирования
 *  map_name          - имя используемого отображения. Может быть опущено,
 *                      если существует лишь одно отображение системы
 *                      кодирования, указанной в параметре fromConcept_id,
 *                      на систему кодирования toCodeSystem_id
 *
 *    Возвращаемое значение    - результирующее понятие из целевой системы
 *
 *    Исключения:
 *  UnknownCodeSystem    - система кодирования-источик или целевая система
 *                         кодирования не поддерживается этой службой
 *                         отображения
 *  UnknownConceptCode   - отображаемый код понятия не принадлежит системе
 *                         кодирования
 *  MappingNotAvailable  - не существует отображения переданного кода
 *                         понятия на целевую систему кодирования
 *
 *  UnableToMap          - требуемый код понятия не может быть отображен
 *  UnknownMapName       - значение параметра mapping_name не распознано
 *                         службой
 *  MapSourceMismatch    - идентификатор системы кодирования-источника,
 *                         указанный в отображении, не совпадает с
 *                         идентификатором системы кодирования,
 *                         указанным в параметре fromConcept_id
 *  MapTargetMismatch    - идентификатор целевой системы кодирования, указанный
 *                         в отображении, не совпадает с идентификатором
 *                         системы кодирования, указанным в параметре
 *                         targetCodeSystem_id
 *  AmbiguousMapRequest  - существует более одного возможного отображения
 *                         понятия-источника в целевую систему кодирования
 *  UnexpectedError       - не специфичная ошибка, воспрепятствовавшая
 *                         успешному завершению вызова
*/
                MappedConceptCode mapConceptCode(
                        in ConceptId                    fromConcept_id,
                        in CodeSystemId                 toCodeSystem_id,
                        in string               map_name
                        )
                        raises ( UnknownCodeSystem,
                                 UnknownConceptCode,
                                 MappingNotAvailable,
                                 UnknownMapName,
                                 AmbiguousMapRequest,
                                 MapNameSourceMismatch,
                                 MapNameTargetMismatch,
                                 UnableToMap,
                                 UnexpectedError);
/* HL7SpecBlockEnd */
        };
```

```
/*************************************************************
 *     Службы времени исполнения
 *    Службы времени исполнения предназначены
 *    для среды времени исполнения (производственной среды)
 *************************************************************/
 /* HL7SpecBlock:runtime */
       interface Runtime : Identification {
 /* HL7SpecBlockEndElipsis */
 /* HL7SpecBlock:getSupportedCodeSystems */

/* getSupportedCodeSystems - возвратить список систем кодирования и их
 * версий, поддерживаемых данной службой *
 *  timeout              - время ожидания завершения операции в миллисекундах
 *                         (0 - не ограничено)
 *  sizeLimit            - максимальное число возвращаемых элементов
 *                         (0 - не ограничено)
 *
 * Возвращаемое значение - список поддерживаемых систем кодирования
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 * Исключения:
 *   TimeoutError  - превышен лимит времени
 */
        CodeSystemIdAndVersionsList     getSupportedCodeSystems(
           in long           timeout,
           in long           sizeLimit

           raises ( TimeoutError,
              UnexpectedError);
/* HL7SpecBlockEnd */
 /* HL7SpecBlock:lookupCodeSystemInfo */

/* lookupCodeSystemInfo - возвратить детальное описание системы кодирования
 *  codeSystem_id   - ОИД запрашиваемой системы кодирования
 *  codeSystem_name - имя запрашиваемой системы кодирования
 *
 *  Идентификатор codeSystem_id либо имя codeSystem_name должны
 *  присутствовать. Если имя присутствует, оно должно соответствовать
 *  идентификатору, иначе генерируется исключение
 *
 *  Возвращаемые данные    - детальное описание системы кодирования
 *
 * Исключения:
 *   UnknownCodeSystem - если система кодирования неизвестна службе
 *   CodeSystemNameIdMismatch - имя системы кодирования не соответствует
 *                              ее идентификатору
 */
    CodeSystemInfo lookupCodeSystemInfo(
      in CodeSystemId        codeSystem_id,
      in CodeSystemName      codeSystem_name
      )
      raises (UnknownCodeSystem,
```

```
        CodeSystemNameIdMismatch,
        UnexpectedError);
/* HL7SpecBlockEnd */
/* HL7SpecBlock:isConceptIdValid */

/* isConceptIdValid - определить, является ли данный код понятия
 * действительным в системе кодирования
 *  concept_id          - идентификатор системы кодирования / проверяемый код
 *  activeConceptsOnly  - TRUE указывает, что понятие должно иметь статус
 *                        активного, FALSE указывает, что оно может иметь
 *                        любой статус
 *
 *  Возвращаемое значение - TRUE, если код существует в текущей системе
 *                          кодирования и соответствует критерию статуса
 *
 * Исключения:
 *   UnknownCodeSystem - система кодирования не поддерживается службой
 */
              boolean isConceptIdValid(
               in ConceptId       concept_id,
                      in boolean        activeConceptsOnly
                      )
                      raises (UnknownCodeSystem,
                            UnexpectedError);
/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupDesignation */

/* lookupDesignation - просмотреть предпочтительное обозначение и
 * ассоциированный язык заданного кода для заданного языка и контекста
 *
 *  code      - просматриваемый код
 *  language  - предпочтительный язык вывода
 *
 *  Возвращаемое значение  - предпочтительное обозначение на этом языке в этом
 *                           контексте
 *
 * Исключения:
 *  UnknownLanguageCode  - переданный код языка не действителен для системы
 *                         кодирования
 *  UnknownCodeSystem    - система кодирования не поддерживается службой
 *  UnknownConceptCode   - код понятия не действителен для системы
 *                         кодирования
 *  NoApplicableDesignationFound - обозначение, соответствующее требуемому
 *                                 языку, не найдено
 *
 */
              StringAndLanguage lookupDesignation(
                      in ConceptId            concept_id,
                      in LanguageCode         language_code
                      )
                      raises (UnknownLanguageCode,
                            UnknownCodeSystem,
                            UnknownConceptCode,
                            NoApplicableDesignationFound,
                            UnexpectedError);
```

```
/* HL7SpecBlockEnd */
/* HL7SpecBlock:areCodesRelated */

/* areCodesRelated       - определить, связаны ли два кода понятий
 *  codeSystem_id        - идентификатор системы кодирования родительского и
 *                         дочернего кода
 *  source_code          - источник отношения
 *  target_code          - цель отношения
 *  relationship_code    - запрашиваемое отношение
 *  relationship_qualifiers - квалификаторы отношения, если имеются
 *  directRelationsOnly  - TRUE указывает, что проверяются только
 *                         непосредственные отношения родительский - дочерний
 *                         код, FALSE указывает, что учитываются также
 *                         косвенные, транзитивные связи
*
 *  Возвращаемое значение - TRUE: система кодирования содержит прямое или
 *                          косвенное отношение между родительским и дочерним
 *                          кодом
 *                          FALSE: система кодирования не содержит такое
 *                          отношение
 *
 * Исключения:
 *   UnknownConceptCode - родительский или дочерний код не действителен для
 *                        системы кодирования
 *  UnknownCodeSystem  - система кодирования не поддерживается службой
 *  UnknownRelation    - код отношения не действителен для системы
 *                       кодирования
 *  UnknownRelationshipQualifier - квалификатор отношения не действителен для
 *                                 системы кодирования
 *
 */
              boolean areCodesRelated(
               in CodeSystemId          codeSystem_id,
                      in ConceptCode               source_code,
                      in ConceptCode               target_code,
                      in RelationshipCode          relationship_code,
                      in RelationQualifierCodeList relationQualifiers,
                      in boolean             directRelationsOnly
                      )
                      raises (UnknownConceptCode,
                            UnknownCodeSystem,
                              UnknownRelationshipCode,
                              UnknownRelationQualifier,
                              UnexpectedError);
/* HL7SpecBlockEnd */

       };

/* HL7SpecBlock:browser */

/*********************************************
 *      Интерфейс обозревателя               *
 *********************************************/
 interface Browser : Identification{

 /* HL7SpecBlockEndEllipsis */
```

```
 /* HL7SpecBlock:browserSupport */

/* getSupportedMatchAlgorithms - возвратить список поддерживаемых алгоритмов
 *                               совпадения
 */
    MatchAlgorithmCodeList              getSupportedMatchAlgorithms() raises
(UnexpectedError);

/* getSupportedCodeSystems - возвратить список систем кодирования и их
 *                           версий, поддерживаемых данной службой
 *
 *  timeout              - время ожидания завершения операции в миллисекундах
 *                         (0 - не ограничено)
 *  sizeLimit            - максимальное число возвращаемых элементов
 *                         (0 - не ограничено)
 *
 *  Возвращаемое значение - список систем кодирования
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 * Исключения:
 *  TimeoutError - превышен лимит времени
 */
        CodeSystemIdAndVersionsList     getSupportedCodeSystems(
          in long     timeout,
          in long     sizeLimit)
          raises (TimeoutError,
            UnexpectedError);
/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupConceptCodesByDesignation */

/* lookupConceptCodesByDesignation - возвратить коды понятий, обозначения
 *                                   которых совпадают с образцом
 *  codeSystem_id        - запрашиваемая система кодирования
 *  matchText            - искомый образец. Формат зависит от алгоритма
 *                         совпадения. Пустая строка означает возвращение
 *                         всех имен
 *  matchAlgorithm_code  - код алгоритма совпадения
 *  language_code        - фильтр языка. Если присутствует, то ведется поиск
 *                         обозначений / описаний только на этом языке
 *                         (по умолчанию поиск ведется на всех языках)
 *  activeConceptsOnly   - TRUE указывает, что рассматриваются только
 *                         активные коды
 *                         FALSE означает все статусы (По умолчанию TRUE)
 *  timeout              - время ожидания завершения операции в миллисекундах
 *                         (0 - не ограничено)
 *  sizeLimit            - максимальное число возвращаемых элементов
 *                         (0 - не ограничено)
 *
 * Возвращаемое значение - список кодов понятий, удовлетворяющий заданным
 *                         критериям поиска
 *
```

```
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 * Исключения:
 *  UnknownCodeSystem     - система кодирования не поддерживается службой
 *  BadlyFormedMatchText  - искомый образец не может быть разобран
 *  UnknownLanguageCode   - код языка не поддерживается службой
 *  TimeoutError          - превышен лимит времени
 *  UnknownMatchAlgorithm - алгоритм совпадения не поддерживается службой
 *
 */
              ConceptIdList lookupConceptCodesByDesignation (
                      in CodeSystemId              codeSystem_id,
                      in string                 matchText,
                      in MatchAlgorithmCode        matchAlgorithm_code,
                      in LanguageCode              language_code,
                      in boolean                activeConceptsOnly,
                      in long                      timeout,
                      in long                      sizeLimit
                      )
                      raises (UnknownCodeSystem,
                            BadlyFormedMatchText,
                            UnknownMatchAlgorithm,
                              UnknownLanguageCode,
                              TimeoutError,
                              UnexpectedError);
/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupConceptCodesByProperty */

/* lookupConceptCodesByProperty - возвратить коды понятий, чьи свойства
 *                                совпадают с заданным образцом
 *  codeSystem_id        - запрашиваемая система кодирования
 *  matchText            - искомый образец. Формат зависит от алгоритма
 *                         совпадения. Пустая строка означает возвращение
 *                         всех имен
 *  matchAlgorithm_code  - код алгоритма совпадения
 *  language_code        - фильтр языка. Если присутствует, то ведется поиск
 *                         обозначений / описаний только на этом языке
 *                         (по умолчанию поиск ведется на всех языках)
 *  activeConceptsOnly   - TRUE указывает, что рассматриваются только
 *                         активные коды. FALSE означает все статусы
 *                         (По умолчанию TRUE)
 *  properties           - список кодов свойств, участвующих в поиске
 *                         (По умолчанию все свойства)
 *  mimeTypes            - ищутся только свойства с данными типами среды
 *                         MIME (По умолчанию - все типы MIME)
 *  timeout              - время ожидания завершения операции в миллисекундах
 *                         (0 - не ограничено)
 *  sizeLimit            - максимальное число возвращаемых элементов
 *                         (0 - не ограничено)
 *
 * Возвращаемое значение - список кодов понятий, свойства которых совпадают
 *                         с заданными критериями поиска
 *
```

```
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 *
 * Исключения:
 *  UnknownCodeSystem      - система кодирования не поддерживается службой
 *  BadlyFormedMatchText   - синтаксис искомого образа не соответствует
 *                           алгоритму совпадения
 *  UnknownLanguageCode    - код языка не поддерживается службой
 *  UnknownPropertyCode    - код свойства не поддерживается службой
 *  UnknownMimeTypeCode    - код типа среды MIME не распознан службой
 *  TimeoutError           - превышен лимит времени
 *   UnknownMatchAlgorithm  - алгоритм совпадения не поддерживается службой
 *
 */
             ConceptIdList lookupConceptCodesByProperty (
      in CodeSystemId              codeSystem_id,
                        in string               matchText,
                        in MatchAlgorithmCode         matchAlgorithm_code,
                        in LanguageCode               language_code,
                        in boolean              activeConceptsOnly,
      in PropertyCodeList          properties,
                        in MimeTypeCodeList           mimeTypes,
                        in long                       timeout,
                        in long                       sizeLimit
                        )
                        raises (UnknownCodeSystem,
                                BadlyFormedMatchText,
                                UnknownMatchAlgorithm,
                                UnknownLanguageCode,
                                UnknownPropertyCode,
                                UnknownMimeTypeCode,
                                TimeoutError,
                                UnexpectedError);
/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupCompleteCodedConcept */

/* lookupDesignations - получить полное описание кода понятия
 *  concept_id     - идентификатор запрашиваемого понятия
 *
 * Возвращаемое значение - полное описание понятия
 *
 * Исключения:
 *  UnknownCodeSystem      - система кодирования не поддерживается службой
 *  UnknwonConceptCode     - код понятия не действителен в системе кодирования
 */
    CompleteCodedConceptDescription lookupCompleteCodedConcept (
      in ConceptId  concept_id

      raises (UnknownCodeSystem,
        UnknownConceptCode,
        UnexpectedError);
/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupDesignations */
```

```
/* lookupDesignations - возвратить обозначения конкретного понятия,
 *                     совпадающие с заданными критериями
 *  concept_id          - идентификатор системы кодирования и код понятия,
 *                        чьи обозначения требуется найти
 *  matchText           - искомый образец. Формат зависит от алгоритма
 *                        совпадения. Пустая строка означает возвращение
 *                        всех имен
 *  matchAlgorithm_code - код алгоритма совпадения
 *  language_code       - если указан, совпадения ищутся только в
 *                        обозначениях на этом языке
 *
 * Исключения:
 *  UnknownCodeSystem     - идентификатор системы кодирования, содержащийся в
 *                          параметре concept_id, не распознан
 *  UnknownConceptCode    - код понятия не распознан
 *  BadlyFormedMatchText  - синтаксис искомого образца не действителен для
 *                          алгоритма совпадения
 *  UnknownLanguageCode   - код языка не поддерживается службой
 *  UnknownMatchAlgorithm - алгоритм совпадения не поддерживается службой
 *
 *
 */
              ConceptDesignationList lookupDesignations (
                      in ConceptId           concept_id,
                      in string         matchText,
                      in MatchAlgorithmCode   matchAlgorithm_code,
                      in LanguageCode   language_code
                      )
                      raises (UnknownCodeSystem,
                              UnknownConceptCode,
                              BadlyFormedMatchText,
                              UnknownMatchAlgorithm,
                              UnknownLanguageCode,
                              UnexpectedError);
/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupProperties */

/* lookupProperties - возвратить свойства конкретного понятия, совпадающие с
 *                    заданными критериями
 *  concept_id          - идентификатор системы кодирования и код понятия,
 *                        чьи свойства требуется найти
 *  properties          - если указан, ограничивает поиск этим списком
 *                        свойств
 *  matchText           - искомый образец. Формат зависит от алгоритма
 *                        совпадения. Пустая строка означает возвращение
 *                        всех имен
 *  matchAlgorithm_code - код алгоритма совпадения
 *  language            - если указан, то возвращаются свойства только на
 *                        этом языке
 *  mimeType            - ограничивает поиск только свойствами, имеющими
 *                        этот тип среды MIME
 *
 * Исключения:
 *  UnknownCodeSystem     - система кодирования не поддерживается службой
 *  UnknownConceptCode    - код понятия не действителен в системе кодирования
 *  UnknownPropertyCode   - один из кодов свойств не распознан
```

```
 *  BadlyFormedMatchText  - синтаксис искомого образца не может быть разобран
 *  UnknownLanguageCode   - код языка не поддерживается службой
 *  UnknownMimeTypeCode   - один из кодов типа среды MIME не распознан
 *  UnknownMatchAlgorithm - алгоритм совпадения не поддерживается службой
 *
 */
              ConceptPropertyList lookupProperties (
                      in ConceptId            concept_id,
                      in PropertyCodeList     properties,
                      in string         matchText,
                      in MatchAlgorithmCode   matchAlgorithm_code,
                      in LanguageCode   language_code,
                      in MimeTypeCodeList     mimeTypes
                      )
                      raises (UnknownCodeSystem,
                              UnknownConceptCode,
                              BadlyFormedMatchText,
                              UnknownLanguageCode,
                              UnknownPropertyCode,
                              UnknownMatchAlgorithm,
                              UnknownMimeTypeCode,
                              UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:lookupCodeExpansion */

/* lookupCodeExpansion - возвратить список детей (или родителей) заданного
 *                       кода
 *  expandConcept_id     - система кодирования / стартовый код понятия
 *                         Если код понятия опущен и указана только система
 *                         кодирования, то возвратить все коды понятий,
 *                         "корневые" для данного отношения
 *  relationship         - раскрываемое отношение
 *  sourceToTarget       - TRUE указывает направление от источника отношения
 *                         к цели, FALSE - от цели к источнику
 *  directRelationsOnly  - TRUE указывает, что возвращаются только
 *                         непосредственно связанные узлы. FALSE указывает,
 *                         что возвращается полный транзитивный граф (если
 *                         отношение транзитивно)
 *  designationLanguage_code - язык, используемый при возвращении обозначений
 *  timeout              - время ожидания завершения операции в миллисекундах
 *                         (0 - не ограничено)
 *  sizeLimit            - максимальное число возвращаемых элементов
 *                         (0 - не ограничено)
*
 *      Возвращаемое значение      - список раскрытых кодов
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон
 *
 * Исключения:
 *  UnknownConceptCode - переданный код понятия не действителен для системы ко-
дирования
 *  UnknownCodeSystem   - система кодирования не поддерживается службой
```

```
 *  UnknownRelation      - код отношения не действителен для системы
 *                         кодирования
 *  UnknownLanguageCode - указанный язык не действителен для системы
 *                         кодирования
 *  TimeoutError         - превышен лимит времени
 */

               RelatedCodeList lookupCodeExpansion (
                       in ConceptId            expandConcept_id,
                       in RelationshipCode     relationship_code,
                       in boolean        sourceToTarget,
                       in boolean        directRelationsOnly,
                       in LanguageCode   designationLanguage_code,
                       in long                 timeout,
                       in long                 sizeLimit
                       )
                       raises (UnknownConceptCode,
                             UnknownCodeSystem,
                               UnknownRelationshipCode,
                               UnknownLanguageCode,
                               TimeoutError,
                               UnexpectedError);

/* HL7SpecBlockEnd */
/* HL7SpecBlock:expandCodeExpansionContext */

/* expandCodeExpansionContext - раскрыть далее результаты предшествующего
 *                              вызова метода lookupCodeExpansion
 *  contextToExpand     - контекст узла из предыдущего сеанса раскрытия кода
 *
 *      Возвращаемое значение       - список раскрытых кодов
 *
 * Примечание - Если указан параметр sizeLimit и число записей в возвращенном
 * списке совпадает с его значением, то клиент должен предполагать, что
 * список не полон.
 *
 * Исключения:
 *  InvalidExpansionContext - контекст раскрытия не действителен или его срок
 *                            действия истек
 *  TimeoutError            - превышен лимит времени
 */
               RelatedCodeList expandCodeExpansionContext(
                       in ExpansionContext     contextToExpand
                       )
                       raises (InvalidExpansionContext,
                             TimeoutError,
                             UnexpectedError);

  );
/* HL7SpecBlockEnd */

);
```

## 17 Описание API ОТС на языке WSDL

### 17.1 Описание API времени исполнения на уровне сообщений

Ниже приведено описание службы времени исполнения MessageRuntime на языке WSDL (файл MessageRuntime.wsdl):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2013 sp1 (http://www.altova.com) by n"iAMD (Ru-Board) -->
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn://hl7.org/CTSMAPI" xmlns:intf="urn://hl7.org/CTSMAPI"
xmlns:tns2="urn://cts.hl7.org/types" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="urn://hl7.org/
CTSMAPI">
  <wsdl:import namespace="urn://cts.hl7.org/types"
location="MessageRuntime.wsdl#schema2.xsd"/>
  <wsdl:import namespace="urn://hl7.org/CTSMAPI"
location="MessageRuntime.wsdl#schema1.xsd"/>
  <wsdl:types/>
  <wsdl:message name="UnknownMatchAlgorithm">
    <wsdl:part name="fault" element="impl:fault2"/>
  </wsdl:message>
  <wsdl:message name="NoApplicableDesignationFound">
    <wsdl:part name="fault" element="impl:fault11"/>
  </wsdl:message>
  <wsdl:message name="UnexpectedError">
    <wsdl:part name="fault" element="impl:fault"/>
  </wsdl:message>
  <wsdl:message name="UnrecognizedQualifier">
    <wsdl:part name="fault" element="impl:fault13"/>
  </wsdl:message>
  <wsdl:message name="getCTSVersionRequest">
    <wsdl:part name="parameters" element="impl:getCTSVersion"/>
  </wsdl:message>
  <wsdl:message name="subsumesRequest">
    <wsdl:part name="parameters" element="impl:subsumes"/>
  </wsdl:message>
  <wsdl:message name="getSupportedVocabularyDomainsRequest">
    <wsdl:part name="parameters" element="impl:getSupportedVocabularyDomains"/>
  </wsdl:message>
  <wsdl:message name="UnknownConceptCode">
    <wsdl:part name="fault" element="impl:fault9"/>
  </wsdl:message>
  <wsdl:message name="getServiceDescriptionRequest">
    <wsdl:part name="parameters" element="impl:getServiceDescription"/>
  </wsdl:message>
  <wsdl:message name="subsumesResponse">
    <wsdl:part name="parameters" element="impl:subsumesResponse"/>
  </wsdl:message>
  <wsdl:message name="UnknownApplicationContextCode">
    <wsdl:part name="fault" element="impl:fault5"/>
  </wsdl:message>
  <wsdl:message name="UnknownCodeSystem">
    <wsdl:part name="fault" element="impl:fault7"/>
  </wsdl:message>
```

```
  <wsdl:message name="getServiceVersionResponse">
    <wsdl:part name="parameters" element="impl:getServiceVersionResponse"/>
  </wsdl:message>
  <wsdl:message name="getSupportedMatchAlgorithmsRequest">
    <wsdl:part name="parameters" element="impl:getSupportedMatchAlgorithms"/>
  </wsdl:message>
  <wsdl:message name="UnableToTranslate">
    <wsdl:part name="fault" element="impl:fault8"/>
  </wsdl:message>
  <wsdl:message name="lookupValueSetExpansionRequest">
    <wsdl:part name="parameters" element="impl:lookupValueSetExpansion"/>
  </wsdl:message>
  <wsdl:message name="getCTSVersionResponse">
    <wsdl:part name="parameters" element="impl:getCTSVersionResponse"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameResponse">
    <wsdl:part name="parameters" element="impl:getServiceNameResponse"/>
  </wsdl:message>
  <wsdl:message name="translateCodeRequest">
    <wsdl:part name="parameters" element="impl:translateCode"/>
  </wsdl:message>
  <wsdl:message name="areEquivalentResponse">
    <wsdl:part name="parameters" element="impl:areEquivalentResponse"/>
  </wsdl:message>
  <wsdl:message name="getSupportedMatchAlgorithmsResponse">
    <wsdl:part name="parameters" element="impl:getSupportedMatchAlgorithmsRespo
nse"/>
  </wsdl:message>
  <wsdl:message name="validateTranslationResponse">
    <wsdl:part name="parameters" element="impl:validateTranslationResponse"/>
  </wsdl:message>
  <wsdl:message name="TimeoutError">
    <wsdl:part name="fault" element="impl:fault3"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameRequest">
    <wsdl:part name="parameters" element="impl:getServiceName"/>
  </wsdl:message>
  <wsdl:message name="validateTranslationRequest">
    <wsdl:part name="parameters" element="impl:validateTranslation"/>
  </wsdl:message>
  <wsdl:message name="expandValueSetExpansionContextResponse">
    <wsdl:part name="parameters" element="impl:expandValueSetExpansionContextRe
sponse"/>
  </wsdl:message>
  <wsdl:message name="getHL7ReleaseVersionResponse">
    <wsdl:part name="parameters" element="impl:getHL7ReleaseVersionResponse"/>
  </wsdl:message>
  <wsdl:message name="expandValueSetExpansionContextRequest">
    <wsdl:part name="parameters" element="impl:expandValueSetExpansionContext"/>
  </wsdl:message>
  <wsdl:message name="validateCodeRequest">
    <wsdl:part name="parameters" element="impl:validateCode"/>
  </wsdl:message>
  <wsdl:message name="QualifiersNotSupported">
    <wsdl:part name="fault" element="impl:fault14"/>
  </wsdl:message>
```

```
    <wsdl:message name="getServiceDescriptionResponse">
      <wsdl:part name="parameters" element="impl:getServiceDescriptionResponse"/>
    </wsdl:message>
    <wsdl:message name="validateCodeResponse">
      <wsdl:part name="parameters" element="impl:validateCodeResponse"/>
    </wsdl:message>
    <wsdl:message name="NoApplicableValueSet">
      <wsdl:part name="fault" element="impl:fault6"/>
    </wsdl:message>
    <wsdl:message name="fillInDetailsResponse">
      <wsdl:part name="parameters" element="impl:fillInDetailsResponse"/>
    </wsdl:message>
    <wsdl:message name="areEquivalentRequest">
      <wsdl:part name="parameters" element="impl:areEquivalent"/>
    </wsdl:message>
    <wsdl:message name="getSupportedVocabularyDomainsResponse">
      <wsdl:part name="parameters" element="impl:getSupportedVocabularyDomainsRes
ponse"/>
    </wsdl:message>
    <wsdl:message name="getServiceVersionRequest">
      <wsdl:part name="parameters" element="impl:getServiceVersion"/>
    </wsdl:message>
    <wsdl:message name="SubsumptionNotSupported">
      <wsdl:part name="fault" element="impl:fault12"/>
    </wsdl:message>
    <wsdl:message name="BadlyFormedMatchText">
      <wsdl:part name="fault" element="impl:fault1"/>
    </wsdl:message>
    <wsdl:message name="InvalidExpansionContext">
      <wsdl:part name="fault" element="impl:fault15"/>
    </wsdl:message>
    <wsdl:message name="UnknownLanguage">
      <wsdl:part name="fault" element="impl:fault10"/>
    </wsdl:message>
    <wsdl:message name="UnknownVocabularyDomain">
      <wsdl:part name="fault" element="impl:fault4"/>
    </wsdl:message>
    <wsdl:message name="lookupValueSetExpansionResponse">
      <wsdl:part name="parameters" element="impl:lookupValueSetExpansionResponse"/>
    </wsdl:message>
    <wsdl:message name="getHL7ReleaseVersionRequest">
      <wsdl:part name="parameters" element="impl:getHL7ReleaseVersion"/>
    </wsdl:message>
    <wsdl:message name="translateCodeResponse">
      <wsdl:part name="parameters" element="impl:translateCodeResponse"/>
    </wsdl:message>
    <wsdl:message name="fillInDetailsRequest">
      <wsdl:part name="parameters" element="impl:fillInDetails"/>
    </wsdl:message>
    <wsdl:portType name="RuntimeOperations">
      <wsdl:operation name="getSupportedMatchAlgorithms">
        <wsdl:input name="getSupportedMatchAlgorithmsRequest"
message="impl:getSupportedMatchAlgorithmsRequest"/>
        <wsdl:output name="getSupportedMatchAlgorithmsResponse"
message="impl:getSupportedMatchAlgorithmsResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
```

```
    </wsdl:operation>
    <wsdl:operation name="getSupportedVocabularyDomains">
      <wsdl:input name="getSupportedVocabularyDomainsRequest"
message="impl:getSupportedVocabularyDomainsRequest"/>
      <wsdl:output name="getSupportedVocabularyDomainsResponse"
message="impl:getSupportedVocabularyDomainsResponse"/>
      <wsdl:fault name="UnknownMatchAlgorithm"
message="impl:UnknownMatchAlgorithm"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="BadlyFormedMatchText" message="impl:BadlyFormedMatchText"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
    </wsdl:operation>
    <wsdl:operation name="validateCode">
      <wsdl:input name="validateCodeRequest" message="impl:validateCodeRequest"/>
      <wsdl:output name="validateCodeResponse"
message="impl:validateCodeResponse"/>
      <wsdl:fault name="NoApplicableValueSet" message="impl:NoApplicableValueSet"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownApplicationContextCode" message="impl:UnknownApp
licationContextCode"/>
      <wsdl:fault name="UnknownVocabularyDomain"
message="impl:UnknownVocabularyDomain"/>
    </wsdl:operation>
    <wsdl:operation name="validateTranslation">
      <wsdl:input name="validateTranslationRequest"
message="impl:validateTranslationRequest"/>
      <wsdl:output name="validateTranslationResponse"
message="impl:validateTranslationResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownApplicationContextCode"
message="impl:UnknownApplicationContextCode"/>
      <wsdl:fault name="UnknownVocabularyDomain"
message="impl:UnknownVocabularyDomain"/>
    </wsdl:operation>
    <wsdl:operation name="translateCode">
      <wsdl:input name="translateCodeRequest" message="impl:translateCodeRequest"/>
      <wsdl:output name="translateCodeResponse"
message="impl:translateCodeResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnableToTranslate" message="impl:UnableToTranslate"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      <wsdl:fault name="UnknownApplicationContextCode"
message="impl:UnknownApplicationContextCode"/>
      <wsdl:fault name="UnknownVocabularyDomain"
message="impl:UnknownVocabularyDomain"/>
    </wsdl:operation>
    <wsdl:operation name="fillInDetails">
      <wsdl:input name="fillInDetailsRequest" message="impl:fillInDetailsRequest"/>
      <wsdl:output name="fillInDetailsResponse"
message="impl:fillInDetailsResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownLanguage" message="impl:UnknownLanguage"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      <wsdl:fault name="NoApplicableDesignationFound"
message="impl:NoApplicableDesignationFound"/>
```

```
    </wsdl:operation>
    <wsdl:operation name="subsumes">
      <wsdl:input name="subsumesRequest" message="impl:subsumesRequest"/>
      <wsdl:output name="subsumesResponse" message="impl:subsumesResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="SubsumptionNotSupported"
message="impl:SubsumptionNotSupported"/>
      <wsdl:fault name="QualifiersNotSupported"
message="impl:QualifiersNotSupported"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="UnrecognizedQualifier"
message="impl:UnrecognizedQualifier"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
    </wsdl:operation>
    <wsdl:operation name="areEquivalent">
      <wsdl:input name="areEquivalentRequest" message="impl:areEquivalentRequest"/>
      <wsdl:output name="areEquivalentResponse"
message="impl:areEquivalentResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="SubsumptionNotSupported"
message="impl:SubsumptionNotSupported"/>
      <wsdl:fault name="QualifiersNotSupported"
message="impl:QualifiersNotSupported"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="UnrecognizedQualifier"
message="impl:UnrecognizedQualifier"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
    </wsdl:operation>
    <wsdl:operation name="lookupValueSetExpansion">
      <wsdl:input name="lookupValueSetExpansionRequest"
message="impl:lookupValueSetExpansionRequest"/>
      <wsdl:output name="lookupValueSetExpansionResponse"
message="impl:lookupValueSetExpansionResponse"/>
      <wsdl:fault name="NoApplicableValueSet" message="impl:NoApplicableValueS
et"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownLanguage" message="impl:UnknownLanguage"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
      <wsdl:fault name="UnknownApplicationContextCode"
message="impl:UnknownApplicationContextCode"/>
      <wsdl:fault name="UnknownVocabularyDomain"
message="impl:UnknownVocabularyDomain"/>
    </wsdl:operation>
    <wsdl:operation name="expandValueSetExpansionContext">
      <wsdl:input name="expandValueSetExpansionContextRequest"
message="impl:expandValueSetExpansionContextRequest"/>
      <wsdl:output name="expandValueSetExpansionContextResponse"
message="impl:expandValueSetExpansionContextResponse"/>
      <wsdl:fault name="InvalidExpansionContext"
message="impl:InvalidExpansionContext"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
    </wsdl:operation>
    <wsdl:operation name="getServiceName">
      <wsdl:input name="getServiceNameRequest"
message="impl:getServiceNameRequest"/>
```

```
        <wsdl:output name="getServiceNameResponse"
message="impl:getServiceNameResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceVersion">
        <wsdl:input name="getServiceVersionRequest"
message="impl:getServiceVersionRequest"/>
        <wsdl:output name="getServiceVersionResponse"
message="impl:getServiceVersionResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceDescription">
        <wsdl:input name="getServiceDescriptionRequest"
message="impl:getServiceDescriptionRequest"/>
        <wsdl:output name="getServiceDescriptionResponse"
message="impl:getServiceDescriptionResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getHL7ReleaseVersion">
        <wsdl:input name="getHL7ReleaseVersionRequest"
message="impl:getHL7ReleaseVersionRequest"/>
        <wsdl:output name="getHL7ReleaseVersionResponse"
message="impl:getHL7ReleaseVersionResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getCTSVersion">
        <wsdl:input name="getCTSVersionRequest" message="impl:getCTSVersionRequest"/>
        <wsdl:output name="getCTSVersionResponse"
message="impl:getCTSVersionResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="MessageRuntimeServiceSoapBinding"
type="impl:RuntimeOperations">
      <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/
soap/http"/>
      <wsdl:operation name="getSupportedMatchAlgorithms">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getSupportedMatchAlgorithmsRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getSupportedMatchAlgorithmsResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="UnexpectedError">
          <wsdlsoap:fault name="UnexpectedError" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getSupportedVocabularyDomains">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getSupportedVocabularyDomainsRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getSupportedVocabularyDomainsResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="UnknownMatchAlgorithm">
```

```
      <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="BadlyFormedMatchText">
      <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="TimeoutError">
      <wsdlsoap:fault name="TimeoutError" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="validateCode">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="validateCodeRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="validateCodeResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NoApplicableValueSet">
      <wsdlsoap:fault name="NoApplicableValueSet" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownApplicationContextCode">
      <wsdlsoap:fault name="UnknownApplicationContextCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownVocabularyDomain">
      <wsdlsoap:fault name="UnknownVocabularyDomain" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="validateTranslation">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="validateTranslationRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="validateTranslationResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownApplicationContextCode">
      <wsdlsoap:fault name="UnknownApplicationContextCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownVocabularyDomain">
      <wsdlsoap:fault name="UnknownVocabularyDomain" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="translateCode">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="translateCodeRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="translateCodeResponse">
```

```
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnableToTranslate">
      <wsdlsoap:fault name="UnableToTranslate" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownApplicationContextCode">
      <wsdlsoap:fault name="UnknownApplicationContextCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownVocabularyDomain">
      <wsdlsoap:fault name="UnknownVocabularyDomain" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="fillInDetails">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="fillInDetailsRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="fillInDetailsResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguage">
      <wsdlsoap:fault name="UnknownLanguage" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NoApplicableDesignationFound">
      <wsdlsoap:fault name="NoApplicableDesignationFound" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="subsumes">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="subsumesRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="subsumesResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="SubsumptionNotSupported">
      <wsdlsoap:fault name="SubsumptionNotSupported" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="QualifiersNotSupported">
```

```
      <wsdlsoap:fault name="QualifiersNotSupported" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnrecognizedQualifier">
      <wsdlsoap:fault name="UnrecognizedQualifier" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="areEquivalent">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="areEquivalentRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="areEquivalentResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="SubsumptionNotSupported">
      <wsdlsoap:fault name="SubsumptionNotSupported" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="QualifiersNotSupported">
      <wsdlsoap:fault name="QualifiersNotSupported" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnrecognizedQualifier">
      <wsdlsoap:fault name="UnrecognizedQualifier" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupValueSetExpansion">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupValueSetExpansionRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="lookupValueSetExpansionResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="NoApplicableValueSet">
      <wsdlsoap:fault name="NoApplicableValueSet" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguage">
      <wsdlsoap:fault name="UnknownLanguage" use="literal"/>
    </wsdl:fault>
```

```
      <wsdl:fault name="TimeoutError">
        <wsdlsoap:fault name="TimeoutError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownApplicationContextCode">
        <wsdlsoap:fault name="UnknownApplicationContextCode" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownVocabularyDomain">
        <wsdlsoap:fault name="UnknownVocabularyDomain" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="expandValueSetExpansionContext">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="expandValueSetExpansionContextRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="expandValueSetExpansionContextResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="InvalidExpansionContext">
        <wsdlsoap:fault name="InvalidExpansionContext" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TimeoutError">
        <wsdlsoap:fault name="TimeoutError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceName">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceNameRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceNameResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceDescription">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceDescriptionRequest">
        <wsdlsoap:body use="literal"/>
```

```
      </wsdl:input>
      <wsdl:output name="getServiceDescriptionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getHL7ReleaseVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getHL7ReleaseVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getHL7ReleaseVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getCTSVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getCTSVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getCTSVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="RuntimeOperationsService">
    <wsdl:port name="MessageRuntimeService" binding="impl:MessageRuntimeService
SoapBinding">
      <wsdlsoap:address location="http://localhost:8080/axis/services/
MessageRuntimeService"/>
    </wsdl:port>
  </wsdl:service>
  <!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->
</wsdl:definitions>
```

### 17.2 Описание API обозревателя на уровне сообщений

Ниже приведено описание службы обозревателя MessageBrowser на языке WSDL (файл MessageBrowser.wsdl):

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn://hl7.org/CTSMAPI"
xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="urn://hl7.org/
CTSMAPI" xmlns:intf="urn://hl7.org/CTSMAPI" xmlns:tns2="urn://cts.hl7.org/types"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.
xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->
```

```
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="urn://hl7.org/CTSMAPI"
xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="urn://cts.hl7.org/types"/>
   <element name="getSupportedMatchAlgorithms">
    <complexType/>
   </element>
   <element name="getSupportedMatchAlgorithmsResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="getSupportedMatchAlgorithmsReturn"
type="tns2:ST"/>
     </sequence>
    </complexType>
   </element>
   <complexType name="UnexpectedError">
    <sequence>
     <element name="possible_cause" nillable="true" type="tns2:ST"/>
    </sequence>
   </complexType>
   <element name="fault" type="impl:UnexpectedError"/>
   <element name="getSupportedAttributes">
    <complexType>
     <sequence>
      <element name="in0" type="tns2:ST"/>
      <element name="in1" type="tns2:ST"/>
      <element name="in2" type="xsd:int"/>
      <element name="in3" type="xsd:int"/>
     </sequence>
    </complexType>
   </element>
   <element name="getSupportedAttributesResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="getSupportedAttributesReturn"
type="impl:RIMCodedAttribute"/>
     </sequence>
    </complexType>
   </element>
   <complexType name="RIMAttributeId">
    <sequence>
     <element name="model_id" nillable="true" type="tns2:ST"/>
     <element name="class_name" nillable="true" type="tns2:ST"/>
     <element name="attribute_name" nillable="true" type="tns2:ST"/>
    </sequence>
   </complexType>
   <complexType name="RIMCodedAttribute">
    <sequence>
     <element name="RIMAttribute_id" nillable="true"
type="impl:RIMAttributeId"/>
     <element name="dataType_code" nillable="true" type="tns2:ST"/>
     <element name="codingStrength_code" nillable="true" type="tns2:ST"/>
     <element name="vocabularyDomain_name" nillable="true" type="tns2:ST"/>
    </sequence>
   </complexType>
   <complexType name="BadlyFormedMatchText">
```

```
    <sequence>
     <element name="matchText" nillable="true" type="tns2:ST"/>
    </sequence>
   </complexType>
   <element name="fault1" type="impl:BadlyFormedMatchText"/>
   <complexType name="UnknownMatchAlgorithm">
    <sequence>
     <element name="matchAlgorithm_code" nillable="true" type="tns2:ST"/>
    </sequence>
   </complexType>
   <element name="fault2" type="impl:UnknownMatchAlgorithm"/>
   <complexType name="TimeoutError">
    <sequence/>
   </complexType>
   <element name="fault3" type="impl:TimeoutError"/>
   <element name="getSupportedVocabularyDomains">
    <complexType>
     <sequence>
      <element name="in0" type="tns2:ST"/>
      <element name="in1" type="tns2:ST"/>
      <element name="in2" type="xsd:int"/>
      <element name="in3" type="xsd:int"/>
     </sequence>
    </complexType>
   </element>
   <element name="getSupportedVocabularyDomainsResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="getSupportedVocabularyDomainsReturn"
type="tns2:ST"/>
     </sequence>
    </complexType>
   </element>
   <element name="getSupportedValueSets">
    <complexType>
     <sequence>
      <element name="in0" type="tns2:ST"/>
      <element name="in1" type="tns2:ST"/>
      <element name="in2" type="xsd:int"/>
      <element name="in3" type="xsd:int"/>
     </sequence>
    </complexType>
   </element>
   <element name="getSupportedValueSetsResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="getSupportedValueSetsReturn" type="im
pl:ValueSetDescriptor"/>
     </sequence>
    </complexType>
   </element>
   <complexType name="ValueSetDescriptor">
    <sequence>
     <element name="valueSet_id" nillable="true" type="tns2:UID"/>
     <element name="valueSet_name" nillable="true" type="tns2:ST"/>
    </sequence>
```

```
    </complexType>
    <element name="getSupportedCodeSystems">
     <complexType>
      <sequence>
       <element name="in0" type="tns2:ST"/>
       <element name="in1" type="tns2:ST"/>
       <element name="in2" type="xsd:int"/>
       <element name="in3" type="xsd:int"/>
      </sequence>
     </complexType>
    </element>
    <element name="getSupportedCodeSystemsResponse">
     <complexType>
      <sequence>
       <element maxOccurs="unbounded" name="getSupportedCodeSystemsReturn" type="
impl:CodeSystemDescriptor"/>
      </sequence>
     </complexType>
    </element>
    <complexType name="ArrayOf_tns2_ST">
     <sequence>
      <element maxOccurs="unbounded" minOccurs="0" name="item" type="tns2:ST"/>
     </sequence>
    </complexType>
    <complexType name="CodeSystemDescriptor">
     <sequence>
      <element name="codeSystem_id" nillable="true" type="tns2:UID"/>
      <element name="codeSystem_name" nillable="true" type="tns2:ST"/>
      <element name="copyright" nillable="true" type="tns2:ST"/>
      <element name="availableReleases" nillable="true" type="impl:ArrayOf_tns2_ST"/>
     </sequence>
    </complexType>
    <element name="lookupVocabularyDomain">
     <complexType>
      <sequence>
       <element name="in0" type="tns2:ST"/>
      </sequence>
     </complexType>
    </element>
    <element name="lookupVocabularyDomainResponse">
     <complexType>
      <sequence>
       <element name="lookupVocabularyDomainReturn" type="impl:VocabularyDomainDe
scription"/>
      </sequence>
     </complexType>
    </element>
    <complexType name="ArrayOfRIMCodedAttribute">
     <sequence>
      <element maxOccurs="unbounded" minOccurs="0" name="item"
type="impl:RIMCodedAttribute"/>
     </sequence>
    </complexType>
    <complexType name="VocabularyDomainValueSet">
     <sequence>
      <element name="definedByValueSet" nillable="true" type="impl:ValueSetDescrip
tor"/>
```

```
      <element name="applicationContext_code" nillable="true" type="tns2:ST"/>
     </sequence>
    </complexType>
    <complexType name="ArrayOfVocabularyDomainValueSet">
     <sequence>
      <element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:Vocabul
aryDomainValueSet"/>
     </sequence>
    </complexType>
    <complexType name="VocabularyDomainDescription">
     <sequence>
      <element name="vocabularyDomain_name" nillable="true" type="tns2:ST"/>
      <element name="description" nillable="true" type="tns2:ST"/>
      <element name="restrictsDomain_name" nillable="true" type="tns2:ST"/>
      <element name="basisOfDomains" nillable="true" type="impl:ArrayOf_tns2_ST"/>
      <element name="constrainsAttributes" nillable="true" type="impl:ArrayOfRIMC
odedAttribute"/>
      <element name="representedByValueSets" nillable="true" type="impl:ArrayOfVo
cabularyDomainValueSet"/>
     </sequence>
    </complexType>
    <complexType name="UnknownVocabularyDomain">
     <sequence>
      <element name="vocabularyDomain_name" nillable="true" type="tns2:ST"/>
     </sequence>
    </complexType>
    <element name="fault4" type="impl:UnknownVocabularyDomain"/>
    <element name="lookupValueSet">
     <complexType>
      <sequence>
       <element name="in0" type="tns2:UID"/>
       <element name="in1" type="tns2:ST"/>
      </sequence>
     </complexType>
    </element>
    <element name="lookupValueSetResponse">
     <complexType>
      <sequence>
       <element name="lookupValueSetReturn" type="impl:FullValueSetDescription"/>
      </sequence>
     </complexType>
    </element>
    <complexType name="ValueSetDescription">
     <sequence>
      <element name="idAndName" nillable="true" type="impl:ValueSetDescriptor"/>
      <element name="description" nillable="true" type="tns2:ST"/>
      <element name="definingExpression" nillable="true" type="tns2:ST"/>
      <element name="basedOnCodeSystem" nillable="true" type="impl:CodeSystemDesc
riptor"/>
      <element name="allCodes" nillable="true" type="tns2:BL"/>
      <element name="head_code" nillable="true" type="tns2:ST"/>
     </sequence>
    </complexType>
    <complexType name="ValueSetConstructor">
     <sequence>
      <element name="includedValueSet" nillable="true" type="impl:ValueSetDescrip
tor"/>
```

```
      <element name="includeHeadCode" nillable="true" type="tns2:BL"/>
     </sequence>
    </complexType>
    <complexType name="ArrayOfValueSetConstructor">
     <sequence>
      <element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:ValueSe
tConstructor"/>
     </sequence>
    </complexType>
    <complexType name="ArrayOfValueSetDescriptor">
     <sequence>
      <element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:ValueSe
tDescriptor"/>
     </sequence>
    </complexType>
    <complexType name="ValueSetCodeReference">
     <sequence>
      <element name="referenced_code" nillable="true" type="tns2:ST"/>
      <element name="relationship_code" nillable="true" type="tns2:ST"/>
      <element name="includeReferencedCode" nillable="true" type="tns2:BL"/>
      <element name="leafOnly" nillable="true" type="tns2:BL"/>
     </sequence>
    </complexType>
    <complexType name="ArrayOfValueSetCodeReference">
     <sequence>
      <element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:ValueSe
tCodeReference"/>
     </sequence>
    </complexType>
    <complexType name="FullValueSetDescription">
     <sequence>
      <element name="description" nillable="true" type="impl:ValueSetDescripti
on"/>
      <element name="constructedUsingValueSets" nillable="true" type="impl:Array
OfValueSetConstructor"/>
      <element name="usedToDefine" nillable="true" type="impl:ArrayOfValueSetDescr
iptor"/>
      <element name="referencesCodes" nillable="true" type="impl:ArrayOfValueSetC
odeReference"/>
     </sequence>
    </complexType>
    <complexType name="UnknownValueSet">
     <sequence>
      <element name="valueSet" nillable="true" type="impl:ValueSetDescriptor"/>
     </sequence>
    </complexType>
    <element name="fault5" type="impl:UnknownValueSet"/>
    <complexType name="ValueSetNameIdMismatch">
     <sequence>
      <element name="valueSet_id" nillable="true" type="tns2:UID"/>
      <element name="valueSet_name" nillable="true" type="tns2:ST"/>
     </sequence>
    </complexType>
    <element name="fault6" type="impl:ValueSetNameIdMismatch"/>
    <element name="lookupCodeSystem">
     <complexType>
```

```
      <sequence>
       <element name="in0" type="tns2:UID"/>
       <element name="in1" type="tns2:ST"/>
      </sequence>
     </complexType>
    </element>
    <element name="lookupCodeSystemResponse">
     <complexType>
      <sequence>
       <element name="lookupCodeSystemReturn" type="impl:CodeSystemInfo"/>
      </sequence>
     </complexType>
    </element>
    <complexType name="CodeSystemRegistration">
     <sequence>
      <element name="sponsor" nillable="true" type="tns2:ST"/>
      <element name="publisher" nillable="true" type="tns2:ST"/>
      <element name="versionReportingMethod" nillable="true" type="tns2:ST"/>
      <element name="licensingInformation" nillable="true" type="tns2:ST"/>
      <element name="inUMLS" nillable="true" type="tns2:BL"/>
      <element name="systemSpecificLocatorInfo" nillable="true" type="tns2:ST"/>
      <element name="codeSystemType_code" nillable="true" type="tns2:ST"/>
     </sequence>
    </complexType>
    <complexType name="CodeSystemInfo">
     <sequence>
      <element name="description" nillable="true" type="impl:CodeSystemDescriptor"/>
      <element name="registrationInfo" nillable="true" type="impl:CodeSystemRegis
tration"/>
     </sequence>
    </complexType>
    <complexType name="UnknownCodeSystem">
     <sequence>
      <element name="codeSystem_id" nillable="true" type="tns2:UID"/>
     </sequence>
    </complexType>
    <element name="fault7" type="impl:UnknownCodeSystem"/>
    <complexType name="CodeSystemNameIdMismatch">
     <sequence>
      <element name="codeSystem_id" nillable="true" type="tns2:UID"/>
      <element name="codeSystem_name" nillable="true" type="tns2:ST"/>
     </sequence>
    </complexType>
    <element name="fault8" type="impl:CodeSystemNameIdMismatch"/>
    <element name="lookupValueSetForDomain">
     <complexType>
      <sequence>
       <element name="in0" type="tns2:ST"/>
       <element name="in1" type="tns2:ST"/>
      </sequence>
     </complexType>
    </element>
    <element name="lookupValueSetForDomainResponse">
     <complexType>
      <sequence>
       <element name="lookupValueSetForDomainReturn" type="impl:ValueSetDescript
or"/>
```

```
    </sequence>
   </complexType>
  </element>
  <complexType name="UnknownApplicationContextCode">
   <sequence>
    <element name="applicationContext_code" nillable="true" type="tns2:ST"/>
   </sequence>
  </complexType>
  <element name="fault9" type="impl:UnknownApplicationContextCode"/>
  <complexType name="NoApplicableValueSet">
   <sequence>
    <element name="vocabularyDomain_name" nillable="true" type="tns2:ST"/>
    <element name="applicationContext_code" nillable="true" type="tns2:ST"/>
   </sequence>
  </complexType>
  <element name="fault10" type="impl:NoApplicableValueSet"/>
  <element name="isCodeInValueSet">
   <complexType>
    <sequence>
     <element name="in0" type="tns2:UID"/>
     <element name="in1" type="tns2:ST"/>
     <element name="in2" type="tns2:BL"/>
     <element name="in3" type="impl:ConceptId"/>
    </sequence>
   </complexType>
  </element>
  <complexType name="ConceptId">
   <sequence>
    <element name="codeSystem_id" nillable="true" type="tns2:UID"/>
    <element name="concept_code" nillable="true" type="tns2:ST"/>
   </sequence>
  </complexType>
  <element name="isCodeInValueSetResponse">
   <complexType>
    <sequence>
     <element name="isCodeInValueSetReturn" type="tns2:BL"/>
    </sequence>
   </complexType>
  </element>
  <complexType name="UnknownConceptCode">
   <sequence>
    <element name="concept_id" nillable="true" type="impl:ConceptId"/>
   </sequence>
  </complexType>
  <element name="fault11" type="impl:UnknownConceptCode"/>
  <element name="getServiceName">
   <complexType/>
  </element>
  <element name="getServiceNameResponse">
   <complexType>
    <sequence>
     <element name="getServiceNameReturn" type="tns2:ST"/>
    </sequence>
   </complexType>
  </element>
  <element name="getServiceVersion">
```

```
    <complexType/>
   </element>
   <element name="getServiceVersionResponse">
    <complexType>
     <sequence>
      <element name="getServiceVersionReturn" type="tns2:ST"/>
     </sequence>
    </complexType>
   </element>
   <element name="getServiceDescription">
    <complexType/>
   </element>
   <element name="getServiceDescriptionResponse">
    <complexType>
     <sequence>
      <element name="getServiceDescriptionReturn" type="tns2:ST"/>
     </sequence>
    </complexType>
   </element>
   <element name="getHL7ReleaseVersion">
    <complexType/>
   </element>
   <element name="getHL7ReleaseVersionResponse">
    <complexType>
     <sequence>
      <element name="getHL7ReleaseVersionReturn" type="tns2:ST"/>
     </sequence>
    </complexType>
   </element>
   <element name="getCTSVersion">
    <complexType/>
   </element>
   <element name="getCTSVersionResponse">
    <complexType>
     <sequence>
      <element name="getCTSVersionReturn" type="impl:CTSVersionId"/>
     </sequence>
    </complexType>
   </element>
   <complexType name="CTSVersionId">
    <sequence>
     <element name="major" nillable="true" type="tns2:INT"/>
     <element name="minor" nillable="true" type="tns2:INT"/>
    </sequence>
   </complexType>
  </schema>
  <schema elementFormDefault="qualified" targetNamespace="urn://cts.hl7.org/
types" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="urn://hl7.org/CTSMAPI"/>
   <complexType name="ST">
    <sequence>
     <element name="v" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="UID">
    <sequence>
```

```
      <element name="v" nillable="true" type="xsd:string"/>
     </sequence>
    </complexType>
    <complexType name="BL">
     <sequence>
      <element name="v" type="xsd:boolean"/>
     </sequence>
    </complexType>
    <complexType name="INT">
     <sequence>
      <element name="v" type="xsd:int"/>
     </sequence>
    </complexType>
   </schema>
  </wsdl:types>

    <wsdl:message name="UnknownMatchAlgorithm">
       <wsdl:part element="impl:fault2" name="fault"/>
    </wsdl:message>
    <wsdl:message name="lookupValueSetForDomainRequest">
       <wsdl:part element="impl:lookupValueSetForDomain" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="UnexpectedError">
       <wsdl:part element="impl:fault" name="fault"/>
    </wsdl:message>
    <wsdl:message name="getSupportedCodeSystemsRequest">
       <wsdl:part element="impl:getSupportedCodeSystems" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="lookupValueSetRequest">
       <wsdl:part element="impl:lookupValueSet" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="getCTSVersionRequest">
       <wsdl:part element="impl:getCTSVersion" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="getSupportedVocabularyDomainsRequest">
       <wsdl:part element="impl:getSupportedVocabularyDomains"
name="parameters"/>
    </wsdl:message>
    <wsdl:message name="UnknownConceptCode">
       <wsdl:part element="impl:fault11" name="fault"/>
    </wsdl:message>
    <wsdl:message name="getServiceDescriptionRequest">
       <wsdl:part element="impl:getServiceDescription" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="UnknownApplicationContextCode">
       <wsdl:part element="impl:fault9" name="fault"/>
    </wsdl:message>
    <wsdl:message name="UnknownCodeSystem">
       <wsdl:part element="impl:fault7" name="fault"/>
    </wsdl:message>
    <wsdl:message name="getServiceVersionResponse">
       <wsdl:part element="impl:getServiceVersionResponse" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="getSupportedMatchAlgorithmsRequest">
       <wsdl:part element="impl:getSupportedMatchAlgorithms" name="parameters"/>
    </wsdl:message>
```

```
   <wsdl:message name="lookupCodeSystemResponse">
      <wsdl:part element="impl:lookupCodeSystemResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="lookupValueSetResponse">
      <wsdl:part element="impl:lookupValueSetResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getSupportedAttributesResponse">
      <wsdl:part element="impl:getSupportedAttributesResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getSupportedValueSetsResponse">
      <wsdl:part element="impl:getSupportedValueSetsResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getCTSVersionResponse">
      <wsdl:part element="impl:getCTSVersionResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getServiceNameResponse">
      <wsdl:part element="impl:getServiceNameResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="UnknownValueSet">
      <wsdl:part element="impl:fault5" name="fault"/>
   </wsdl:message>
   <wsdl:message name="getSupportedMatchAlgorithmsResponse">
      <wsdl:part element="impl:getSupportedMatchAlgorithmsResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="CodeSystemNameIdMismatch">
      <wsdl:part element="impl:fault8" name="fault"/>
   </wsdl:message>
   <wsdl:message name="lookupValueSetForDomainResponse">
      <wsdl:part element="impl:lookupValueSetForDomainResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="TimeoutError">
      <wsdl:part element="impl:fault3" name="fault"/>
   </wsdl:message>
   <wsdl:message name="getServiceNameRequest">
      <wsdl:part element="impl:getServiceName" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getHL7ReleaseVersionResponse">
      <wsdl:part element="impl:getHL7ReleaseVersionResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getSupportedValueSetsRequest">
      <wsdl:part element="impl:getSupportedValueSets" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getServiceDescriptionResponse">
      <wsdl:part element="impl:getServiceDescriptionResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="lookupVocabularyDomainRequest">
      <wsdl:part element="impl:lookupVocabularyDomain" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="NoApplicableValueSet">
      <wsdl:part element="impl:fault10" name="fault"/>
   </wsdl:message>
```

```
   <wsdl:message name="getSupportedAttributesRequest">
      <wsdl:part element="impl:getSupportedAttributes" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getSupportedCodeSystemsResponse">
      <wsdl:part element="impl:getSupportedCodeSystemsResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getSupportedVocabularyDomainsResponse">
      <wsdl:part element="impl:getSupportedVocabularyDomainsResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getServiceVersionRequest">
      <wsdl:part element="impl:getServiceVersion" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="BadlyFormedMatchText">
      <wsdl:part element="impl:fault3" name="fault"/>
   </wsdl:message>
   <wsdl:message name="lookupVocabularyDomainResponse">
      <wsdl:part element="impl:lookupVocabularyDomainResponse"
name="parameters"/>
   </wsdl:message>
   <wsdl:message name="isCodeInValueSetRequest">
      <wsdl:part element="impl:isCodeInValueSet" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="lookupCodeSystemRequest">
      <wsdl:part element="impl:lookupCodeSystem" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="UnknownVocabularyDomain">
      <wsdl:part element="impl:fault4" name="fault"/>
   </wsdl:message>
   <wsdl:message name="isCodeInValueSetResponse">
      <wsdl:part element="impl:isCodeInValueSetResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="getHL7ReleaseVersionRequest">
      <wsdl:part element="impl:getHL7ReleaseVersion" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="ValueSetNameIdMismatch">
      <wsdl:part element="impl:fault6" name="fault"/>
   </wsdl:message>
   <wsdl:portType name="BrowserOperations">
      <wsdl:operation name="getSupportedMatchAlgorithms">
         <wsdl:input message="impl:getSupportedMatchAlgorithmsRequest" name="get
SupportedMatchAlgorithmsRequest"/>
         <wsdl:output message="impl:getSupportedMatchAlgorithmsResponse" name="g
etSupportedMatchAlgorithmsResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getSupportedAttributes">
         <wsdl:input message="impl:getSupportedAttributesRequest" name="getSuppo
rtedAttributesRequest"/>
         <wsdl:output message="impl:getSupportedAttributesResponse" name="getSup
portedAttributesResponse"/>
         <wsdl:fault message="impl:UnknownMatchAlgorithm"
name="UnknownMatchAlgorithm"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
        <wsdl:fault message="impl:BadlyFormedMatchText" name="BadlyFormedMatchText"/>
```

```
         <wsdl:fault message="impl:TimeoutError" name="TimeoutError"/>
      </wsdl:operation>
      <wsdl:operation name="getSupportedVocabularyDomains">
         <wsdl:input message="impl:getSupportedVocabularyDomainsRequest" name="g
etSupportedVocabularyDomainsRequest"/>
         <wsdl:output message="impl:getSupportedVocabularyDomainsResponse" name=
"getSupportedVocabularyDomainsResponse"/>
         <wsdl:fault message="impl:UnknownMatchAlgorithm"
name="UnknownMatchAlgorithm"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:BadlyFormedMatchText"
name="BadlyFormedMatchText"/>
         <wsdl:fault message="impl:TimeoutError" name="TimeoutError"/>
      </wsdl:operation>
      <wsdl:operation name="getSupportedValueSets">
         <wsdl:input message="impl:getSupportedValueSetsRequest" name="getSuppor
tedValueSetsRequest"/>
         <wsdl:output message="impl:getSupportedValueSetsResponse" name="getSupp
ortedValueSetsResponse"/>
         <wsdl:fault message="impl:UnknownMatchAlgorithm"
name="UnknownMatchAlgorithm"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:BadlyFormedMatchText"
name="BadlyFormedMatchText"/>
         <wsdl:fault message="impl:TimeoutError" name="TimeoutError"/>
      </wsdl:operation>
      <wsdl:operation name="getSupportedCodeSystems">
         <wsdl:input message="impl:getSupportedCodeSystemsRequest" name="getSupp
ortedCodeSystemsRequest"/>
         <wsdl:output message="impl:getSupportedCodeSystemsResponse" name="getSu
pportedCodeSystemsResponse"/>
         <wsdl:fault message="impl:UnknownMatchAlgorithm"
name="UnknownMatchAlgorithm"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:BadlyFormedMatchText"
name="BadlyFormedMatchText"/>
         <wsdl:fault message="impl:TimeoutError" name="TimeoutError"/>
      </wsdl:operation>
      <wsdl:operation name="lookupVocabularyDomain">
         <wsdl:input message="impl:lookupVocabularyDomainRequest" name="lookupVo
cabularyDomainRequest"/>
         <wsdl:output message="impl:lookupVocabularyDomainResponse" name="lookup
VocabularyDomainResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:UnknownVocabularyDomain" name="UnknownVocabul
aryDomain"/>
      </wsdl:operation>
      <wsdl:operation name="lookupValueSet">
         <wsdl:input message="impl:lookupValueSetRequest"
name="lookupValueSetRequest"/>
         <wsdl:output message="impl:lookupValueSetResponse"
name="lookupValueSetResponse"/>
         <wsdl:fault message="impl:UnknownValueSet" name="UnknownValueSet"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:ValueSetNameIdMismatch"
name="ValueSetNameIdMismatch"/>
```

```
      </wsdl:operation>
      <wsdl:operation name="lookupCodeSystem">
         <wsdl:input message="impl:lookupCodeSystemRequest" name="lookupCodeSyst
emRequest"/>
         <wsdl:output message="impl:lookupCodeSystemResponse" name="lookupCodeSy
stemResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:CodeSystemNameIdMismatch" name="CodeSystemNam
eIdMismatch"/>
         <wsdl:fault message="impl:UnknownCodeSystem" name="UnknownCodeSystem"/>
      </wsdl:operation>
      <wsdl:operation name="lookupValueSetForDomain">
         <wsdl:input message="impl:lookupValueSetForDomainRequest" name="lookupV
alueSetForDomainRequest"/>
         <wsdl:output message="impl:lookupValueSetForDomainResponse" name="looku
pValueSetForDomainResponse"/>
         <wsdl:fault message="impl:NoApplicableValueSet"
name="NoApplicableValueSet"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:UnknownApplicationContextCode" name="UnknownA
pplicationContextCode"/>
         <wsdl:fault message="impl:UnknownVocabularyDomain" name="UnknownVocabul
aryDomain"/>
      </wsdl:operation>
      <wsdl:operation name="isCodeInValueSet">
         <wsdl:input message="impl:isCodeInValueSetRequest" name="isCodeInValueS
etRequest"/>
         <wsdl:output message="impl:isCodeInValueSetResponse" name="isCodeInValu
eSetResponse"/>
         <wsdl:fault message="impl:UnknownValueSet" name="UnknownValueSet"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
         <wsdl:fault message="impl:ValueSetNameIdMismatch"
name="ValueSetNameIdMismatch"/>
         <wsdl:fault message="impl:UnknownConceptCode"
name="UnknownConceptCode"/>
         <wsdl:fault message="impl:UnknownCodeSystem" name="UnknownCodeSystem"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceName">
         <wsdl:input message="impl:getServiceNameRequest"
name="getServiceNameRequest"/>
         <wsdl:output message="impl:getServiceNameResponse"
name="getServiceNameResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceVersion">
         <wsdl:input message="impl:getServiceVersionRequest" name="getServiceVer
sionRequest"/>
         <wsdl:output message="impl:getServiceVersionResponse" name="getServiceV
ersionResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceDescription">
         <wsdl:input message="impl:getServiceDescriptionRequest" name="getServic
eDescriptionRequest"/>
         <wsdl:output message="impl:getServiceDescriptionResponse" name="getServ
iceDescriptionResponse"/>
```

```
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getHL7ReleaseVersion">
         <wsdl:input message="impl:getHL7ReleaseVersionRequest" name="getHL7Rele
aseVersionRequest"/>
         <wsdl:output message="impl:getHL7ReleaseVersionResponse" name="getHL7R
eleaseVersionResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getCTSVersion">
         <wsdl:input message="impl:getCTSVersionRequest"
name="getCTSVersionRequest"/>
         <wsdl:output message="impl:getCTSVersionResponse"
name="getCTSVersionResponse"/>
         <wsdl:fault message="impl:UnexpectedError" name="UnexpectedError"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="MessageBrowserServiceSoapBinding"
type="impl:BrowserOperations">
      <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/
soap/http"/>
      <wsdl:operation name="getSupportedMatchAlgorithms">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getSupportedMatchAlgorithmsRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getSupportedMatchAlgorithmsResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault name="UnexpectedError" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getSupportedAttributes">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getSupportedAttributesRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getSupportedAttributesResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="UnknownMatchAlgorithm">
            <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
         </wsdl:fault>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault name="UnexpectedError" use="literal"/>
         </wsdl:fault>
         <wsdl:fault name="BadlyFormedMatchText">
            <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
         </wsdl:fault>
         <wsdl:fault name="TimeoutError">
            <wsdlsoap:fault name="TimeoutError" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getSupportedVocabularyDomains">
         <wsdlsoap:operation soapAction=""/>
```

```
      <wsdl:input name="getSupportedVocabularyDomainsRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSupportedVocabularyDomainsResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnknownMatchAlgorithm">
         <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnexpectedError">
         <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="BadlyFormedMatchText">
         <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TimeoutError">
         <wsdlsoap:fault name="TimeoutError" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="getSupportedValueSets">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSupportedValueSetsRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSupportedValueSetsResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnknownMatchAlgorithm">
         <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnexpectedError">
         <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="BadlyFormedMatchText">
         <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TimeoutError">
         <wsdlsoap:fault name="TimeoutError" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="getSupportedCodeSystems">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSupportedCodeSystemsRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSupportedCodeSystemsResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnknownMatchAlgorithm">
         <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnexpectedError">
         <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="BadlyFormedMatchText">
         <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
```

```
      </wsdl:fault>
      <wsdl:fault name="TimeoutError">
         <wsdlsoap:fault name="TimeoutError" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="lookupVocabularyDomain">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="lookupVocabularyDomainRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="lookupVocabularyDomainResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
         <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownVocabularyDomain">
         <wsdlsoap:fault name="UnknownVocabularyDomain" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="lookupValueSet">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="lookupValueSetRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="lookupValueSetResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnknownValueSet">
         <wsdlsoap:fault name="UnknownValueSet" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnexpectedError">
         <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ValueSetNameIdMismatch">
         <wsdlsoap:fault name="ValueSetNameIdMismatch" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="lookupCodeSystem">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="lookupCodeSystemRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="lookupCodeSystemResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
         <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="CodeSystemNameIdMismatch">
         <wsdlsoap:fault name="CodeSystemNameIdMismatch" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownCodeSystem">
         <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
```

```
<wsdl:operation name="lookupValueSetForDomain">
   <wsdlsoap:cperation soapAction=""/>
   <wsdl:input name="lookupValueSetForDomainRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="lookupValueSetForDomainResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
   <wsdl:fault name="NoApplicableValueSet">
      <wsdlsoap:fault name="NoApplicableValueSet" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="UnknownApplicationContextCode">
      <wsdlsoap:fault name="UnknownApplicationContextCode" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="UnknownVocabularyDomain">
      <wsdlsoap:fault name="UnknownVocabularyDomain" use="literal"/>
   </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="isCodeInValueSet">
   <wsdlsoap:cperation soapAction=""/>
   <wsdl:input name="isCodeInValueSetRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="isCodeInValueSetResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
   <wsdl:fault name="UnknownValueSet">
      <wsdlsoap:fault name="UnknownValueSet" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="ValueSetNameIdMismatch">
      <wsdlsoap:fault name="ValueSetNameIdMismatch" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
   </wsdl:fault>
   <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
   </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getServiceName">
   <wsdlsoap:cperation soapAction=""/>
   <wsdl:input name="getServiceNameRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="getServiceNameResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
   <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
   </wsdl:fault>
```

```
      </wsdl:operation>
      <wsdl:operation name="getServiceVersion">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getServiceVersionRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getServiceVersionResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault name="UnexpectedError" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getServiceDescription">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getServiceDescriptionRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getServiceDescriptionResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault name="UnexpectedError" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getHL7ReleaseVersion">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getHL7ReleaseVersionRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getHL7ReleaseVersionResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault name="UnexpectedError" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getCTSVersion">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getCTSVersionRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getCTSVersionResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="UnexpectedError">
            <wsdlsoap:fault name="UnexpectedError" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="BrowserOperationsService">
      <wsdl:port binding="impl:MessageBrowserServiceSoapBinding"
name="MessageBrowserService">
         <wsdlsoap:address location="http://localhost:8080/axis/services/
MessageBrowserService"/>
      </wsdl:port>
```

```
   </wsdl:service>
</wsdl:definitions>
```

### 17.3 Описание API времени исполнения на уровне словаря

Ниже приведено описание службы времени исполнения VocabRuntime на языке WSDL (файл VocabRuntime.wsdl):

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn://hl7.org/CTSVAPI" xmlns:intf="urn://hl7.org/CTSVAPI"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://
schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn://hl7.org/CTSVAPI">
   <wsdl:import namespace="urn://hl7.org/CTSVAPI" location="VocabRuntime.
wsdl#schema1.xsd"/>
   <wsdl:types/>
   <wsdl:message name="areCodesRelatedRequest">
      <wsdl:part name="parameters" element="impl:areCodesRelated"/>
   </wsdl:message>
   <wsdl:message name="UnknownRelationQualifier">
      <wsdl:part name="fault" element="impl:fault8"/>
   </wsdl:message>
   <wsdl:message name="getServiceVersionRequest">
      <wsdl:part name="parameters" element="impl:getServiceVersion"/>
   </wsdl:message>
   <wsdl:message name="lookupDesignationResponse">
      <wsdl:part name="parameters" element="impl:lookupDesignationResponse"/>
   </wsdl:message>
   <wsdl:message name="TimeoutError">
      <wsdl:part name="fault" element="impl:fault"/>
   </wsdl:message>
   <wsdl:message name="getSupportedCodeSystemsResponse">
      <wsdl:part name="parameters" element="impl:getSupportedCodeSystemsRespon
se"/>
   </wsdl:message>
   <wsdl:message name="getCTSVersionRequest">
      <wsdl:part name="parameters" element="impl:getCTSVersion"/>
   </wsdl:message>
   <wsdl:message name="lookupCodeSystemInfoResponse">
      <wsdl:part name="parameters" element="impl:lookupCodeSystemInfoResponse"/>
   </wsdl:message>
   <wsdl:message name="getCTSVersionResponse">
      <wsdl:part name="parameters" element="impl:getCTSVersionResponse"/>
   </wsdl:message>
   <wsdl:message name="getServiceVersionResponse">
      <wsdl:part name="parameters" element="impl:getServiceVersionResponse"/>
   </wsdl:message>
   <wsdl:message name="UnknownRelationshipCode">
      <wsdl:part name="fault" element="impl:fault7"/>
   </wsdl:message>
   <wsdl:message name="isConceptIdValidResponse">
      <wsdl:part name="parameters" element="impl:isConceptIdValidResponse"/>
   </wsdl:message>
   <wsdl:message name="CodeSystemNameIdMismatch">
      <wsdl:part name="fault" element="impl:fault3"/>
   </wsdl:message>
```

```
  <wsdl:message name="UnknownLanguageCode">
    <wsdl:part name="fault" element="impl:fault4"/>
  </wsdl:message>
  <wsdl:message name="isConceptIdValidRequest">
    <wsdl:part name="parameters" element="impl:isConceptIdValid"/>
  </wsdl:message>
  <wsdl:message name="lookupDesignationRequest">
    <wsdl:part name="parameters" element="impl:lookupDesignation"/>
  </wsdl:message>
  <wsdl:message name="getServiceDescriptionResponse">
    <wsdl:part name="parameters" element="impl:getServiceDescriptionResponse"/>
  </wsdl:message>
  <wsdl:message name="lookupCodeSystemInfoRequest">
    <wsdl:part name="parameters" element="impl:lookupCodeSystemInfo"/>
  </wsdl:message>
  <wsdl:message name="getSupportedCodeSystemsRequest">
    <wsdl:part name="parameters" element="impl:getSupportedCodeSystems"/>
  </wsdl:message>
  <wsdl:message name="UnknownConceptCode">
    <wsdl:part name="fault" element="impl:fault5"/>
  </wsdl:message>
  <wsdl:message name="UnexpectedError">
    <wsdl:part name="fault" element="impl:fault1"/>
  </wsdl:message>
  <wsdl:message name="UnknownCodeSystem">
    <wsdl:part name="fault" element="impl:fault2"/>
  </wsdl:message>
  <wsdl:message name="NoApplicableDesignationFound">
    <wsdl:part name="fault" element="impl:fault6"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameResponse">
    <wsdl:part name="parameters" element="impl:getServiceNameResponse"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameRequest">
    <wsdl:part name="parameters" element="impl:getServiceName"/>
  </wsdl:message>
  <wsdl:message name="getServiceDescriptionRequest">
    <wsdl:part name="parameters" element="impl:getServiceDescription"/>
  </wsdl:message>
  <wsdl:message name="areCodesRelatedResponse">
    <wsdl:part name="parameters" element="impl:areCodesRelatedResponse"/>
  </wsdl:message>
  <wsdl:portType name="RuntimeOperations">
    <wsdl:operation name="getSupportedCodeSystems">
      <wsdl:input name="getSupportedCodeSystemsRequest" message="impl:getSuppor
tedCodeSystemsRequest"/>
      <wsdl:output name="getSupportedCodeSystemsResponse" message="impl:getSupp
ortedCodeSystemsResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
    </wsdl:operation>
    <wsdl:operation name="lookupCodeSystemInfo">
      <wsdl:input name="lookupCodeSystemInfoRequest" message="impl:lookupCodeSy
stemInfoRequest"/>
      <wsdl:output name="lookupCodeSystemInfoResponse" message="impl:lookupCode
SystemInfoResponse"/>
```

```
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
        <wsdl:fault name="CodeSystemNameIdMismatch" message="impl:CodeSystemNameI
dMismatch"/>
        <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      </wsdl:operation>
      <wsdl:operation name="isConceptIdValid">
        <wsdl:input name="isConceptIdValidRequest" message="impl:isConceptIdValid
Request"/>
        <wsdl:output name="isConceptIdValidResponse" message="impl:isConceptIdVal
idResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
        <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      </wsdl:operation>
      <wsdl:operation name="lookupDesignation">
        <wsdl:input name="lookupDesignationRequest" message="impl:lookupDesignati
onRequest"/>
        <wsdl:output name="lookupDesignationResponse" message="impl:lookupDesigna
tionResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
        <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
        <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
        <wsdl:fault name="NoApplicableDesignationFound" message="impl:NoApplicabl
eDesignationFound"/>
        <wsdl:fault name="UnknownLanguageCode" message="impl:UnknownLanguageCo
de"/>
      </wsdl:operation>
      <wsdl:operation name="areCodesRelated">
        <wsdl:input name="areCodesRelatedRequest" message="impl:areCodesRelatedRe
quest"/>
        <wsdl:output name="areCodesRelatedResponse" message="impl:areCodesRelated
Response"/>
        <wsdl:fault name="UnknownRelationshipCode" message="impl:UnknownRelations
hipCode"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
        <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
        <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
        <wsdl:fault name="UnknownRelationQualifier" message="impl:UnknownRelationQ
ualifier"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceName">
        <wsdl:input name="getServiceNameRequest" message="impl:getServiceNameRequ
est"/>
        <wsdl:output name="getServiceNameResponse" message="impl:getServiceNameRe
sponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceVersion">
        <wsdl:input name="getServiceVersionRequest" message="impl:getServiceVersi
onRequest"/>
        <wsdl:output name="getServiceVersionResponse" message="impl:getServiceVer
sionResponse"/>
        <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      </wsdl:operation>
      <wsdl:operation name="getServiceDescription">
        <wsdl:input name="getServiceDescriptionRequest" message="impl:getServiceD
escriptionRequest"/>
```

```
      <wsdl:output name="getServiceDescriptionResponse" message="impl:getServic
eDescriptionResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getCTSVersion">
      <wsdl:input name="getCTSVersionRequest" message="impl:getCTSVersionReque
st"/>
      <wsdl:output name="getCTSVersionResponse" message="impl:getCTSVersionResp
onse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="VocabRuntimeServiceSoapBinding"
type="impl:RuntimeOperations">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/
soap/http"/>
    <wsdl:operation name="getSupportedCodeSystems">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSupportedCodeSystemsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSupportedCodeSystemsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TimeoutError">
        <wsdlsoap:fault name="TimeoutError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="lookupCodeSystemInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="lookupCodeSystemInfoRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="lookupCodeSystemInfoResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="CodeSystemNameIdMismatch">
        <wsdlsoap:fault name="CodeSystemNameIdMismatch" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownCodeSystem">
        <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="isConceptIdValid">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="isConceptIdValidRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="isConceptIdValidResponse">
        <wsdlsoap:body use="literal"/>
```

```
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupDesignation">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupDesignationRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="lookupDesignationResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NoApplicableDesignationFound">
      <wsdlsoap:fault name="NoApplicableDesignationFound" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguageCode">
      <wsdlsoap:fault name="UnknownLanguageCode" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="areCodesRelated">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="areCodesRelatedRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="areCodesRelatedResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnknownRelationshipCode">
      <wsdlsoap:fault name="UnknownRelationshipCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownRelationQualifier">
      <wsdlsoap:fault name="UnknownRelationQualifier" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
```

```
    <wsdl:operation name="getServiceName">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceNameRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceNameResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceDescription">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceDescriptionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceDescriptionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getCTSVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getCTSVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getCTSVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="RuntimeOperationsService">
    <wsdl:port name="VocabRuntimeService" binding="impl:VocabRuntimeServiceSoap
Binding">
      <wsdlsoap:address location="http://localhost:8080/axis/services/
VocabRuntimeService"/>
    </wsdl:port>
  </wsdl:service>
```

```
  <!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->
</wsdl:definitions>
```

### 17.4 Описание API обозревателя на уровне словаря

Ниже приведено описание службы обозревателя VocabBrowser на языке WSDL (файл VocabBrowser.wsdl):

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn://hl7.org/CTSVAPI" xmlns:intf="urn://hl7.org/CTSVAPI"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://
schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn://hl7.org/CTSVAPI">
  <wsdl:types>
    <schema elementFormDefault="qualified" targetNamespace="urn://hl7.org/
CTSVAPI" xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="getSupportedMatchAlgorithms">
        <complexType/>
      </element>
      <element name="getSupportedMatchAlgorithmsResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" name="getSupportedMatchAlgorithmsRe
turn" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="UnexpectedError">
        <sequence>
          <element name="possible_cause" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault" type="impl:UnexpectedError"/>
      <element name="getSupportedCodeSystems">
        <complexType>
          <sequence>
            <element name="in0" type="xsd:int"/>
            <element name="in1" type="xsd:int"/>
          </sequence>
        </complexType>
      </element>
      <element name="getSupportedCodeSystemsResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" name="getSupportedCodeSystemsRetu
rn" type="impl:CodeSystemIdAndVersions"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="ArrayOf_xsd_string">
        <sequence>
          <element maxOccurs="unbounded" minOccurs="0" name="item"
type="xsd:string"/>
        </sequence>
      </complexType>
```

```
      <complexType name="CodeSystemIdAndVersions">
        <sequence>
          <element name="codeSystem_id" nillable="true" type="xsd:string"/>
          <element name="codeSystem_name" nillable="true" type="xsd:string"/>
          <element name="copyright" nillable="true" type="xsd:string"/>
          <element name="codeSystem_versions" nillable="true"
type="impl:ArrayOf_xsd_string"/>
        </sequence>
      </complexType>
      <complexType name="TimeoutError">
        <sequence/>
      </complexType>
      <element name="fault1" type="impl:TimeoutError"/>
      <element name="lookupConceptCodesByDesignation">
        <complexType>
          <sequence>
            <element name="in0" type="xsd:string"/>
            <element name="in1" type="xsd:string"/>
            <element name="in2" type="xsd:string"/>
            <element name="in3" type="xsd:string"/>
            <element name="in4" type="xsd:boolean"/>
            <element name="in5" type="xsd:int"/>
            <element name="in6" type="xsd:int"/>
          </sequence>
        </complexType>
      </element>
      <element name="lookupConceptCodesByDesignationResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" name="lookupConceptCodesByDesignati
onReturn" type="impl:ConceptId"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="ConceptId">
        <sequence>
          <element name="codeSystem_id" nillable="true" type="xsd:string"/>
          <element name="concept_code" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="UnknownCodeSystem">
        <sequence>
          <element name="codeSystem_id" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault2" type="impl:UnknownCodeSystem"/>
      <complexType name="BadlyFormedMatchText">
        <sequence>
          <element name="matchText" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault3" type="impl:BadlyFormedMatchText"/>
      <complexType name="UnknownMatchAlgorithm">
        <sequence>
          <element name="matchAlgorithm_code" nillable="true"
type="xsd:string"/>
```

```
        </sequence>
      </complexType>
      <element name="fault4" type="impl:UnknownMatchAlgorithm"/>
      <complexType name="UnknownLanguageCode">
        <sequence>
          <element name="language_code" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault5" type="impl:UnknownLanguageCode"/>
      <element name="lookupConceptCodesByProperty">
        <complexType>
          <sequence>
            <element name="in0" type="xsd:string"/>
            <element name="in1" type="xsd:string"/>
            <element name="in2" type="xsd:string"/>
            <element name="in3" type="xsd:string"/>
            <element name="in4" type="xsd:boolean"/>
            <element maxOccurs="unbounded" name="in5" type="xsd:string"/>
            <element maxOccurs="unbounded" name="in6" type="xsd:string"/>
            <element name="in7" type="xsd:int"/>
            <element name="in8" type="xsd:int"/>
          </sequence>
        </complexType>
      </element>
      <element name="lookupConceptCodesByPropertyResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" name="lookupConceptCodesByPropertyR
eturn" type="impl:ConceptId"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="UnknownPropertyCode">
        <sequence>
          <element name="property_code" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault6" type="impl:UnknownPropertyCode"/>
      <complexType name="UnknownMimeTypeCode">
        <sequence>
          <element name="mimeType_code" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault7" type="impl:UnknownMimeTypeCode"/>
      <element name="lookupCompleteCodedConcept">
        <complexType>
          <sequence>
            <element name="in0" type="impl:ConceptId"/>
          </sequence>
        </complexType>
      </element>
      <element name="lookupCompleteCodedConceptResponse">
        <complexType>
          <sequence>
            <element name="lookupCompleteCodedConceptReturn" type="impl:Comple
teCodedConceptDescription"/>
```

```
            </sequence>
          </complexType>
        </element>
        <complexType name="ConceptDesignation">
          <sequence>
            <element name="designation" nillable="true" type="xsd:string"/>
            <element name="language_code" nillable="true" type="xsd:string"/>
            <element name="preferredForLanguage" type="xsd:boolean"/>
          </sequence>
        </complexType>
        <complexType name="ArrayOfConceptDesignation">
          <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:
ConceptDesignation"/>
          </sequence>
        </complexType>
        <complexType name="ConceptProperty">
          <sequence>
            <element name="property_code" nillable="true" type="xsd:string"/>
            <element name="propertyValue" nillable="true" type="xsd:string"/>
            <element name="language_code" nillable="true" type="xsd:string"/>
            <element name="mimeType_code" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <complexType name="ArrayOfConceptProperty">
          <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="item"
type="impl:ConceptProperty"/>
          </sequence>
        </complexType>
        <complexType name="ConceptRelationship">
          <sequence>
            <element name="sourceConcept_id" nillable="true"
type="impl:ConceptId"/>
            <element name="relationship_code" nillable="true"
type="xsd:string"/>
            <element name="relationQualifiers" nillable="true"
type="impl:ArrayOf_xsd_string"/>
            <element name="targetConcept_id" nillable="true"
type="impl:ConceptId"/>
          </sequence>
        </complexType>
        <complexType name="ArrayOfConceptRelationship">
          <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="item" type="impl:
ConceptRelationship"/>
          </sequence>
        </complexType>
        <complexType name="CompleteCodedConceptDescription">
          <sequence>
            <element name="concept_id" nillable="true" type="impl:ConceptId"/>
            <element name="conceptStatus_code" nillable="true"
type="xsd:string"/>
            <element name="codeSystem_version" nillable="true"
type="xsd:string"/>
            <element name="designatedBy" nillable="true" type="impl:ArrayOfConce
ptDesignation"/>
```

```
            <element name="hasProperties" nillable="true" type="impl:ArrayOfConc
eptProperty"/>
            <element name="sourceFor" nillable="true" type="impl:ArrayOfConceptR
elationship"/>
            <element name="targetOf" nillable="true" type="impl:ArrayOfConceptRe
lationship"/>
          </sequence>
        </complexType>
        <complexType name="UnknownConceptCode">
          <sequence>
            <element name="concept_code" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="fault8" type="impl:UnknownConceptCode"/>
        <element name="lookupDesignations">
          <complexType>
            <sequence>
              <element name="in0" type="impl:ConceptId"/>
              <element name="in1" type="xsd:string"/>
              <element name="in2" type="xsd:string"/>
              <element name="in3" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
        <element name="lookupDesignationsResponse">
          <complexType>
            <sequence>
              <element maxOccurs="unbounded" name="lookupDesignationsReturn" typ
e="impl:ConceptDesignation"/>
            </sequence>
          </complexType>
        </element>
        <element name="lookupProperties">
          <complexType>
            <sequence>
              <element name="in0" type="impl:ConceptId"/>
              <element maxOccurs="unbounded" name="in1" type="xsd:string"/>
              <element name="in2" type="xsd:string"/>
              <element name="in3" type="xsd:string"/>
              <element name="in4" type="xsd:string"/>
              <element maxOccurs="unbounded" name="in5" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
        <element name="lookupPropertiesResponse">
          <complexType>
            <sequence>
              <element maxOccurs="unbounded" name="lookupPropertiesReturn"
type="impl:ConceptProperty"/>
            </sequence>
          </complexType>
        </element>
        <element name="lookupCodeExpansion">
          <complexType>
            <sequence>
              <element name="in0" type="impl:ConceptId"/>
```

```
            <element name="in1" type="xsd:string"/>
            <element name="in2" type="xsd:boolean"/>
            <element name="in3" type="xsd:boolean"/>
            <element name="in4" type="xsd:string"/>
            <element name="in5" type="xsd:int"/>
            <element name="in6" type="xsd:int"/>
          </sequence>
        </complexType>
      </element>
      <element name="lookupCodeExpansionResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" name="lookupCodeExpansionReturn"
type="impl:RelatedCode"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="RelatedCode">
        <sequence>
          <element name="pathLength" type="xsd:short"/>
          <element name="concept_code" nillable="true" type="xsd:string"/>
          <element name="designation" nillable="true" type="xsd:string"/>
          <element name="relationQualifiers" nillable="true"
type="impl:ArrayOf_xsd_string"/>
          <element name="canExpand" type="xsd:boolean"/>
          <element name="expansionContext" type="xsd:base64Binary"/>
        </sequence>
      </complexType>
      <complexType name="UnknownRelationshipCode">
        <sequence>
          <element name="relationship_code" nillable="true"
type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault9" type="impl:UnknownRelationshipCode"/>
      <element name="expandCodeExpansionContext">
        <complexType>
          <sequence>
            <element name="in0" type="xsd:base64Binary"/>
          </sequence>
        </complexType>
      </element>
      <element name="expandCodeExpansionContextResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" name="expandCodeExpansionContextRet
urn" type="impl:RelatedCode"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="InvalidExpansionContext">
        <sequence/>
      </complexType>
      <element name="fault10" type="impl:InvalidExpansionContext"/>
      <element name="getServiceName">
        <complexType/>
```

```
        </element>
        <element name="getServiceNameResponse">
          <complexType>
            <sequence>
              <element name="getServiceNameReturn" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
        <element name="getServiceVersion">
          <complexType/>
        </element>
        <element name="getServiceVersionResponse">
          <complexType>
            <sequence>
              <element name="getServiceVersionReturn" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
        <element name="getServiceDescription">
          <complexType/>
        </element>
        <element name="getServiceDescriptionResponse">
          <complexType>
            <sequence>
              <element name="getServiceDescriptionReturn" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
        <element name="getCTSVersion">
          <complexType/>
        </element>
        <element name="getCTSVersionResponse">
          <complexType>
            <sequence>
              <element name="getCTSVersionReturn" type="impl:CTSVersionId"/>
            </sequence>
          </complexType>
        </element>
        <complexType name="CTSVersionId">
          <sequence>
            <element name="major" type="xsd:short"/>
            <element name="minor" type="xsd:short"/>
          </sequence>
        </complexType>
      </schema>
    </wsdl:types>
    <wsdl:message name="expandCodeExpansionContextResponse">
      <wsdl:part name="parameters" element="impl:expandCodeExpansionContextRespon
se"/>
    </wsdl:message>
    <wsdl:message name="getSupportedMatchAlgorithmsRequest">
      <wsdl:part name="parameters" element="impl:getSupportedMatchAlgorithms"/>
    </wsdl:message>
    <wsdl:message name="lookupDesignationsRequest">
      <wsdl:part name="parameters" element="impl:lookupDesignations"/>
    </wsdl:message>
```

```
  <wsdl:message name="getServiceVersionRequest">
    <wsdl:part name="parameters" element="impl:getServiceVersion"/>
  </wsdl:message>
  <wsdl:message name="lookupConceptCodesByDesignationResponse">
    <wsdl:part name="parameters" element="impl:lookupConceptCodesByDesignationR
esponse"/>
  </wsdl:message>
  <wsdl:message name="TimeoutError">
    <wsdl:part name="fault" element="impl:fault1"/>
  </wsdl:message>
  <wsdl:message name="getSupportedCodeSystemsResponse">
    <wsdl:part name="parameters" element="impl:getSupportedCodeSystemsResponse"/>
  </wsdl:message>
  <wsdl:message name="getCTSVersionRequest">
    <wsdl:part name="parameters" element="impl:getCTSVersion"/>
  </wsdl:message>
  <wsdl:message name="InvalidExpansionContext">
    <wsdl:part name="fault" element="impl:fault10"/>
  </wsdl:message>
  <wsdl:message name="getSupportedMatchAlgorithmsResponse">
    <wsdl:part name="parameters" element="impl:getSupportedMatchAlgorithmsRespo
nse"/>
  </wsdl:message>
  <wsdl:message name="UnknownPropertyCode">
    <wsdl:part name="fault" element="impl:fault6"/>
  </wsdl:message>
  <wsdl:message name="lookupCompleteCodedConceptResponse">
    <wsdl:part name="parameters" element="impl:lookupCompleteCodedConceptRespon
se"/>
  </wsdl:message>
  <wsdl:message name="getCTSVersionResponse">
    <wsdl:part name="parameters" element="impl:getCTSVersionResponse"/>
  </wsdl:message>
  <wsdl:message name="lookupDesignationsResponse">
    <wsdl:part name="parameters" element="impl:lookupDesignationsResponse"/>
  </wsdl:message>
  <wsdl:message name="getServiceVersionResponse">
    <wsdl:part name="parameters" element="impl:getServiceVersionResponse"/>
  </wsdl:message>
  <wsdl:message name="lookupCodeExpansionRequest">
    <wsdl:part name="parameters" element="impl:lookupCodeExpansion"/>
  </wsdl:message>
  <wsdl:message name="UnknownRelationshipCode">
    <wsdl:part name="fault" element="impl:fault9"/>
  </wsdl:message>
  <wsdl:message name="lookupConceptCodesByDesignationRequest">
    <wsdl:part name="parameters" element="impl:lookupConceptCodesByDesignati
on"/>
  </wsdl:message>
  <wsdl:message name="UnknownMatchAlgorithm">
    <wsdl:part name="fault" element="impl:fault4"/>
  </wsdl:message>
  <wsdl:message name="UnknownLanguageCode">
    <wsdl:part name="fault" element="impl:fault5"/>
  </wsdl:message>
  <wsdl:message name="lookupConceptCodesByPropertyRequest">
```

```
    <wsdl:part name="parameters" element="impl:lookupConceptCodesByProperty"/>
  </wsdl:message>
  <wsdl:message name="lookupPropertiesResponse">
    <wsdl:part name="parameters" element="impl:lookupPropertiesResponse"/>
  </wsdl:message>
  <wsdl:message name="expandCodeExpansionContextRequest">
    <wsdl:part name="parameters" element="impl:expandCodeExpansionContext"/>
  </wsdl:message>
  <wsdl:message name="getServiceDescriptionResponse">
    <wsdl:part name="parameters" element="impl:getServiceDescriptionResponse"/>
  </wsdl:message>
  <wsdl:message name="lookupCodeExpansionResponse">
    <wsdl:part name="parameters" element="impl:lookupCodeExpansionResponse"/>
  </wsdl:message>
  <wsdl:message name="getSupportedCodeSystemsRequest">
    <wsdl:part name="parameters" element="impl:getSupportedCodeSystems"/>
  </wsdl:message>
  <wsdl:message name="UnknownConceptCode">
    <wsdl:part name="fault" element="impl:fault8"/>
  </wsdl:message>
  <wsdl:message name="UnexpectedError">
    <wsdl:part name="fault" element="impl:fault"/>
  </wsdl:message>
  <wsdl:message name="UnknownCodeSystem">
    <wsdl:part name="fault" element="impl:fault2"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameResponse">
    <wsdl:part name="parameters" element="impl:getServiceNameResponse"/>
  </wsdl:message>
  <wsdl:message name="lookupCompleteCodedConceptRequest">
    <wsdl:part name="parameters" element="impl:lookupCompleteCodedConcept"/>
  </wsdl:message>
  <wsdl:message name="lookupPropertiesRequest">
    <wsdl:part name="parameters" element="impl:lookupProperties"/>
  </wsdl:message>
  <wsdl:message name="getServiceDescriptionRequest">
    <wsdl:part name="parameters" element="impl:getServiceDescription"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameRequest">
    <wsdl:part name="parameters" element="impl:getServiceName"/>
  </wsdl:message>
  <wsdl:message name="lookupConceptCodesByPropertyResponse">
    <wsdl:part name="parameters" element="impl:lookupConceptCodesByPropertyResp
onse"/>
  </wsdl:message>
  <wsdl:message name="BadlyFormedMatchText">
    <wsdl:part name="fault" element="impl:fault3"/>
  </wsdl:message>
  <wsdl:message name="UnknownMimeTypeCode">
    <wsdl:part name="fault" element="impl:fault7"/>
  </wsdl:message>
  <wsdl:portType name="BrowserOperations">
    <wsdl:operation name="getSupportedMatchAlgorithms">
      <wsdl:input name="getSupportedMatchAlgorithmsRequest" message="impl:getSu
pportedMatchAlgorithmsRequest"/>
      <wsdl:output name="getSupportedMatchAlgorithmsResponse" message="impl:get
SupportedMatchAlgorithmsResponse"/>
```

```
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getSupportedCodeSystems">
      <wsdl:input name="getSupportedCodeSystemsRequest" message="impl:getSuppor
tedCodeSystemsRequest"/>
      <wsdl:output name="getSupportedCodeSystemsResponse" message="impl:getSupp
ortedCodeSystemsResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
    </wsdl:operation>
    <wsdl:operation name="lookupConceptCodesByDesignation">
      <wsdl:input name="lookupConceptCodesByDesignationRequest" message="impl:l
ookupConceptCodesByDesignationRequest"/>
      <wsdl:output name="lookupConceptCodesByDesignationResponse" message="impl
:lookupConceptCodesByDesignationResponse"/>
      <wsdl:fault name="UnknownMatchAlgorithm" message="impl:UnknownMatchAlgori
thm"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="BadlyFormedMatchText" message="impl:BadlyFormedMatchTe
xt"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      <wsdl:fault name="UnknownLanguageCode" message="impl:UnknownLanguageCo
de"/>
    </wsdl:operation>
    <wsdl:operation name="lookupConceptCodesByProperty">
      <wsdl:input name="lookupConceptCodesByPropertyRequest" message="impl:look
upConceptCodesByPropertyRequest"/>
      <wsdl:output name="lookupConceptCodesByPropertyResponse" message="impl:lo
okupConceptCodesByPropertyResponse"/>
      <wsdl:fault name="UnknownMatchAlgorithm" message="impl:UnknownMatchAlgori
thm"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownMimeTypeCode" message="impl:UnknownMimeTypeCo
de"/>
      <wsdl:fault name="BadlyFormedMatchText" message="impl:BadlyFormedMatchTe
xt"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      <wsdl:fault name="UnknownPropertyCode" message="impl:UnknownPropertyCo
de"/>
      <wsdl:fault name="UnknownLanguageCode" message="impl:UnknownLanguageCo
de"/>
    </wsdl:operation>
    <wsdl:operation name="lookupCompleteCodedConcept">
      <wsdl:input name="lookupCompleteCodedConceptRequest" message="impl:lookup
CompleteCodedConceptRequest"/>
      <wsdl:output name="lookupCompleteCodedConceptResponse" message="impl:look
upCompleteCodedConceptResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
    </wsdl:operation>
    <wsdl:operation name="lookupDesignations">
      <wsdl:input name="lookupDesignationsRequest" message="impl:lookupDesignat
ionsRequest"/>
```

```
      <wsdl:output name="lookupDesignationsResponse" message="impl:lookupDesign
ationsResponse"/>
      <wsdl:fault name="UnknownMatchAlgorithm" message="impl:UnknownMatchAlgori
thm"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="BadlyFormedMatchText" message="impl:BadlyFormedMatchTe
xt"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      <wsdl:fault name="UnknownLanguageCode" message="impl:UnknownLanguageCo
de"/>
    </wsdl:operation>
    <wsdl:operation name="lookupProperties">
      <wsdl:input name="lookupPropertiesRequest" message="impl:lookupProperties
Request"/>
      <wsdl:output name="lookupPropertiesResponse" message="impl:lookupProperti
esResponse"/>
      <wsdl:fault name="UnknownMatchAlgorithm" message="impl:UnknownMatchAlgori
thm"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownMimeTypeCode" message="impl:UnknownMimeTypeCo
de"/>
      <wsdl:fault name="BadlyFormedMatchText" message="impl:BadlyFormedMatchTe
xt"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      <wsdl:fault name="UnknownPropertyCode" message="impl:UnknownPropertyCo
de"/>
      <wsdl:fault name="UnknownLanguageCode" message="impl:UnknownLanguageCo
de"/>
    </wsdl:operation>
    <wsdl:operation name="lookupCodeExpansion">
      <wsdl:input name="lookupCodeExpansionRequest" message="impl:lookupCodeExp
ansionRequest"/>
      <wsdl:output name="lookupCodeExpansionResponse" message="impl:lookupCodeE
xpansionResponse"/>
      <wsdl:fault name="UnknownRelationshipCode" message="impl:UnknownRelations
hipCode"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
      <wsdl:fault name="UnknownLanguageCode" message="impl:UnknownLanguageCo
de"/>
    </wsdl:operation>
    <wsdl:operation name="expandCodeExpansionContext">
      <wsdl:input name="expandCodeExpansionContextRequest" message="impl:expand
CodeExpansionContextRequest"/>
      <wsdl:output name="expandCodeExpansionContextResponse" message="impl:expa
ndCodeExpansionContextResponse"/>
      <wsdl:fault name="InvalidExpansionContext" message="impl:InvalidExpansion
Context"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="TimeoutError" message="impl:TimeoutError"/>
    </wsdl:operation>
    <wsdl:operation name="getServiceName">
```

```
      <wsdl:input name="getServiceNameRequest" message="impl:getServiceNameRequ
est"/>
      <wsdl:output name="getServiceNameResponse" message="impl:getServiceNameRe
sponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getServiceVersion">
      <wsdl:input name="getServiceVersionRequest" message="impl:getServiceVersi
onRequest"/>
      <wsdl:output name="getServiceVersionResponse" message="impl:getServiceVer
sionResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getServiceDescription">
      <wsdl:input name="getServiceDescriptionRequest" message="impl:getServiceD
escriptionRequest"/>
      <wsdl:output name="getServiceDescriptionResponse" message="impl:getServic
eDescriptionResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getCTSVersion">
      <wsdl:input name="getCTSVersionRequest" message="impl:getCTSVersionReque
st"/>
      <wsdl:output name="getCTSVersionResponse" message="impl:getCTSVersionResp
onse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="VocabBrowserServiceSoapBinding"
type="impl:BrowserOperations">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/
soap/http"/>
    <wsdl:operation name="getSupportedMatchAlgorithms">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSupportedMatchAlgorithmsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSupportedMatchAlgorithmsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getSupportedCodeSystems">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getSupportedCodeSystemsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSupportedCodeSystemsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="TimeoutError">
```

```
      <wsdlsoap:fault name="TimeoutError" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupConceptCodesByDesignation">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupConceptCodesByDesignationRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="lookupConceptCodesByDesignationResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnknownMatchAlgorithm">
      <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="BadlyFormedMatchText">
      <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="TimeoutError">
      <wsdlsoap:fault name="TimeoutError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguageCode">
      <wsdlsoap:fault name="UnknownLanguageCode" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupConceptCodesByProperty">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupConceptCodesByPropertyRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="lookupConceptCodesByPropertyResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnknownMatchAlgorithm">
      <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownMimeTypeCode">
      <wsdlsoap:fault name="UnknownMimeTypeCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="BadlyFormedMatchText">
      <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="TimeoutError">
      <wsdlsoap:fault name="TimeoutError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
```

```
    <wsdl:fault name="UnknownPropertyCode">
      <wsdlsoap:fault name="UnknownPropertyCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguageCode">
      <wsdlsoap:fault name="UnknownLanguageCode" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupCompleteCodedConcept">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupCompleteCodedConceptRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="lookupCompleteCodedConceptResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupDesignations">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupDesignationsRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="lookupDesignationsResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnknownMatchAlgorithm">
      <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="BadlyFormedMatchText">
      <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguageCode">
      <wsdlsoap:fault name="UnknownLanguageCode" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupProperties">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupPropertiesRequest">
      <wsdlsoap:body use="literal"/>
```

```
    </wsdl:input>
    <wsdl:output name="lookupPropertiesResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnknownMatchAlgorithm">
      <wsdlsoap:fault name="UnknownMatchAlgorithm" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownMimeTypeCode">
      <wsdlsoap:fault name="UnknownMimeTypeCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="BadlyFormedMatchText">
      <wsdlsoap:fault name="BadlyFormedMatchText" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownPropertyCode">
      <wsdlsoap:fault name="UnknownPropertyCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguageCode">
      <wsdlsoap:fault name="UnknownLanguageCode" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="lookupCodeExpansion">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="lookupCodeExpansionRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="lookupCodeExpansionResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="UnknownRelationshipCode">
      <wsdlsoap:fault name="UnknownRelationshipCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnexpectedError">
      <wsdlsoap:fault name="UnexpectedError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownConceptCode">
      <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="TimeoutError">
      <wsdlsoap:fault name="TimeoutError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownCodeSystem">
      <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnknownLanguageCode">
      <wsdlsoap:fault name="UnknownLanguageCode" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
```

```
<wsdl:operation name="expandCodeExpansionContext">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="expandCodeExpansionContextRequest">
    <wsdlsoap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="expandCodeExpansionContextResponse">
    <wsdlsoap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="InvalidExpansionContext">
    <wsdlsoap:fault name="InvalidExpansionContext" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="UnexpectedError">
    <wsdlsoap:fault name="UnexpectedError" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="TimeoutError">
    <wsdlsoap:fault name="TimeoutError" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getServiceName">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="getServiceNameRequest">
    <wsdlsoap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getServiceNameResponse">
    <wsdlsoap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="UnexpectedError">
    <wsdlsoap:fault name="UnexpectedError" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getServiceVersion">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="getServiceVersionRequest">
    <wsdlsoap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getServiceVersionResponse">
    <wsdlsoap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="UnexpectedError">
    <wsdlsoap:fault name="UnexpectedError" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getServiceDescription">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="getServiceDescriptionRequest">
    <wsdlsoap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getServiceDescriptionResponse">
    <wsdlsoap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="UnexpectedError">
    <wsdlsoap:fault name="UnexpectedError" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getCTSVersion">
```

```
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getCTSVersionRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getCTSVersionResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
        <wsdl:fault name="UnexpectedError">
          <wsdlsoap:fault name="UnexpectedError" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="BrowserOperationsService">
      <wsdl:port name="VocabBrowserService" binding="impl:VocabBrowserServiceSoap
Binding">
        <wsdlsoap:address location="http://localhost:8080/axis/services/
VocabBrowserService"/>
      </wsdl:port>
    </wsdl:service>
    <!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->
</wsdl:definitions>
```

### 17.5 Описание API отображения кодов

Ниже приведено описание службы обозревателя CodeMapping на языке WSDL (файл CodeMapping.wsdl):

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn://hl7.org/CTSVAPI" xmlns:intf="urn://hl7.org/CTSVAPI"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://
schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn://hl7.org/CTSVAPI">
  <wsdl:types>
    <schema elementFormDefault="qualified" targetNamespace="urn://hl7.org/
CTSVAPI" xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="getSupportedMaps">
        <complexType/>
      </element>
      <element name="getSupportedMapsResponse">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" name="getSupportedMapsReturn"
type="impl:CodeMap"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="CodeMap">
        <sequence>
          <element name="map_name" nillable="true" type="xsd:string"/>
          <element name="mapDescription" nillable="true" type="xsd:string"/>
          <element name="fromCodeSystem_id" nillable="true"
type="xsd:string"/>
          <element name="fromCodeSystem_name" nillable="true"
type="xsd:string"/>
```

```
            <element name="fromCodeSystem_version" nillable="true"
type="xsd:string"/>
            <element name="toCodeSystem_id" nillable="true" type="xsd:string"/>
            <element name="toCodeSystem_name" nillable="true"
type="xsd:string"/>
            <element name="toCodeSystem_version" nillable="true"
type="xsd:string"/>
          </sequence>
        </complexType>
        <complexType name="UnexpectedError">
          <sequence>
            <element name="possible_cause" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="fault" type="impl:UnexpectedError"/>
        <element name="mapConceptCode">
          <complexType>
            <sequence>
              <element name="in0" type="impl:ConceptId"/>
              <element name="in1" type="xsd:string"/>
              <element name="in2" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
        <complexType name="ConceptId">
          <sequence>
            <element name="codeSystem_id" nillable="true" type="xsd:string"/>
            <element name="concept_code" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="mapConceptCodeResponse">
          <complexType>
            <sequence>
              <element name="mapConceptCodeReturn"
type="impl:MappedConceptCode"/>
            </sequence>
          </complexType>
        </element>
        <complexType name="MappedConceptCode">
          <sequence>
            <element name="mappedConcept_id" nillable="true"
type="impl:ConceptId"/>
            <element name="mapQuality_code" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <complexType name="UnknownCodeSystem">
          <sequence>
            <element name="codeSystem_id" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="fault1" type="impl:UnknownCodeSystem"/>
        <complexType name="UnknownConceptCode">
          <sequence>
            <element name="concept_code" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
```

```
        <element name="fault2" type="impl:UnknownConceptCode"/>
        <complexType name="MappingNotAvailable">
          <sequence>
            <element name="fromCodeSystem_id" nillable="true"
type="xsd:string"/>
            <element name="toCodeSystem_id" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="fault3" type="impl:MappingNotAvailable"/>
        <complexType name="UnknownMapName">
          <sequence>
            <element name="map_name" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="fault4" type="impl:UnknownMapName"/>
        <complexType name="ArrayOf_xsd_string">
          <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="item"
type="xsd:string"/>
          </sequence>
        </complexType>
        <complexType name="AmbiguousMapRequest">
          <sequence>
            <element name="possible_maps" nillable="true" type="impl:ArrayOf_
xsd_string"/>
          </sequence>
        </complexType>
        <element name="fault5" type="impl:AmbiguousMapRequest"/>
        <complexType name="MapNameSourceMismatch">
          <sequence>
            <element name="fromCodeSystem_id" nillable="true"
type="xsd:string"/>
            <element name="mapSourceCodeSystem_id" nillable="true"
type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="fault6" type="impl:MapNameSourceMismatch"/>
        <complexType name="MapNameTargetMismatch">
          <sequence>
            <element name="toCodeSystem_id" nillable="true" type="xsd:string"/>
            <element name="mapTargetCodeSystem_id" nillable="true"
type="xsd:string"/>
          </sequence>
        </complexType>
        <element name="fault7" type="impl:MapNameTargetMismatch"/>
        <complexType name="UnableToMap">
          <sequence/>
        </complexType>
        <element name="fault8" type="impl:UnableToMap"/>
        <element name="getServiceName">
          <complexType/>
        </element>
        <element name="getServiceNameResponse">
          <complexType>
            <sequence>
              <element name="getServiceNameReturn" type="xsd:string"/>
```

```
        </sequence>
      </complexType>
    </element>
    <element name="getServiceVersion">
      <complexType/>
    </element>
    <element name="getServiceVersionResponse">
      <complexType>
        <sequence>
          <element name="getServiceVersionReturn" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="getServiceDescription">
      <complexType/>
    </element>
    <element name="getServiceDescriptionResponse">
      <complexType>
        <sequence>
          <element name="getServiceDescriptionReturn" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="getCTSVersion">
      <complexType/>
    </element>
    <element name="getCTSVersionResponse">
      <complexType>
        <sequence>
          <element name="getCTSVersionReturn" type="impl:CTSVersionId"/>
        </sequence>
      </complexType>
    </element>
    <complexType name="CTSVersionId">
      <sequence>
        <element name="major" type="xsd:short"/>
        <element name="minor" type="xsd:short"/>
      </sequence>
    </complexType>
  </schema>
</wsdl:types>
<wsdl:message name="UnknownMapName">
  <wsdl:part name="fault" element="impl:fault4"/>
</wsdl:message>
<wsdl:message name="MapNameTargetMismatch">
  <wsdl:part name="fault" element="impl:fault7"/>
</wsdl:message>
<wsdl:message name="AmbiguousMapRequest">
  <wsdl:part name="fault" element="impl:fault5"/>
</wsdl:message>
<wsdl:message name="getServiceDescriptionResponse">
  <wsdl:part name="parameters" element="impl:getServiceDescriptionResponse"/>
</wsdl:message>
<wsdl:message name="MappingNotAvailable">
  <wsdl:part name="fault" element="impl:fault3"/>
</wsdl:message>
```

```
  <wsdl:message name="getServiceVersionRequest">
    <wsdl:part name="parameters" element="impl:getServiceVersion"/>
  </wsdl:message>
  <wsdl:message name="getSupportedMapsResponse">
    <wsdl:part name="parameters" element="impl:getSupportedMapsResponse"/>
  </wsdl:message>
  <wsdl:message name="MapNameSourceMismatch">
    <wsdl:part name="fault" element="impl:fault6"/>
  </wsdl:message>
  <wsdl:message name="UnknownConceptCode">
    <wsdl:part name="fault" element="impl:fault2"/>
  </wsdl:message>
  <wsdl:message name="UnexpectedError">
    <wsdl:part name="fault" element="impl:fault"/>
  </wsdl:message>
  <wsdl:message name="UnknownCodeSystem">
    <wsdl:part name="fault" element="impl:fault1"/>
  </wsdl:message>
  <wsdl:message name="UnableToMap">
    <wsdl:part name="fault" element="impl:fault8"/>
  </wsdl:message>
  <wsdl:message name="getCTSVersionRequest">
    <wsdl:part name="parameters" element="impl:getCTSVersion"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameResponse">
    <wsdl:part name="parameters" element="impl:getServiceNameResponse"/>
  </wsdl:message>
  <wsdl:message name="getServiceDescriptionRequest">
    <wsdl:part name="parameters" element="impl:getServiceDescription"/>
  </wsdl:message>
  <wsdl:message name="getServiceNameRequest">
    <wsdl:part name="parameters" element="impl:getServiceName"/>
  </wsdl:message>
  <wsdl:message name="mapConceptCodeRequest">
    <wsdl:part name="parameters" element="impl:mapConceptCode"/>
  </wsdl:message>
  <wsdl:message name="mapConceptCodeResponse">
    <wsdl:part name="parameters" element="impl:mapConceptCodeResponse"/>
  </wsdl:message>
  <wsdl:message name="getCTSVersionResponse">
    <wsdl:part name="parameters" element="impl:getCTSVersionResponse"/>
  </wsdl:message>
  <wsdl:message name="getSupportedMapsRequest">
    <wsdl:part name="parameters" element="impl:getSupportedMaps"/>
  </wsdl:message>
  <wsdl:message name="getServiceVersionResponse">
    <wsdl:part name="parameters" element="impl:getServiceVersionResponse"/>
  </wsdl:message>
  <wsdl:portType name="CodeMappingOperations">
    <wsdl:operation name="getSupportedMaps">
      <wsdl:input name="getSupportedMapsRequest" message="impl:getSupportedMaps
Request"/>
      <wsdl:output name="getSupportedMapsResponse" message="impl:getSupportedMa
psResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
```

```
    </wsdl:operation>
    <wsdl:operation name="mapConceptCode">
      <wsdl:input name="mapConceptCodeRequest" message="impl:mapConceptCodeRequ
est"/>
      <wsdl:output name="mapConceptCodeResponse" message="impl:mapConceptCodeRe
sponse"/>
      <wsdl:fault name="AmbiguousMapRequest" message="impl:AmbiguousMapReque
st"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
      <wsdl:fault name="MapNameSourceMismatch" message="impl:MapNameSourceMisma
tch"/>
      <wsdl:fault name="MapNameTargetMismatch" message="impl:MapNameTargetMisma
tch"/>
      <wsdl:fault name="UnableToMap" message="impl:UnableToMap"/>
      <wsdl:fault name="MappingNotAvailable" message="impl:MappingNotAvailab
le"/>
      <wsdl:fault name="UnknownMapName" message="impl:UnknownMapName"/>
      <wsdl:fault name="UnknownConceptCode" message="impl:UnknownConceptCode"/>
      <wsdl:fault name="UnknownCodeSystem" message="impl:UnknownCodeSystem"/>
    </wsdl:operation>
    <wsdl:operation name="getServiceName">
      <wsdl:input name="getServiceNameRequest" message="impl:getServiceNameRequ
est"/>
      <wsdl:output name="getServiceNameResponse" message="impl:getServiceNameRe
sponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getServiceVersion">
      <wsdl:input name="getServiceVersionRequest" message="impl:getServiceVersi
onRequest"/>
      <wsdl:output name="getServiceVersionResponse" message="impl:getServiceVer
sionResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getServiceDescription">
      <wsdl:input name="getServiceDescriptionRequest" message="impl:getServiceD
escriptionRequest"/>
      <wsdl:output name="getServiceDescriptionResponse" message="impl:getServic
eDescriptionResponse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
    <wsdl:operation name="getCTSVersion">
      <wsdl:input name="getCTSVersionRequest" message="impl:getCTSVersionReque
st"/>
      <wsdl:output name="getCTSVersionResponse" message="impl:getCTSVersionResp
onse"/>
      <wsdl:fault name="UnexpectedError" message="impl:UnexpectedError"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CodeMappingServiceSoapBinding" type="impl:CodeMappingOper
ations">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/
soap/http"/>
    <wsdl:operation name="getSupportedMaps">
      <wsdlsoap:operation soapAction=""/>
```

```
      <wsdl:input name="getSupportedMapsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSupportedMapsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="mapConceptCode">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="mapConceptCodeRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="mapConceptCodeResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="AmbiguousMapRequest">
        <wsdlsoap:fault name="AmbiguousMapRequest" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="MapNameSourceMismatch">
        <wsdlsoap:fault name="MapNameSourceMismatch" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="MapNameTargetMismatch">
        <wsdlsoap:fault name="MapNameTargetMismatch" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnableToMap">
        <wsdlsoap:fault name="UnableToMap" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="MappingNotAvailable">
        <wsdlsoap:fault name="MappingNotAvailable" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownMapName">
        <wsdlsoap:fault name="UnknownMapName" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownConceptCode">
        <wsdlsoap:fault name="UnknownConceptCode" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="UnknownCodeSystem">
        <wsdlsoap:fault name="UnknownCodeSystem" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceName">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceNameRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceNameResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
```

```
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getServiceDescription">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getServiceDescriptionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getServiceDescriptionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getCTSVersion">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getCTSVersionRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getCTSVersionResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="UnexpectedError">
        <wsdlsoap:fault name="UnexpectedError" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CodeMappingOperationsService">
    <wsdl:port name="CodeMappingService" binding="impl:CodeMappingServiceSoapBi
nding">
      <wsdlsoap:address location="http://localhost:8080/axis/services/
CodeMappingService"/>
    </wsdl:port>
  </wsdl:service>
  <!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->
</wsdl:definitions>
```

**Приложение А**
**(справочное)**

## Эталонная информационная модель HL7

Версия: январь 2009 г.
Ответственная рабочая группа комитета HL7: Modeling and Methodology Work Group.

### А.1 Введение

#### А.1.1 Примечание для голосования в ИСО

Эта версия модели RIM предназначена для обеспечения интерпретации проекта международного стандарта prEN ISO 27953 — Individual Case Safety Report (ICSR) (Индивидуальный отчет по безопасности лекарственного средства). Эта модель HL7 RIM версии 2.25 служит основой указанного стандарта. Она подготовлена на основе ИСО/HL7 21731:2006.

#### А.1.2 Основные сведения

Эталонная информационная модель RIM (Reference Information Model) комитета Health Level Seven (HL7) представляет собой статическую модель информации о здоровье и медицинской помощи в соответствии с областями применения деятельности по развитию стандартов HL7. Она является информационной точкой зрения, согласованной комитетом HL7 и его национальными филиалами. RIM служит основным источником, из которого производится все содержание стандартных спецификаций протокола HL7 Версии 3, связанное с информацией.

А.1.2.1 История разработки модели RIM

Развитие модели RIM началось в апреле 1996 г. Первый выпуск модели RIM был принят Техническим руководящим комитетом HL7 (Technical Steering Committee) на заседании рабочей группы в январе 1997 г. Этот выпуск был известен как проект RIM версии 0.80.

На следующих двух заседаниях рабочей группы основное внимание уделялось ознакомлению с проектом модели RIM и реализации процесса получения и согласования предложений по развитию модели. Процесс сопровождения модели RIM стал известен как «гармонизация модели RIM». Между 1998 г. и 2000 г. было девять заседаний, посвященных гармонизации, на которых рассматривались предложения по улучшения проекта RIM. Кульминацией этой работы стала модель HL7 RIM версии 1.0, представленная на заседании рабочей группы в январе 2001 г. На заседаниях по гармонизации основное внимание уделялось следующим пяти основным темам:

- гарантия охвата стандарта HL7 Версии 2.x. Этот набор предложенных изменений привносил в проект модели все информационное содержание стандарта HL7 Версии 2.x;
- устранение из модели содержания, не имеющего реализации. Этот набор предложенных изменений концентрировался на удалении той информации из проекта модели, которую проблемный технический комитет (steward technical committee) не инициировал и для сохранения которой в стандарте не мог найти достаточных оснований;
- объединенная модель обслуживания. Этот набор предложенных изменений представлял краткий, точно определенный набор структур и словарей, отвечавший информационным потребностям широкого круга клинических сценариев. Эта коллекция предложений, известная как USAM (Unified service action model), явилась плодом объединенных усилий ряда технических комитетов;
- обеспечение качества. Этот набор предложенных изменений был предназначен для устранения несогласованностей проекта модели и конфликтов между моделью и руководством по стилю моделирования. Он положил начало практике записи и отслеживания открытых проблем модели;
- внимание к «левой части» модели. Этот набор предложенных изменений был предназначен для определения развитых структур и словарей, рассчитанных на неклинические части модели (управление движением пациентов, финансы, ведение расписаний). Как и объединенная модель обслуживания, эти предложения явились плодом усилий ряда технических комитетов.

Процесс гармонизации и темы особого внимания позволили получить модель RIM, которая является надежной, ясной, всеобъемлющей, лаконичной и непротиворечивой. Ее структура является и гибкой, и расширяемой, а ее содержание может быть достаточно легко отображено на стандарт передачи медицинских данных HL7 2.x и другие широко используемые спецификации структур данных в здравоохранении.

Содержание настоящего стандарта представляет собой выпуск 1.22 модели HL7 RIM. Тонкая подгонка этой модели продолжается по мере ее использования техническими комитетами для все более расширяющейся области применения семейства разрабатываемых стандартов HL7, основанных на применении моделей. Это семейство получило название HL7 Версия 3.0. В настоящее время модель RIM является весьма стабильной. В то же время процесс гармонизации обеспечивает ее постоянную релевантность и соответствие информационным потребностям разрабатываемым стандартам HL7.

А.1.2.2 Использование RIM в комитете HL7

Модель RIM является критичным компонентом процесса развития стандартов HL7 Версии 3. Она положена в основу всех информационных моделей и структур, разрабатываемых в процессе развития Версии 3.

Разработка стандартов HL7 Версии 3 ведется в соответствии с методологией, ориентированной на применение моделей, согласно которой для описания статических и динамических аспектов требований и разработки стандартов HL7, а также для описания соответствующей семантики и правил деятельности разрабатывается комплекс взаимосвязанных моделей.

Модель RIM представляет собой статическую точку зрения на информационные потребности стандартов HL7 Версии 3. Она включает в себя классы информационных объектов и диаграммы перехода состояний, дополняемые моделями вариантов использования, моделями взаимодействия, моделями типов данных, моделями терминологии и другими типами моделей, необходимых для обеспечения полной точки зрения на требования и архитектуру стандартов HL7. Классы, атрибуты, переходы состояний и ассоциации, описанные в RIM, используются для производства предметно-ориентированных информационных моделей, которые с помощью процессов уточнения ограничений преобразуются в статическую модель содержания информации, используемую в стандарте HL7.

Процесс разработки стандарта HL7 Версии 3 определяет правила производства предметно-ориентированных информационных моделей из модели RIM и уточнения этих моделей в спецификациях стандартов HL7. Правила требуют, чтобы все информационные структуры, описанные в производных моделях, могли быть прослежены обратно к модели RIM и чтобы их семантика и соответствующие правила деятельности не противоречили тем, которые определены в модели RIM. Поэтому модель RIM является основным источником всего информационного содержания стандартов HL7 Версии 3.

Модель RIM используется международными филиалами HL7 для адаптации стандартов HL7 Версии 3 к местным особенностям. С помощью процесса, называемого локализацией, спецификации стандарта Версии 3 могут быть расширены, используя модель RIM как источник нового информационного содержания. Такая новая информация производится из модели RIM и уточняется тем же способом, что и применяется для создания исходной спецификации.

А.1.2.3 Использование RIM вне HL7

В основном модель RIM предназначена для использования комитетом HL7 и его международными филиалами. Однако эта модель может использоваться и вне структур комитета HL7. Первые разработчики, принявшие на вооружение методологию разработки стандартов Версии 3, использовали RIM для конструирования спецификаций сообщений, похожих на описанные в стандарте HL7, в своих собственных средах. К этим разработчикам относятся производители информационных систем, большие сети интегрированных медицинских организаций и правительственные агентства как в США, так и в других странах. Эти же самые первые разработчики крайне активны в комитете HL7 и вносят практические предложения по усовершенствованию RIM и других аспектов процесса разработки Версии 3.

Некоторые организации — участники комитета HL7 сообщили об использовании модели RIM как источника разработки корпоративной информационной архитектуры или как исходной точки для системного анализа и проектирования. Модель RIM действительно может использоваться для таких целей, однако комитет HL7 не дает никаких гарантий, что она полезна не только как эталонная модель, предназначенная для разработки стандартов HL7.

RIM — только одна модель информационных потребностей здравоохранения. Абстрактный стиль модели RIM и возможность ее расширения с помощью спецификаций словарей данных позволяют применять модель RIM для любого мыслимого сценария обмена данными между информационными системами здравоохранения. На самом деле она концептуально применима для любой предметной области, в которой определенные сущности выполняют роли и участвуют в действиях.

Универсальная применимость модели RIM делает ее особенно полезной для организаций, подобных комитету HL7, которые должны учитывать потребности большого и разнообразного членства. Стиль модели RIM делает ее чрезвычайно стабильной, что является еще одним свойством, важным для комитета HL7. Процесс разработки стандартов HL7 состоит в создании предметно-ориентированных моделей, являющихся производными от модели RIM, и последовательного уточнения этих моделей, позволяющего получить модели, специфичные для данной предметной области. Эти специфичные модели предметной области конкретизируют модель RIM и задают ограничения значений атрибутов и ассоциаций между классами, применимые для конкретных случаев. Внешним организациям, рассматривающим возможность использование модели HL7 RIM, рекомендуется принять подобный процесс построения производных моделей в форме преобразования модели RIM.

**А.1.3 Утверждение модели RIM**

Утверждение модели RIM в соответствии с принятыми процедурами голосования позволило вывести эту модель за рамки внутренних артефактов комитета HL7. Как утвержденный стандарт она будет представлена для принятия в качестве стандарта Американского института стандартов (American National Standards Institute — ANSI), а затем, возможно, для принятия в качестве стандарта ИСО. Другие организации — разработчики стандартов, аккредитованные в ANSI или ИСО, в том числе организация ASC X12 и ТК 251 «Медицинская информатика» Европейского комитета по стандартизации (CEN TC 251), выразили интерес в использовании модели RIM или в ссылках на эту модель при разработке своих собственных стандартов. Появление модели RIM в качестве стандарта значительно облегчает ее использование другими организациями — разработчиками стандартов (standards development organizations — SDO). Наличие общей эталонной информационной модели, используемой разными разработчиками стандартов информатизации здоровья, сулит значительные выгоды.

Утверждение модели RIM имеет большое значение для местной стандартизации, осуществляемой национальными филиалами комитета HL7. Чтобы филиалы могли разрабатывать местные расширения стандартов HL7 в соответствии с процессом разработки стандарта HL7 Версии 3, сама модель RIM должна быть стандартом. Это

нередко требуется и местными юридическими нормами, например, местные стандарты должны быть расширениями уже принятых общих стандартов.

При проведении мероприятий, в которых применимы стандарты, органы государственного управления на федеральном уровне, уровне штата и муниципальном уровне в первую очередь рассматривают аккредитованные национальные стандарты. Поскольку модель RIM стандартизует информационное содержание стандартов интероперабельности, разрабатываемых комитетом HL7, органы государственного управления США будут рассматривать ее в контексте решений, обеспечивающих интероперабельность информационных систем здравоохранения.

А.1.3.1 Нормативные части модели RIM

Модель RIM состоит из классов, включенных в один или несколько пакетов предметных областей. Атрибуты, отношения и машины состояний связаны с классами. Нормативными являются только те классы, их атрибуты, отношения и машины состояний, которые включены в предметную область NormativeContent.

Модель RIM представлена на языке UML, расширенном с помощью тегов, специфичных для стандартов HL7 и включенных в метаданные элементов UML-модели. Все стандартные значения метаданных элементов модели UML нормативны, но нормативны также и следующие расширения HL7:

- Class.stateAttribute;
- Class.classCode;
- Attribute.mandatoryInclusion;
- Attribute.cardinality;
- Attribute.vocabDomain;
- Attribute.vocabStrength.

Остальные расширения, предложенные комитетом HL7, предназначены только для целей управления и не являются нормативной частью спецификации модели RIM.

А.1.3.2 Нормативные ссылки

Спецификации типов данных HL7 Версии 3 («Data Types Abstract Specification» и «V3 Data Types Implementable Technology Specification for XML») представляют собой сопутствующие нормативные документы, утверждаемые независимо от модели RIM. Спецификация словарных доменов «HL7 Vocabulary Domain» используется как справочная. Она содержит множество таблиц и терминологических ссылок, цитируемых в качестве доменов значений различных атрибутов модели RIM. Словарные спецификации так называемых «структурных атрибутов» являются нормативной частью модели RIM, поскольку они имеют принципиальное значение для правильного представления медицинской информации с помощью модели RIM. Список этих нормативных таблиц и гиперссылки на них приведены в конце настоящего введения.

А.1.3.3 Смысл нормативности

Нормативными частями модели RIM являются только те части, которые подлежат утверждению и для которых определены правила соответствия. Существенные изменения в нормативной модели RIM требуют повторного утверждения. Следующие изменения модели RIM не считаются существенными:

- изменения описаний элементов модели, не оказывающие материального воздействия на семантику элемента модели;
- перемещение классов из одного пакета в другой, не затрагивающие предметную область NormativeContent;
- изменение иерархии пакетов или наименований пакетов;
- изменение расположения элементов на диаграммах классов;
- изменение элементов модели, не входящих в состав пакета предметной области NormativeContent.

**А.1.4 Понимание модели RIM**

В RIM использован очень абстрактный стиль моделирования. Ее ядром служат базовые классы и их структурные атрибуты. Понимание этих классов и атрибутов существенно для понимания RIM.

А.1.4.1 RIM как абстрактная модель

Модель RIM содержит шесть «базовых» классов:

- класс Act (действие), представляющий действия, которые выполняются и должны быть документированы при оказании медицинской помощи;
- класс Participation (участие), представляющий контекст действия, а именно: кто выполнил действие, для кого оно было выполнено, где было выполнено и т. д.;
- класс Entity (сущность), представляющий физические предметы и существа, которые используются при оказании медицинской помощи и принимают участие в ее оказании;
- класс Role (роль), представляющий роли, выполняемые сущностями, участвующими в действиях по оказанию медицинской помощи;
- класс ActRelashionship (связь действий), представляющий связь одного действия с другим, например, связь между направлением на исследование и состоявшимся исследованием;
- класс RoleLink (связь ролей), представляющий отношения между отдельными ролями.

Три из этих классов — Act, Entity и Role (действия, сущности и роли) — детализируются в виде множества классов-специализаций, или подтипов. В представлении стандарта HL7 подтипы добавляются к модели RIM только в том случае, если требуется определить один или более атрибутов либо ассоциаций, которые не могут быть унаследованы от родительского класса. Классы, описывающие отдельные понятия, не нуждающиеся ни в каких дальнейших атрибутах или ассоциациях, представлены исключительно как уникальный код в контролируемом сло-

варе. Поэтому эти три базовых класса имеют следующие кодируемые атрибуты, используемые для дальнейшего определения моделируемого понятия:

- classCode (в классах Act, Entity и Role), уточняющий назначение класса или соответствующего ему понятия независимо от того, представлен ли он как класс в иерархии RIM;
- moodCode (в классе Act) и determinerCode (в классе Entity), с помощью которых можно указать, представляет ли класс экземпляр или разновидность класса Act или Entity. Если класс является специализацией класса Act, то атрибут moodCode позволяет указать, описывает ли экземпляр класса Act свершившееся или планируемое действие;
- code (в классах Act, Entity и Role), обеспечивающий более детальную классификацию значения атрибута classCode, например, конкретный вид исследования в классе Observation (исследование).

Другие три базовых класса модели RIM — Participation, ActRelationship и RoleLink — не представлены иерархиями обобщения-специализации. Тем не менее эти классы представляют разнообразные понятия, например, разные формы участия или разные типы отношений между действиями. Эти отличия представлены атрибутом typeCode, который должен быть определен в каждом из этих классов.

А.1.4.2 Представление структуры класса в модели RIM

Как упоминалось ранее, модель RIM сконструирована с использованием подмножества семантики языка UML. Она представляет собой набор UML-классов, содержащих один или более атрибутов, которым присвоены типы данных, основанные на независимой спецификации типов данных HL7 Версии 3. Классы связаны рядом отношений ассоциации, идентифицируемых уникальными именами ролей, или отношениями обобщения-специализации.

Каждый из этих элементов имеет текстовое определение.

Внешнее представление атрибутов и ассоциаций управляется кратностью и другими связанными ограничениями, применяемыми к атрибутам и ролям, привязывающим ассоциации к классам.

А.1.4.3 Представление контролируемого словаря

Ряд атрибутов в модели RIM имеют тип данных CS. Это означает, что множество значений такого атрибута должно быть выбрано из множества кодов, определенных в стандарте HL7. Упомянутые ранее атрибуты classCode и typeCode являются примерами атрибутов с типом данных CS.

Все множества кодированных значений этих атрибутов являются частью настоящего стандарта и принимаются в соответствии с теми же принципами голосования, что и классы модели RIM. Каждое множество кодов представлено как словарный домен, то есть множество всех понятий, которые могут использоваться как допустимые значения кодированного поля или атрибута. Важно отметить, что словарный домен состоит из множества понятий, а не слов или кодов.

А.1.4.4 Связанные спецификации

Как отмечено ранее, каждому атрибуту в модели RIM присвоен тип данных. Формальной спецификацией этих типов данных служат нормативная спецификация «HL7 V3 Data Types Implementable Technology Specification for XML» (Реализуемая технологическая спецификация типов данных HL7 Версии 3 для XML) и справочный документ «HL7 Data Types Abstract Specification» (Абстрактная спецификация типов данных HL7). Оба этих документа в настоящее время находятся в процессе утверждения комитетом HL7. Справочная таблица свойств релевантных типов данных включена в приложение В.

**А.1.5 Спецификация модели RIM и диаграммы**

А.1.5.1 Предметные области и классы модели RIM

Содержание следующих разделов настоящего стандарта составляет спецификацию предметных областей и классов модели RIM. Отдельные предметные области и классы «помечены» пиктограммами, обозначающими, какие элементы являются нормативными, а какие — справочными. (Интерпретацию этих пиктограмм можно найти в сопутствующих документах.)

А.1.5.2 Нормативный словарь структурных атрибутов модели RIM

Словарные домены HL7 детализированы в приложении С. Избранное подмножество этих таблиц является частью нормативной спецификации модели RIM. Оно образовано из таблиц значений структурных атрибутов (тех, что имеют тип данных «CS») нормативных классов. В спецификации модели RIM приводятся гиперссылки на эти таблицы в тех местах, где они упоминаются. Кроме того, ниже приведен полный перечень ссылок на эти таблицы.

| | | |
|---|---|---|
| ActClass | ActStatus | ParticipationType |
| ActMood | ContextControl | RelationshipConjunction |
| ActRelationshipCheckpoint | EntityClass | RoleClass |
| ActRelationshipJoin | EntityDeterminer | RoleLinkType |
| ActRelationshipSplit | EntityStatus | RoleStatus |
| ActRelationshipType | ManagedParticipationStatus | |

А.1.5.3 Общие диаграммы модели RIM

Модель RIM представлена диаграммами классов для большинства предметных областей и диаграммами состояний перехода экземпляров классов. Кроме того, полная диаграмма классов модели RIM предоставлена в виде «плаката» в файле формата PDF, обеспечивающего удобные возможности ее увеличения. Ниже приведены диаграммы классов для отдельных предметных областей и диаграммы перехода состояний.

А.1.5.4 Графические диаграммы нормативного содержания модели RIM

Диаграммы классов, включенных в нормативное содержание RIM, представлены на рисунках А.1—А.4.

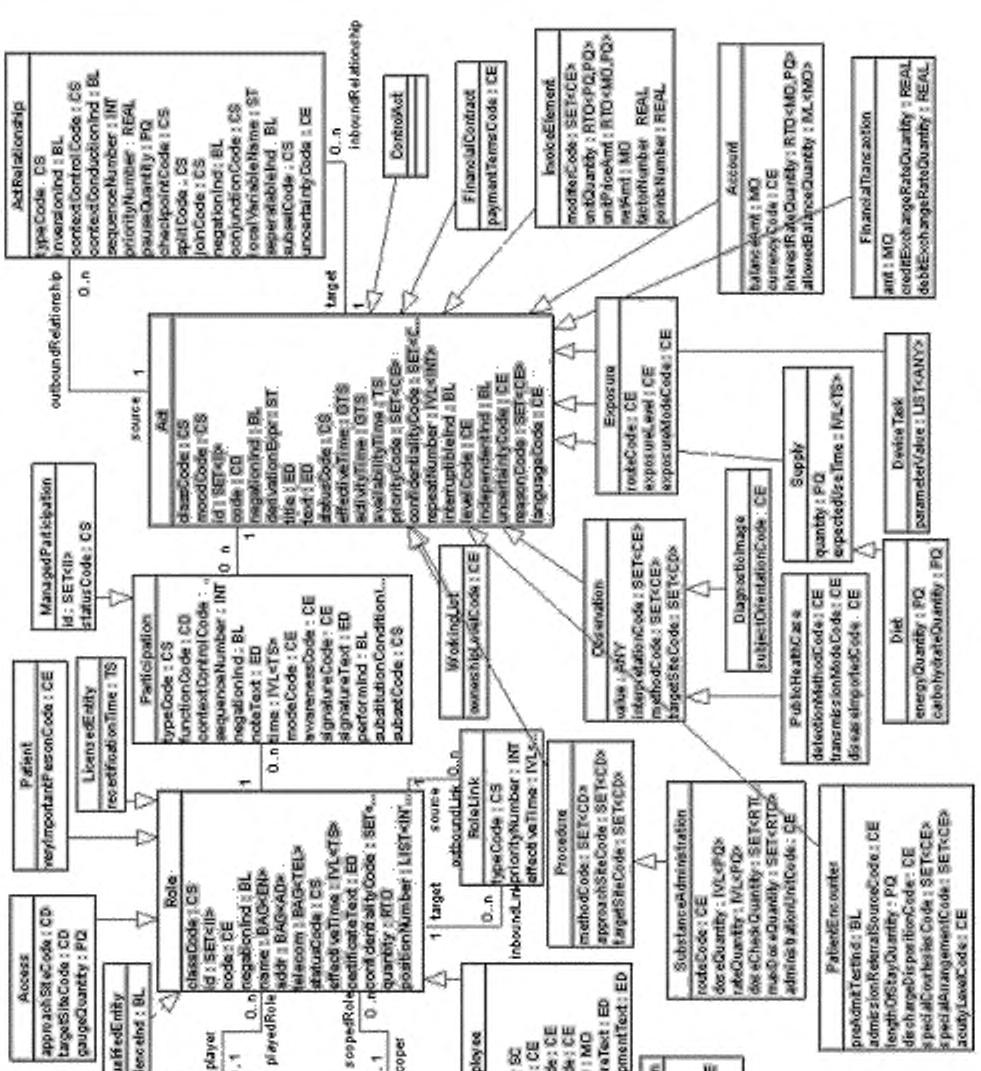


Рисунок А.1 — Диаграмма классов предметной области FoundationClasses

А.1.5.4.2 Предметная область Acts (действия)

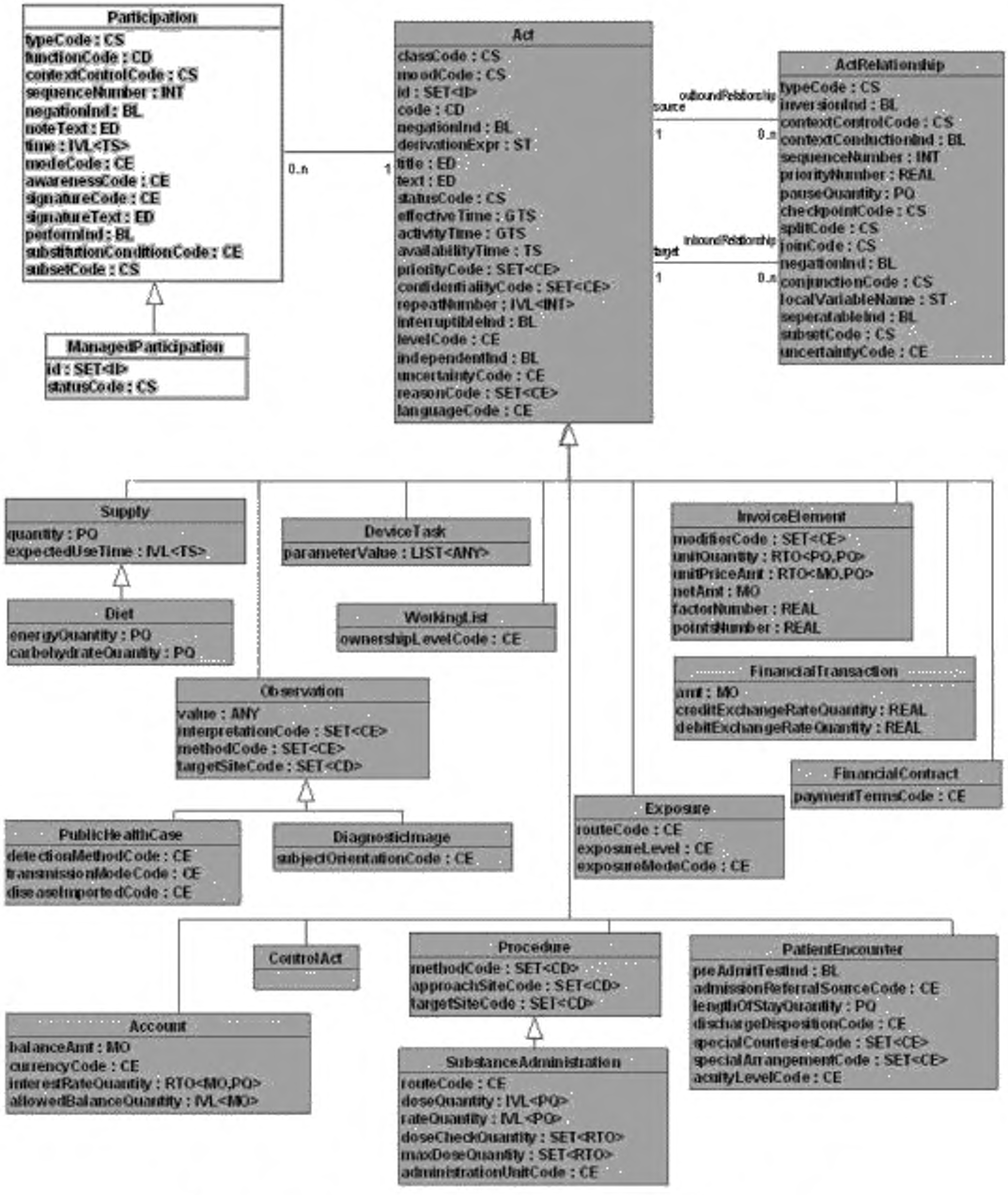


Рисунок А.2 — Диаграмма классов предметной области Acts (действия)

A.1.5.4.3 Предметная область Entities (сущности)

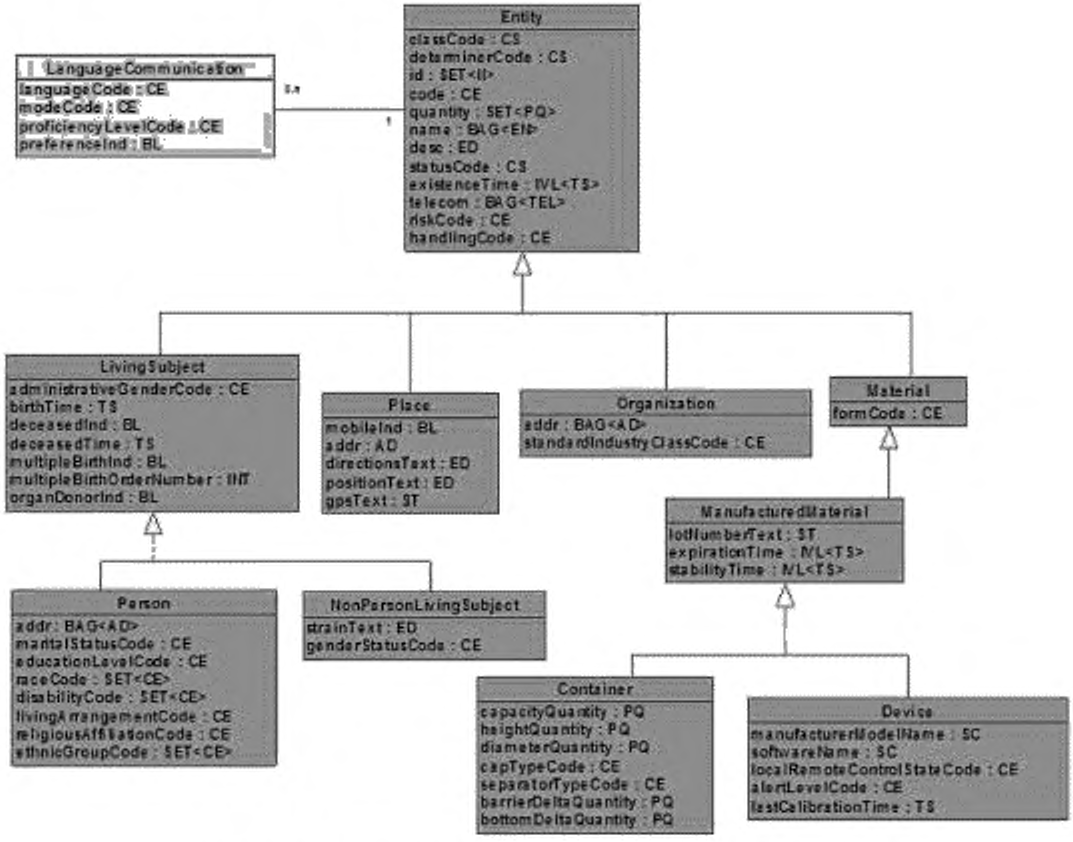


Рисунок А.3 — Диаграмма классов предметной области Entities (сущности)

А.1.5.4.4 Предметная область Roles (роли)

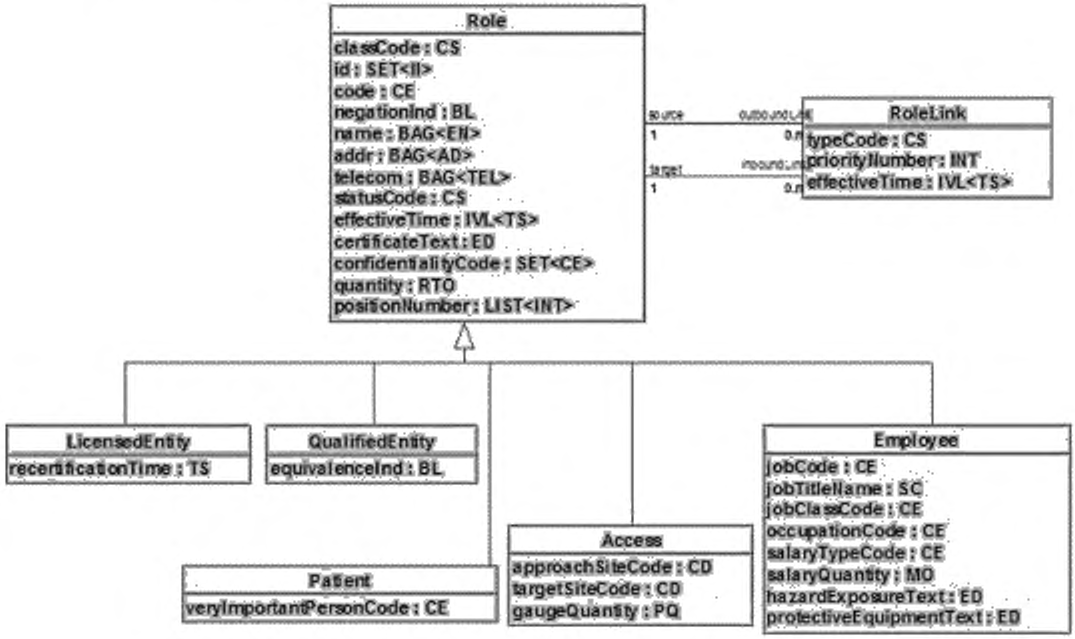


Рисунок А.4 — Диаграмма классов предметной области Roles (роли)

А.1.5.4.5 Диаграмма перехода состояний класса Act (действие)
Диаграмма перехода состояний класса Act (действие) приведена на рисунке А.5.

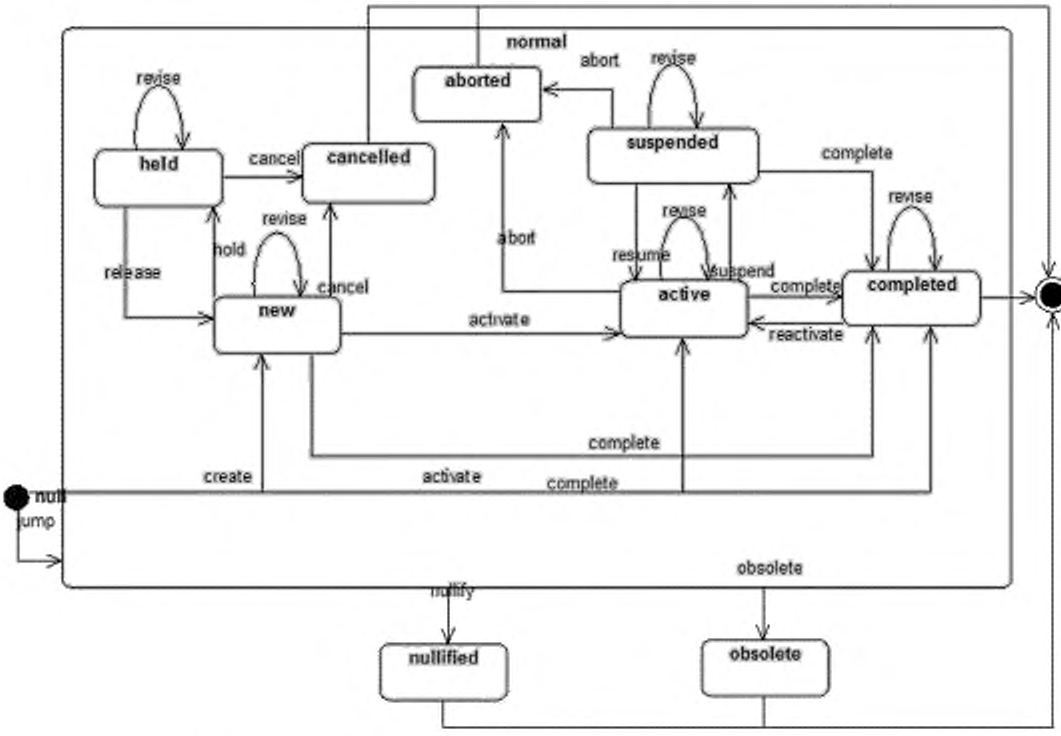


Рисунок А.5 — Диаграмма перехода состояний класса Act (действие)

А.1.5.4.6 Диаграмма перехода состояний класса Entity (сущность)
Диаграмма перехода состояний класса Entity (сущность) приведена на рисунке А.6.

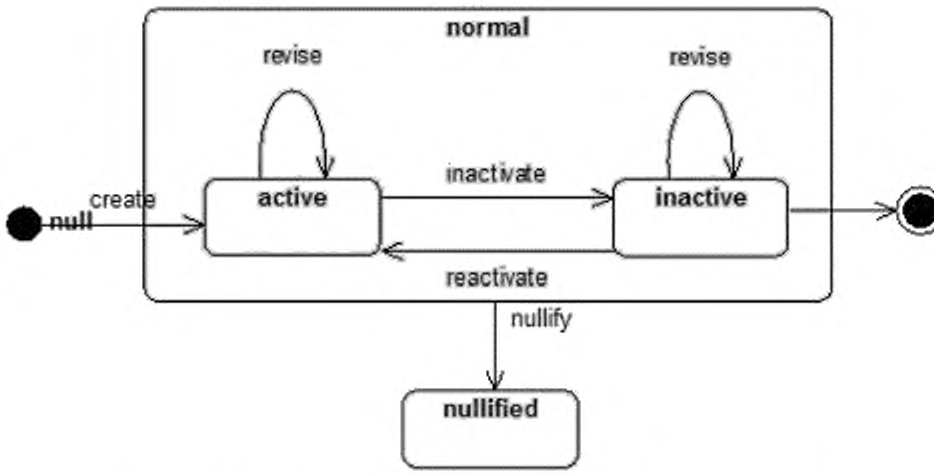


Рисунок А.6 — Диаграмма перехода состояний класса Entity (сущность)

А.1.5.4.7 Диаграмма перехода состояний класса ManagedParticipation (управляемое участие)

Диаграмма перехода состояний класса ManagedParticipation (управляемое участие) приведена на рисунке А.7.

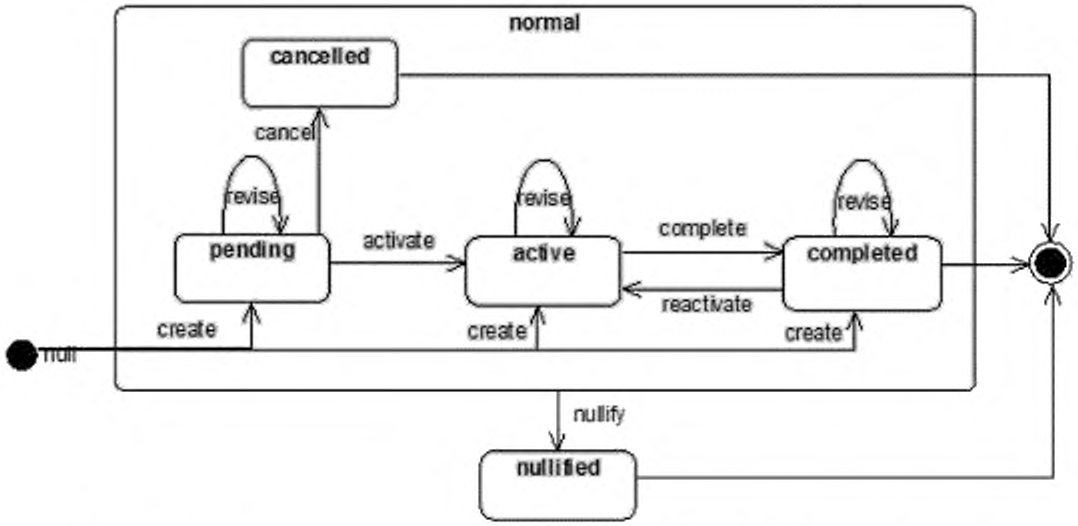


Рисунок А.7 — Диаграмма перехода состояний класса ManagedParticipation (управляемое участие)

А.1.5.4.8 Диаграмма перехода состояний класса Role (роль)

Диаграмма перехода состояний класса Role (роль) приведена на рисунке А.8.

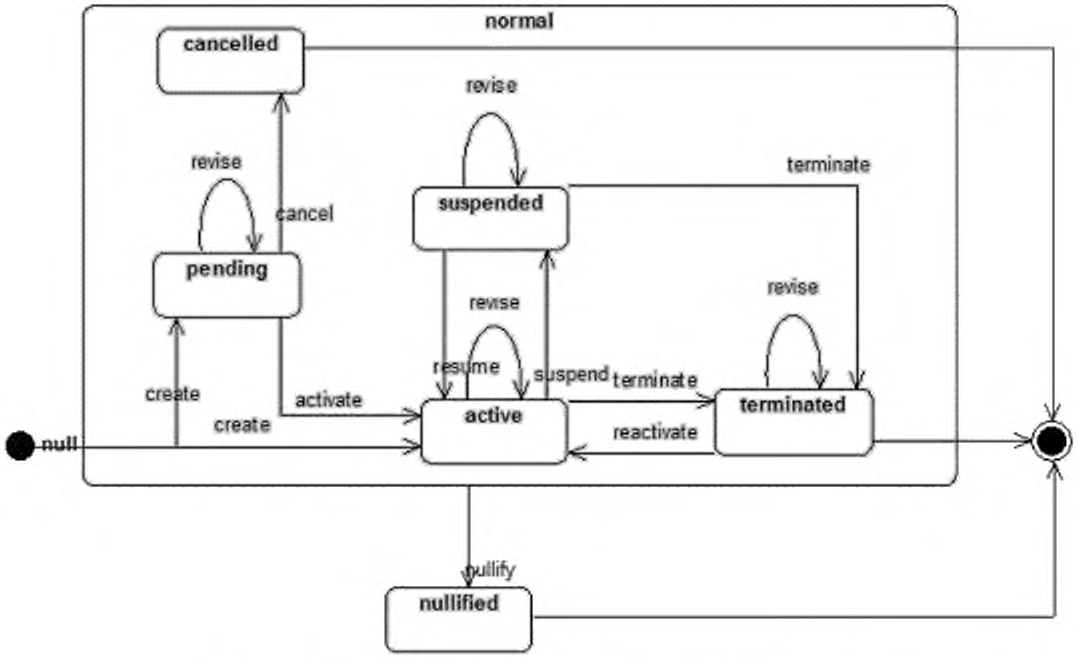


Рисунок А.8 — Диаграмма перехода состояний класса Role (роль)

А.1.5.5 Диаграммы классов инфраструктурных предметных областей модели RIM

Диаграммы классов инфраструктурных предметных областей модели RIM приведены на рисунках А.9—А.13.

А.1.5.5.1 Предметная область StructuredDocuments (структурированные документы)

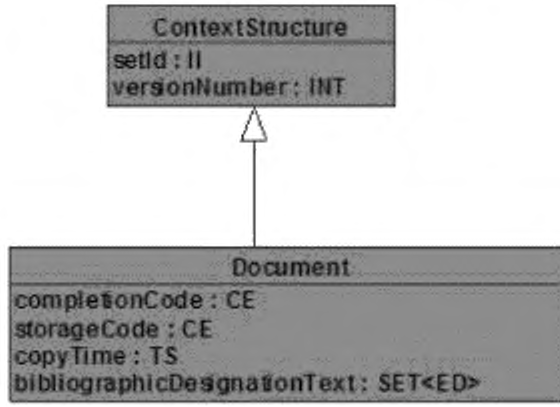


Рисунок А.9 — Диаграмма классов предметной области StructuredDocuments (структурированные документы)

A.1.5.5.2 Предметная область MessageCommunicationsControl (управление передачей сообщений)

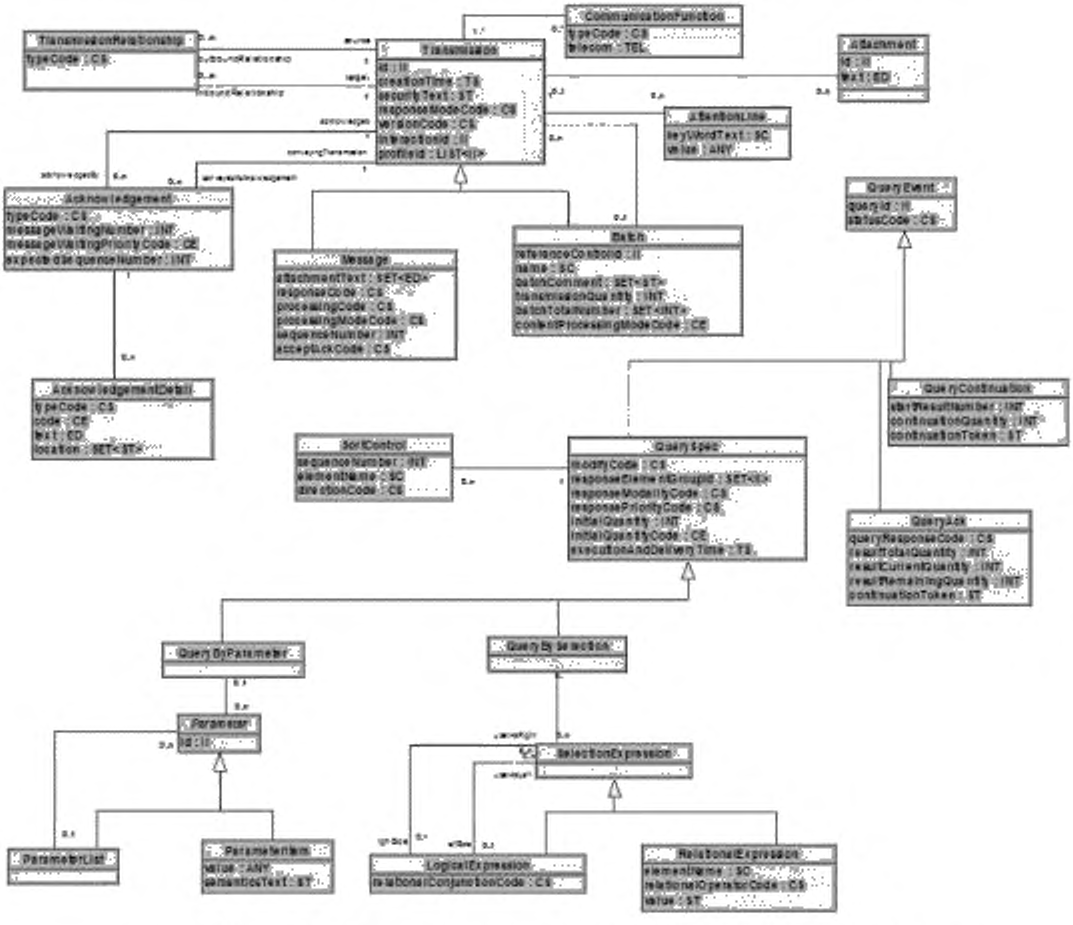


Рисунок А.10 — Диаграмма классов предметной области MessageCommunicationsControl (управление передачей сообщений)

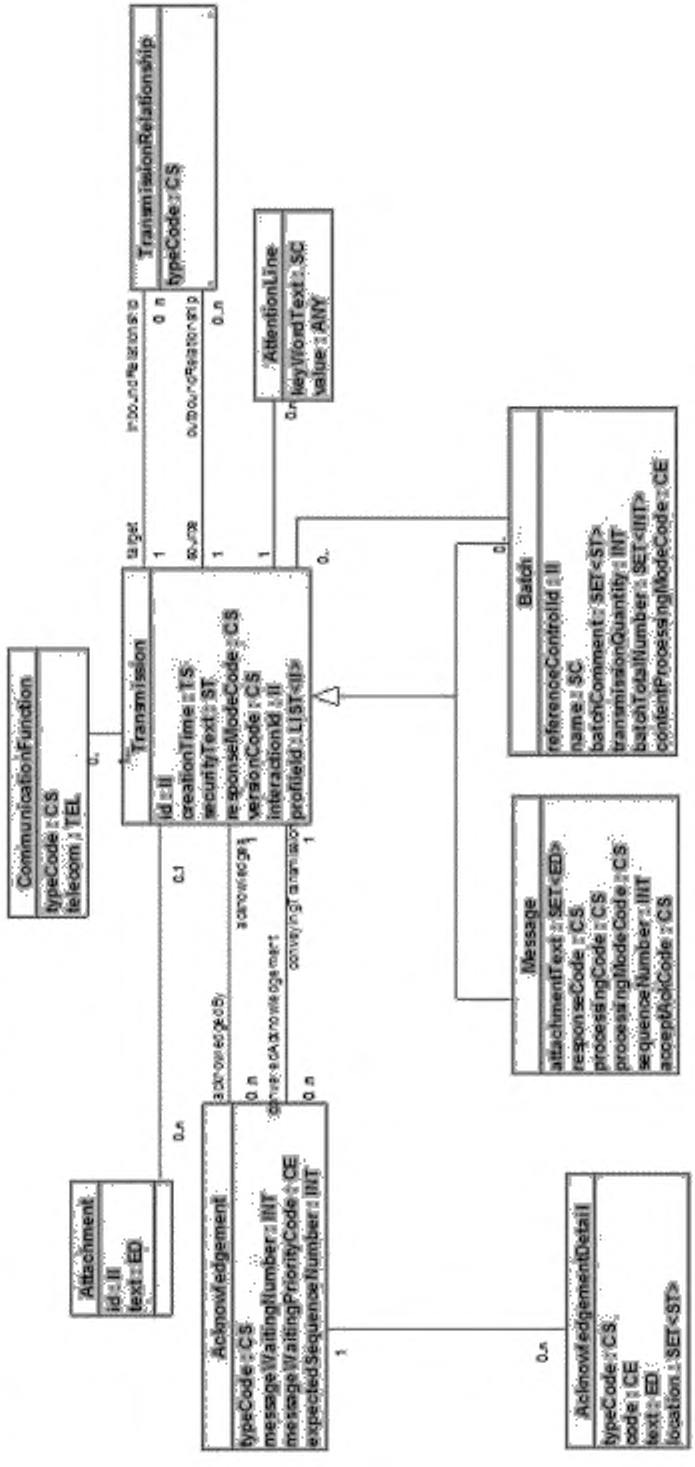


Рисунок А.11 — Диаграмма классов предметной области MessageControl (управление сообщением)

А.1.5.5.4 Предметная область QueryControl (управление запросами)

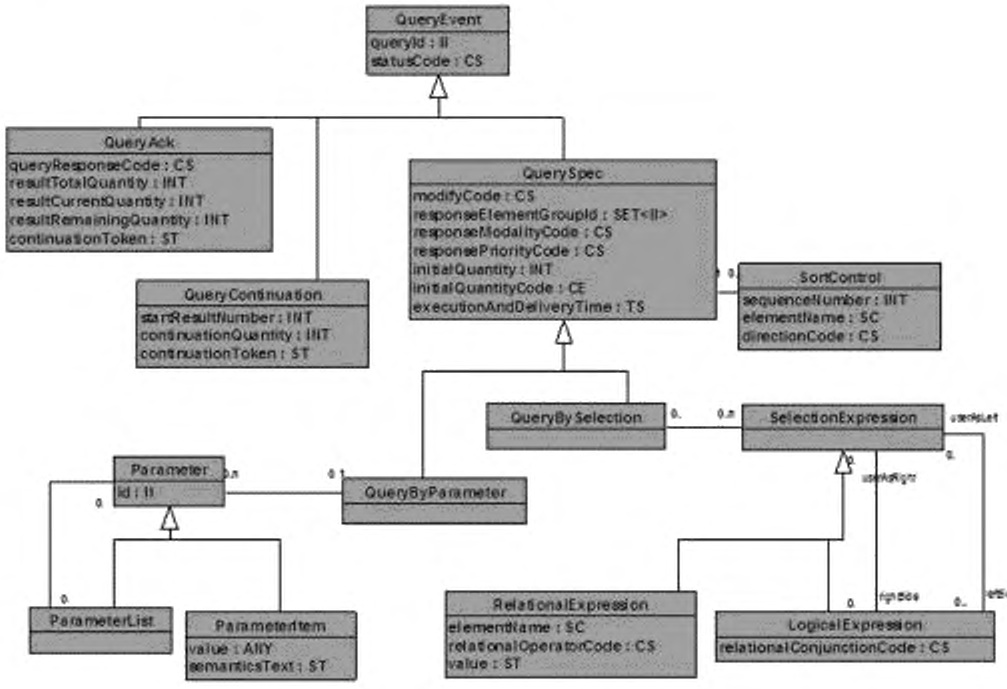


Рисунок А.12 — Диаграмма классов предметной области QueryControl (управление запросами)

А.1.5.5.5 Предметная область CoreInfrastructure (базовая инфраструктура)

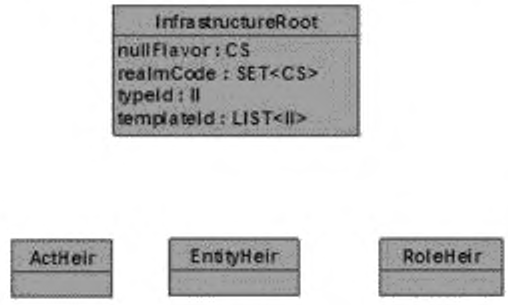


Рисунок А.13 — Диаграмма классов предметной области CoreInfrastructure (базовая инфраструктура)

A.1.5.5.6 Диаграмма перехода состояний класса QueryEvent (событие запроса)

Диаграмма перехода состояний класса QueryEvent (событие запроса) приведена на рисунке А.14.

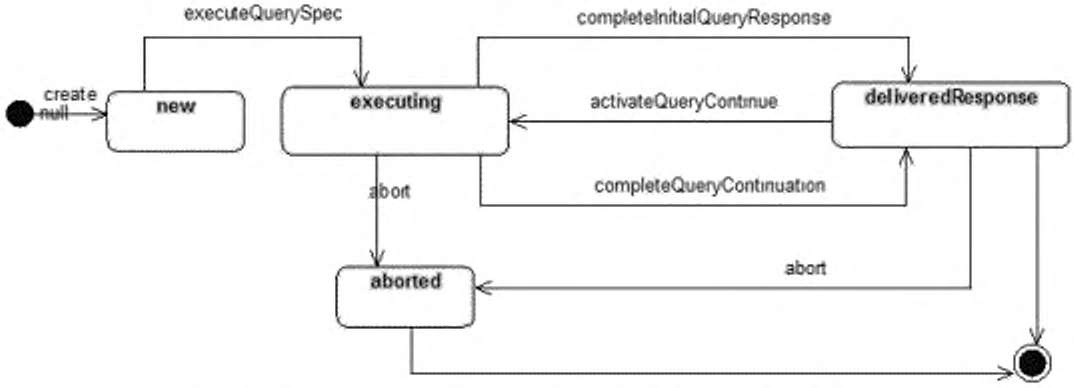


Рисунок А.14 — Диаграмма перехода состояний класса QueryEvent (событие запроса)

**А.1.6 Примечания к версии**

Настоящая версия модели RIM содержит небольшое число технических коррекций (четыре) классов и определений атрибутов. Эти коррекции отмечены «маркированными» аннотациями, включенными в эти определения, а формальные типы аннотаций доступны в файле формата MIF (Model Interchange Format — формат обмена моделями). Скорректированная базовая модель отражает изменения, одобренные в процессе гармонизации по состоянию на ноябрь 2008 года. Эта версия является двадцать пятой по счету, охватывающей содержание, которое должно стать частью выпуска 2 нормативной модели RIM. Она будет предложена для нормативного утверждения в цикле голосования в мае 2009 года в качестве выпуска 2 модели RIM.

Вопросы по содержанию следует адресовать сопредседателям комитета Methodology and Modeling (методология и моделирование) или посылать их в список рассылки этого комитета по адресу mnm@lists.hl7.org.

**А.2 Предметные области**

**А.2.1 Предметная область FoundationClasses (в базовой модели)**

Эта совокупность классов и их ассоциаций представляет «нормативное содержание» модели HL7 RIM. Содержание данной предметной области утверждается в комитете HL7 как нормативный документ.

Диаграмма классов этой предметной области приведена на рисунке А.2.

Предметная область FoundationClasses содержит следующие предметные области:

- Acts (действия);
- Entities (сущности);
- Roles (роли).

А.2.1.1 Предметная область Acts (в предметной области FoundationClasses)

Совокупность классов, включая класс Act и его специализации. Они относятся к действиям и событиям, связанным с оказанием медицинской помощи.

Диаграмма классов этой предметной области приведена на рисунке А.2.

Предметная область Acts содержит следующие классы:

- Account (счет);
- Act (действие);
- ActRelationship (связь действий);
- ControlAct (управляющее действие);
- DeviceTask (функция устройства);
- DiagnosticImage (диагностическое изображение);
- Diet (диета);
- Exposure (воздействие);
- FinancialContract (контракт);
- FinancialTransaction (финансовая операция);
- InvoiceElement (элемент счета-фактуры);
- ManagedParticipation (управляемое участие);
- Observation (исследование);
- Participation (участие),

- PatientEncounter (контакт с пациентом);
- Procedure (процедура);
- PublicHealthCase (событие общественного здоровья);
- SubstanceAdministration (лекарственное назначение);
- Supply (предоставление материала);
- WorkingList (рабочий лист).

А.2.1.2 Предметная область Entities (в предметной области FoundationClasses)

Совокупность классов, содержащая класс Entity, его специализации и связанные с ними квалифицирующие классы. Эти классы представляют участников медицинской помощи и другие сущности здравоохранения.

Диаграмма классов этой предметной области приведена на рисунке А.3.

Предметная область Entities содержит следующие классы:

- Container (контейнер);
- Device (устройство);
- Entity (сущность);
- LanguageCommunication (вербальное общение);
- LivingSubject (живой организм);
- ManufacturedMaterial (произведенный материал);
- Material (материал);
- NonPersonLivingSubject (нечеловеческое живое существо);
- Organization (организация);
- Person (лицо);
- Place (место).

А.2.1.3 Предметная область Roles (в предметной области FoundationClasses)

Совокупность классов, содержащая класс Role и его специализации. Эти классы представляют роли участников процесса оказания медицинской помощи.

Диаграмма классов этой предметной области приведена на рисунке А.4.

Предметная область Roles содержит следующие классы:

- Access (доступ);
- Employee (работник);
- LicensedEntity (лицензированный субъект);
- Patient (пациент);
- Role (роль);
- RoleLink (связь роли).

**А.2.2 Предметная область CommunicationInfrastructure (в базовой модели)**

Эта совокупность предметных областей представляет техническую инфраструктуру стандартов HL7, включая сообщения, структурированные документы и компоненты.

Предметная область CommunicationInfrastructure содержит следующие предметные области:

- CoreInfrastructure (базовая инфраструктура);
- MessageCommunicationsControl (управление передачей сообщений);
- StructuredDocuments (структурированные документы).

А.2.2.1 Предметная область CoreInfrastructure (в предметной области CommunicationInfrastructure)

Эта предметная область содержит классы, являющиеся базовыми элементами коммуникационной инфраструктуры HL7.

Диаграмма классов этой предметной области приведена на рисунке А.13.

Предметная область CoreInfrastructure содержит следующие классы:

- ActHeir (наследование действия);
- EntityHeir (наследование сущности);
- InfrastructureRoot (корень инфраструктуры);
- RoleHeir (наследование роли).

А.2.2.2 Предметная область MessageCommunicationsControl (в предметной области CommunicationInfrastructure)

Эта предметная область содержит классы, относящиеся к техническому определению и управлению коммуникациями, основанными на обмене сообщениями, в стандартах HL7.

Диаграмма классов этой предметной области приведена на рисунке А.10.

Предметная область MessageCommunicationsControl содержит следующие предметные области:

- MessageControl (управление сообщениями);
- QueryControl (управление запросами).

А.2.2.2.1 Предметная область MessageControl (в предметной области MessageCommunicationsControl)

Эта предметная область содержит те элементы модели RIM, которые имеют отношение к управлению, передаче и подтверждению сообщений.

Диаграмма классов этой предметной области приведена на рисунке А.11.

Предметная область MessageControl содержит следующие классы:
- Acknowledgement (подтверждение);
- AcknowledgementDetail (детальная информация подтверждения);
- AttentionLine (идентификация адресата сообщения);
- Batch (пакет сообщений);
- CommunicationFunction (функция коммуникации);
- Message (сообщение);
- Transmission (передача).

A.2.2.2.2 Предметная область QueryControl (в предметной области MessageCommunicationsControl)

Эта предметная область содержит те элементы модели RIM, что имеют отношение к формулировке и передаче запросов, а также к ответам на запросы.

Диаграмма классов этой предметной области приведена на рисунке A.12.

Предметная область QueryControl содержит следующие классы:
- LogicalExpression (логическое выражение);
- Parameter (параметр);
- ParameterItem (компонент параметра);
- ParameterList (список параметров);
- QueryAck (подтверждение запроса);
- QueryByParameter (запрос с параметром);
- QueryBySelection (запрос с фильтрацией);
- QueryContinuation (продолжение запроса);
- QueryEvent (событие запроса);
- QuerySpec (спецификация запроса);
- RelationalExpression (реляционное выражение);
- SelectionExpression (формулировка фильтра);
- SortControl (управление сортировкой).

A.2.2.3 Предметная область StructuredDocuments (в предметной области CommunicationInfrastructure)

Эта предметная область содержит классы, относящиеся к передаче документов в стандартах HL7, представленных архитектурой клинических документов.

Диаграмма классов этой предметной области приведена на рисунке A.9.

Предметная область StructuredDocuments содержит следующие классы:
- ContextStructure (структура контекста);
- Document (документ).

## A.3 Классы

Далее перечислены все классы модели RIM. Порядок сортировки основан на следующих трех критериях:
- основные классы, за которыми следует коммуникационное и инфраструктурное содержание;
- имя основной предметной области (в алфавитном порядке);
- имя класса (в алфавитном порядке).

### A.3.1 Классы предметной области Acts

A.3.1.1 Класс Account (в предметной области Acts)

**Свойства класса Account**

Атрибуты класса Account:
- balanceAmt :: MO;
- currencyCode :: CE;
- interestRateQuantity :: RTO<MO;PQ>;
- allowedBalanceQuantity :: IVL<MO>.

Класс Account является специализацией класса Act.

**Определение класса Account**

Специализация класса Act, представляющая категорию финансовых операций, которые проводятся на отдельном балансе.

**Обсуждение**

Этот класс может использоваться для представления накопленной общей суммы, полученной за товары или услуги, оплаченной за товары или услуги, а также для представления счетов дебита и кредита, между которыми осуществляются операции.

***Примеры — Лицевые счета пациентов, счета за случаи обращения; центры затрат; счета дебиторов.***

**Атрибуты класса Account**

A.3.1.1.1 Account.balanceAmt:: MO (0..1)

Сумма операций по дебету и кредиту счета (остаток) с момента его открытия.

**Обсуждение**

Остаток счета обычно сообщается в валюте, указанной в атрибуте Account.currencyCode. Однако сумму остатка можно сообщать в альтернативных валютах.

A.3.1.1.2 Account.currencyCode:: CE (0..1)

Словарный домен: Currency

Указывает валюту счета.

**Обсуждение**

Конкретные суммы можно сообщать в другой валюте, однако данный атрибут должен представлять валюту, используемую по умолчанию для операций по этому счету.

A.3.1.1.3 Account.interestRateQuantity:: RTO<MO,PQ> (0..1)

Отношение, указывающее процентную ставку остатка по этому счету, и период, для которого она действует.

**Обсуждение**

В этом атрибуте может быть указана процентная ставка для платежей (например, по ссудам, просроченным счетам) или для поступлений (по вложениям капитала и т. д.) в зависимости от типа счета.

***Примеры — 0.10/1a (10 % в год); 0.0005895/1d (0,05895 % в день).***

**Формальное ограничение**

Должна иметься возможность перевода единиц измерения знаменателя, имеющего тип данных PQ, в секунды, то есть знаменателем должно быть время.

A.3.1.1.4 Account.allowedBalanceQuantity:: IVL<MO> (0..1)

Интервал, описывающий минимально и максимально допустимое значение остатка счета.

**Обсуждение**

Этот интервал не обязательно задает «жесткие» пределы (то есть остаток может быть и выше, и ниже указанных пределов). Он указывает такой «целевой» диапазон остатка счета, выход из которого может повлечь за собой определенные последствия. Для остатка счета не обязательно одновременно задавать и верхний, и нижний предел (или какой-либо из них).

***Примеры — Пределы для «прекращения потерь»; пределы для кредита.***

A.3.1.2 Класс Act (в предметной области Acts)

**Свойства класса Act**

Атрибуты класса Act:

- classCode :: CS,
- moodCode :: CS,
- id :: SET<II>,
- code :: CD,
- actionNegationInd :: BL,
- negationInd :: BL,
- derivationExpr :: ST,
- title :: ED,
- text :: ED,
- statusCode :: CS,
- effectiveTime :: GTS,
- activityTime :: GTS,
- availabilityTime :: TS,
- priorityCode :: SET<CE>,
- confidentialityCode :: SET<CE>,
- repeatNumber :: IVL<INT>,
- interruptibleInd :: BL,
- levelCode :: CE,
- independentInd :: BL,
- uncertaintyCode :: CE,
- reasonCode :: SET<CE>,
- languageCode :: CE.

Ассоциации класса Act:

- outboundRelationship::(0..*) ActRelationship::source::(1..1) (ассоциация с классом ActRelationship, роль source — источник);
- inboundRelationship::(0..*) ActRelationship::target::(1..1) (ассоциация с классом ActRelationship, роль target — цель);
- participation::(0..*) Participation::act::(1..1) (ассоциация с классом Participation, роль act — действие).

Класс Act является обобщением следующих классов:

- Account,
- ControlAct,

- DeviceTask,
- FinancialContract,
- FinancialTransaction,
- InvoiceElement,
- Observation,
- PatientEncounter,
- Procedure,
- SubstanceAdministration,
- Supply,
- WorkingList,
- ActHeir,
- ContextStructure.

Диаграмма перехода состояний класса Act приведена на рисунке А.5.

**Определение класса Act**

Запись о чем-то, что делается, что было сделано, что может быть сделано, что планируется или что требуется сделать.

**Примечания к использованию**

Соединение классов действий Act с классами сущностей Entity осуществляется с помощью классов участия Participation и ролей Role. Соединение одного класса действий Act с другим классом Act осуществляется с помощью класса связи действий ActRelationship. Классы участия Participation представляют авторов, исполнителей и другие ответственные стороны, а также субъектов и бенефициариев (в том числе инструменты и материал, используемые при выполнении действия, которые также являются субъектами). Использование атрибута moodCode позволяет различить фактически выполненные действия, запланированные и затребованные действия, а также другие варианты определенности действия.

**Обсуждение и обоснование**

Классы действий являются краеугольным камнем RIM; информация всех предметных областей и процессов в основном сосредоточена в классах действий. Любая профессия или деятельность, включая здравоохранение, состоит из намеренных действий, выполняемых и регистрируемых ответственными лицами. Экземпляр класса действия представляет собой запись о таком намеренном действии.

В этом отношении экземпляр класса Act представляет собой «утверждение» в терминах Rector и Nowlan (1991) [Foundations for an electronic medical record. Methods Inf Med. 30].

В реальном мире отдельная деятельность может развиваться от ее формулировки до выполнения, проходя этапы планирования и указаний. Отражение этих этапов находит свое представление с помощью атрибута moodCode класса Act. Хотя такую деятельность и можно рассматривать в развитии от плана до выполнения, это развитие должно представляться в форме нескольких экземпляров класса Act, каждый из которых имеет ровно одно значение атрибута moodCode, которое не меняется в течение всего срока жизни этого экземпляра. Это связано с тем, что принадлежность и содержание речевых актов в процессе деятельности могут быть различными, и нередко очень важно вести достоверную и постоянно хранящуюся регистрацию этого процесса. Спецификация указаний, обещаний или планов не должна заменяться на спецификацию того, что было выполнено на самом деле, чтобы иметь возможность сравнить результат с более ранними спецификациями. Экземпляры класса Act, являющиеся отражением развития одной деятельности, должны быть связаны между собой с помощью экземпляров класса ActRelationship, у которых атрибут typeCode имеет значение «SEQL» (продолжение).

Экземпляры класса Act, представляющие утверждения или речевые акты, являются единственным способом представления фактов реального мира или процессов в модели HL7 RIM. Правда о реальном мире конструируется только с помощью комбинаций (или выбора) таких утверждений, имеющих определенную принадлежность, и в модели RIM нет ни одного класса, экземпляры которого представляли бы «реальное положение дел» или «реальные процессы» независимо от этих утверждений. В силу этого обстоятельства не проводится никаких различий между деятельностью и ее описанием. Каждый экземпляр класса Act характеризует и то, и другое в различной степени. Примером может служить фактическое утверждение о недавних (но уже завершенных) действиях, сделанное (и подписанное) исполнителем этих действий, обычно известное как протокол или первичная документация (например, протокол хирургической операции, дневниковые записи и т. д.). А изменение состояния текущей деятельности, документируемое исполнителем (или непосредственным наблюдателем), рассматривается как сбор информации об этой деятельности (которая позже заменяется полным протоколом процедуры). Однако и обновление состояния, и протокол процедуры представляются экземплярами класса Act одного рода, которые можно различить по значениям атрибутов наклонения (moodCode) и состояния (statusCode), а также по завершенности информации.

***Пример — Видами действий, типичными для здравоохранения, являются:***

***1 клиническое исследование;***

***2 оценка состояния здоровья (например, жалобы и диагнозы);***

***3 цели медицинской помощи;***

***4 услуги лечения (например, лекарственная терапия, хирургическое лечение, физиотерапия и психотерапия);***

***5 ассистирование, мониторинг, ведение пациента;***

***6 санитарное просвещение пациентов и их близких лиц;***

***7 услуги нотариуса (например, упреждающие указания или отказ от искусственного поддержания жизни);***

***8 редактирование и обработка документов и т. д.***

**Атрибуты класса Act**

A.3.1.2.1 Act.classCode:: CS (1..1) Mandatory

Словарный домен: ActClass

**Определение**

Код, устанавливающий главный тип класса Act, представляющий данный экземпляр этого класса.

**Примечания к использованию**

У тех экземпляров класса Act, которые имеют атрибут code, значения этого атрибута должны быть специализациями значений атрибута classCode. Однако значение атрибута code не может менять смысл значения атрибута classCode.

**Формальное ограничение**

Домен classCode представляет собой строго контролируемый словарь, который не может быть внешним или пользовательским.

Каждый экземпляр класса Act должен иметь атрибут classCode. Если этот класс не специализирован, то атрибут Act.classCode должен иметь наиболее общее значение (ACT).

A.3.1.2.2 Act.moodCode:: CS (1..1) Mandatory

Словарный домен: ActMood

**Определение**

Код, позволяющий различить, что именно представляет данный экземпляр класса Act: фактическое утверждение, команду, возможность, цель и т. д.

**Примечания к использованию**

Чтобы описать развитие конкретной деятельности от ее плана до выполнения, необходимо создать несколько экземпляров классов Act, имеющих разные значения атрибута moodCode, и связать их между собой с помощью экземпляров класса ActRelationship, у которых атрибут typeCode имеет значение «SEQL» (продолжение). (См. описание атрибута ActRelationship.typeCode.)

Поскольку значение атрибута Act.moodCode определяет смысл экземпляра класса Act, то оно должно быть всегда известно. Это означает, что всякий раз, когда создается экземпляр класса Act, его атрибуту moodCode должен быть присвоен допустимый код, который не может меняться в течение всего срока жизни этого экземпляра.

Значение атрибута Act.moodCode контролируемым образом изменяет смысл класса Act подобно тому, как в естественном языке грамматическая форма глагола определенным образом изменяет смысл предложения. Например, если значение атрибута Act.moodCode является признаком фактического события, то весь экземпляр класса Act представляет известный факт. Если оно является признаком плана (намерения), то весь экземпляр класса Act представляет описание того, что должно быть сделано. Значение атрибута Act.moodCode не меняет каким-либо особым способом конкретные свойства класса Act.

Поскольку смысл экземпляра класса Act задается кодом, присвоенным атрибуту moodCode, то значение этого кода влияет на интерпретацию всего этого экземпляра, включая каждое его свойство (атрибут или ассоциацию). Коль скоро значение атрибута moodCode влияет на интерпретацию экземпляра класса, то смысл этого экземпляра, в свою очередь, влияет на смысл его атрибутов. Однако значение атрибута moodCode не может оказать непосредственное влияние на смысл отдельного атрибута.

Классы Act имеют два типа свойств действий — инертные и описательные. Смысл инертных свойств не зависит от значения атрибута Act.moodCode, а интерпретация описательных свойств зависит. Например, у класса Act есть атрибут Act.id, который обеспечивает уникальную идентификацию экземпляра этого класса. Уникальная идентификация объекта никоим образом не зависит от значения атрибута Act.moodCode. Поэтому «интерпретация» идентификатора Act.id является инертной по отношению к атрибуту Act.moodCode.

Напротив, большинство из других атрибутов класса Act являются описательными по отношению к утверждению, передаваемому в форме экземпляра класса Act. Эти атрибуты дают ответы на вопросы, кто выполнил действие, для кого, где, что использовал, как и когда было выполнено это действие. Ответы на вопросы, кто, для кого, где, что использовал, передаются в описательных атрибутах классов Participation, а ответы на вопросы, как и когда — в описательных атрибутах и экземплярах классов ActRelationship. Интерпретация описательного атрибута зависит от интерпретации всего экземпляра и управляется значением атрибута moodCode.

**Обоснование**

Чтобы поддерживать и различать варианты использования классов действий, в них требуется использовать понятие «наклонения», которое в целом трактуется так же, как в грамматике естественного языка. Однако значение

атрибута moodCode ориентировано не на грамматику конкретного естественного языка, а использует более широкий спектр наклонений, близкий к логике с модальностями.

***Пример — Для иллюстрации влияния атрибута moodCode ниже рассмотрены экземпляры класса Act, относящиеся к процессу определения сахара крови.***

Экземпляр специализации класса Act, у которого атрибут moodCode имеет значение «DEF» (definition — описание), содержит справочное описание процесса «определение сахара крови». Связанные с ним экземпляры классов Participation содержат характеристики субъектов, которые должны участвовать в этом процессе, и требуемых для него объектов, например, образец, подразделение, лабораторное оборудование и т. д. Значение атрибута Observation.value указывает абсолютный диапазон значений (домен) результата анализа (например, «15—500 мг/дл»).

Если атрибут moodCode имеет значение «INT» (intent — намерение), это означает, что автор, указанный в экземпляре класса, намерен назначить анализ концентрации сахара в крови («надо определить сахар крови»). Связанные с ним экземпляры классов Participation содержат информацию о тех субъектах и объектах, которые фактически или предположительно участвуют в этом назначении, в первую очередь об авторе намерения или о любом отдельном лице при групповом намерении, а также о передаваемом образце, о требованиях к лабораторному оборудованию и т. д. Атрибут Observation.value в этом случае обычно отсутствует, поскольку речь идет о намерении провести анализ концентрации сахара, а не измерить концентрацию сахара в указанном диапазоне значений. (Иная ситуация описана ниже, когда атрибут moodCode имеет значение «GOL».)

Экземпляр специализации класса Act, у которого атрибут moodCode имеет значение «RQO» (request — требование, что можно рассматривать как разновидность намерения), содержит направление на анализ концентрации сахара в крови («определите сахар крови»). Связанные с ним экземпляры классов Participation содержат информацию о субъектах и объектах, которые фактически или предположительно должны участвовать в процессе выполнения анализа, в первую очередь о заказчике анализа и о выбранном исполнителе, а также о передаваемом образце, о требованиях к лабораторному оборудованию и т. д. Атрибут Observation.value в этом случае обычно отсутствует, поскольку речь идет о намерении провести анализ концентрации сахара, а не измерить концентрацию сахара в указанном диапазоне значений.

Экземпляр специализации класса Act, у которого атрибут moodCode имеет значение «EVN» (event — событие), содержит результат определения сахара крови («сахар крови определен»). Связанные с ним экземпляры классов Participation содержат информацию о субъектах и объектах, фактически участвовавших в процессе определения (включая образец, подразделение, лабораторное оборудование). Атрибут Observation.value содержит измеренное значение (например, «80 мг/дл» или «<15 мг/дл»).

Если атрибут moodCode имеет значение «EVN.CRT» (event-criterion — критерий события), это означает, что автор, указанный в экземпляре класса, рассматривает некоторый класс процессов «определения сахара крови», возможно, с определенным критерием оценки (диапазоном). Связанные с ним экземпляры классов Participation содержат критерий, применяемый, например, к пациенту. Атрибут Observation.value содержит диапазон значений критерия (например, «>180 мг/дл» или «200—300 мг/дл»).

Экземпляр специализации класса Act, у которого атрибут moodCode имеет значение «GOL» (goal — цель, что можно рассматривать как разновидность критерия), содержит информацию о цели, которой требуется достичь («целью является определенный уровень (диапазон) концентрации сахара в крови»). Связанные с ним экземпляры классов Participation содержат информацию, близкую к той, что была указана в намерении определения сахара крови, в первую очередь сведения об авторе цели и о пациенте, по отношению к которому эта цель поставлена. Атрибут Observation.value содержит целевой диапазон значений (например, «80—120 мг/дл»).

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

**Формальное ограничение**

Экземпляр класса Act должен иметь одно и только одно значение атрибута moodCode.

Значение атрибута moodCode конкретного экземпляра класса Act никогда не изменяется.

A.3.1.2.3 Act.id:: SET<II> (0..*)

**Определение**

Уникальный идентификатор экземпляра класса Act.

**Примечания к использованию**

Для успешного обмена данными требуется, чтобы экземпляр класса действия имел единственный идентификатор. Однако при хранении информации в базах данных различных информационных систем экземпляру класса действия могут присваиваться разные идентификаторы.

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.4 Act.code:: CD (0..1)

Словарный домен: ActCode

**Определение**

Код, указывающий конкретный вид действия, представленного экземпляром класса Act.

**Ограничение использования**

*Если атрибут code используется, то его значения должны быть специализациями значения атрибута classCode.*

**Примечания к использованию**

Атрибут Act.code не является обязательным в классе Act. Вместо конкретизации вида действия с помощью атрибута Act.code можно воспользоваться атрибутом Act.classCode и другими атрибутами и свойствами класса Act. Более общий и чаще встречающийся прием состоит в задании вида действия с помощью экземпляра класса Act, в котором атрибут Act.moodCode имеет значение «DEF», связанного с другим экземпляром класса Act с помощью экземпляра класса ActRelationship. Вид действия без труда можно указать и без такой привязки к его определению, используя другие атрибуты, а также классы ActRelationship и Participation. Например, вид лекарственного назначения, передаваемого в экземпляре класса SubstanceAdministarion, можно указать с помощью ассоциации ActRelationship с экземпляром класса Entity, содержащим информацию о конкретном лекарстве.

Для указания вида действия используется код, который берется из какой-либо (обычно внешней) системы кодирования. Система кодирования зависит от конкретной специализации класса Act, например, для класса Observation, описывающего исследования, может использоваться система кодирования LOINC, и т. д.

Атрибуты Act.classCode и Act.code не являются модификаторами друг для друга, однако понятие, передаваемое в атрибуте Act.code, должно логически вытекать из понятия, передаваемого в атрибуте Act.classCode. Негативным примером служит использование атрибута Act.code для передачи понятия «калий» одновременно в экземпляре класса Observation, у которого атрибут Act.classCode имеет значение «SPCOBS» (specimen observation — лабораторное исследование образца), чтобы он означал «лабораторное исследование содержания калия», и в экземпляре класса Medication, у которого атрибут Act.classCode имеет значение «SBADM» (substance administration — лекарственное назначение), чтобы он означал «замещение калия». Такое взаимное изменяющее использование сочетаний атрибутов Act.classCode и Act.code не допускается.

**Примечания к конструированию**

Структура словарного домена ActClass должна найти свое отражение на верхнем уровне структуры словарного домена ActCode и отдельные коды или внешние словари должны быть подчинены структуре словарного домена ActClass.

Необходимо объяснить критерии, по которым целесообразно использовать атрибут code, а не отношение.

***Примеры — Физикальное исследование, определение калия в сыворотке крови, госпитализация, финансовая транзакция по оплате лечения и т. д.***

A.3.1.2.5 Act.actionNegationInd:: BL (0..1)

**Определение**

Признак, указывающий, что утверждение, передаваемое в экземпляре класса Act, является отрицанием действия события, описанного атрибутами этого экземпляра.

**Примечания к использованию**

Атрибут actionNegationInd используется как отрицание квантора существования фактического, планируемого или описанного действия события. Если атрибут Act.moodCode имеет значение «EVN» (event — событие), то атрибут actionNegationInd указывает, что действие, описанное в данном экземпляре класса Act, не произошло. Если атрибут Act.moodCode имеет значение «INT» (intent — намерение), то атрибут actionNegationInd указывает, что совершение действия, описанного в данном экземпляре класса Act, является нежелательным. Если атрибут Act.moodCode имеет значение «EVN.CRT» (criterion — критерий), то атрибут actionNegationInd указывает, что условие основано на несуществовании события, описанного в данном экземпляре класса Act. Значение true атрибута actionNegationInd лишено смысла для действий определения сущностей.

Значение атрибута actionNegationInd воздействует указанным выше образом на описательные атрибуты класса Act (включая Act.code, Act.effectiveTime, Observation.value, Act.doseQty и т. д.) и на любые их компоненты. Инертные свойства, например Act.id, Act.moodCode, Act.confidentialityCode, и особенно ассоциация с классом Participation, имеющая роль автора, остаются неизменными. Эти инертные свойства всегда имеют одно и то же значение: то есть автор остается и автором отрицательного исследования. Кроме того, атрибут actionNegationInd не воздействует и на большинство связей ActRelationship (за исключением компонентов). Конкретные указания см. в описании атрибута isRecordCharacteristic, а также в описаниях систем кодирования ActRelationshipType и ParticipationType.

Например, крайне конфиденциальное указание, записанное д-ром Джонсом в форме «не применять сукцинилхолин» в связи (класс ActRelationship) с имевшейся злокачественной гипертермией (класс Observation), является отрицанием положительного указания «применить сукцинилхолин» (атрибут Act.code), но тем не менее остается указанием, написанным доктором Джонсом для пациента Джона Смита, и причина этого указания — имевшаяся у пациента злокачественная гипертермия.

Однако дополнительные детали, передаваемые в описательных атрибутах, будут частью отрицания, ограничивая его воздействие. Например, если в указании не применять субстанцию присутствует атрибут дозы doseQuantity, то это означает, что нельзя давать эту конкретную дозу субстанции (но любая другая доза могла бы оставаться допустимой).

Экземпляр класса Act, у которого атрибут actionNegationInd имеет значение «true», тем не менее остается утверждением об определенном факте, описанном в этом экземпляре. Например, отрицание утверждения «1 июля пациенту сделана аппендэктомия» означает, что его автор определенно отрицает, что 1 июля была сделана аппендэктомия, и что он несет ту же самую ответственность за это утверждение и те же самые требования к его доказательству, как если бы он не использовал отрицание. И наоборот, признак отрицания, переданный в атрибуте actionNegationInd, никоим образом не отрицает того, что факт был подтвержден или что утверждение имело место. Это равным образом относится ко всем наклонениям утверждения, задаваемым атрибутом moodCode, например, применение отрицания к направлению является указанием не делать того, что в нем написано, а вовсе не лапидарное утверждение, что такого направления нет. Такие лапидарные утверждения обрабатываются как отрицание действия управления, которое создало действие субъекта. Например, направление этого типа (значение DEFN атрибута moodCode) с автором доктором Смитом не было создано.

Следует учесть, что в экземплярах класса Observation этот атрибут указывает, что действие не произошло, то есть исследование не имело места. Если надо указать, что исследование проведено, но результат отрицательный, следует использовать атрибут Observation.valueNegationInd.

***Пример — Использование этого признака в экземпляре класса, у которого атрибут Act.moodCode имеет значение «EVN» (event — событие), позволяет передать утверждение «хирургическая операция не выполнена» или «согласие не было дано». Использование в экземпляре класса, у которого атрибут Act.moodCode имеет значение «ORD» (order — заказ), позволяет передать утверждение «не применять данное лекарственное средство». А в экземпляре класса, у которого атрибут Act.moodCode имеет значение «EVN.CRT» (criterium — критерий), использование признака отрицания позволяет передавать критерии выполнения действия вида «если пациент не был госпитализирован...».***

A.3.1.2.6 Act.negationInd:: BL (0..1)

**Определение**

Признак, указывающий, что к описательным атрибутам утверждения, передаваемого в экземпляре класса Act, должно быть применено отрицание.

**Примечания к использованию**

Атрибут negationInd используется как отрицание квантора существования. Для действия события (задаваемого атрибутом moodCode) он означает, что данное событие не имело места. Для действия намерения этот атрибут указывает, что оно не планировалось или не было желательным. Для действия критерия он указывает, что условие основано на несуществовании события. Значение true атрибута negationInd лишено смысла для действий определения сущностей.

Значение атрибута negationInd воздействует указанным выше образом на описательные атрибуты класса Act (включая Act.code, Act.effectiveTime, Observation.value, Act.doseQty и т. д.) и на любые их компоненты. Инертные свойства, например Act.id, Act.moodCode, Act.confidentialityCode, и особенно ассоциация с классом Participation, имеющая роль автора, остаются неизменными. Эти инертные свойства всегда имеют одно и то же значение: то есть автор остается и автором отрицательного исследования. Кроме того, атрибут negationInd не воздействует и на большинство связей ActRelationship (за исключением компонентов). Конкретные указания см. в описании атрибута isRecordCharacteristic, а также в описаниях систем кодирования ActRelationshipType и ParticipationType.

Например, крайне конфиденциальное указание, записанное доктором Джонсом в форме «не применять сукцинилхолин» в связи (класс ActRelationship) с имевшейся злокачественной гипертермией (класс Observation), является отрицанием положительного указания «применить сукцинилхолин» (атрибут Act.code), но тем не менее остается указанием, написанным доктором Джонсом для пациента Джона Смита, и причина этого указания — имевшаяся у пациента злокачественная гипертермия.

Однако дополнительные детали, передаваемые в описательных атрибутах, будут частью отрицания, ограничивая его воздействие. Например, если в указании не применять субстанцию присутствует атрибут дозы doseQuantity, то это означает, что нельзя давать эту конкретную дозу субстанции (но любая другая доза могла бы оставаться допустимой).

Экземпляр класса Act, у которого атрибут negationInd имеет значение «true», тем не менее остается утверждением об определенном факте, описанном в этом экземпляре. Например, отрицание утверждения «1 июля выявлена одышка» означает, что его автор определенно отрицает, что 1 июля была одышка, и что он несет ту же самую ответственность за это утверждение и те же самые требования к его доказательству, как если бы не использовал отрицание. И наоборот, признак отрицания, переданный в атрибуте negationInd, никоим образом не отрицает того, что факт был подтвержден или что утверждение имело место. Это равным образом относится ко всем наклонениям утверждения, задаваемым атрибутом moodCode, например, применение отрицания к направлению является указанием не делать того, что в нем написано, а вовсе не лапидарное утверждение, что такого направления нет. Такие лапидарные утверждения обрабатываются как отрицание действия управления, которое создало действие субъекта. Например, направление этого типа (значение «DEFN» атрибута moodCode) с автором доктором Смитом не было создано.

Следует учесть, что в экземплярах класса Observation этот атрибут указывает, что действие не произошло, то есть исследование не имело места. Если надо указать, что исследование проведено, но результат отрицательный, следует использовать атрибут Observation.valueNegationInd.

***Пример — Использование этого признака в экземпляре класса, у которого атрибут Act.moodCode имеет значение «EVN» (event — событие), позволяет передать утверждение «хирургическая операция не выполнена» или «согласие не было дано». Использование в экземпляре класса, у которого атрибут Act.moodCode имеет значение «ORD» (order — заказ), позволяет передать утверждение «не применять данное лекарственное средство». А в экземпляре класса, у которого атрибут Act.moodCode имеет значение «EVN.CRT» (criterium — критерий), использование признака отрицания позволяет передавать критерии выполнения действия вида «если пациент не был госпитализирован...».***

A.3.1.2.7 Act.derivationExpr:: ST (0..1)

**Определение**

Строка символов, содержащая выражение на формальном языке, указывающее, каким образом значения атрибутов экземпляра класса Act выводятся из входных параметров, связанных с этим экземпляром отношениями «произведен из».

**Примечания к использованию**

Производный результат исследования, передаваемый в экземпляре класса Observation, может быть определен с помощью ассоциаций ActRelationship, у которых атрибут typeCode имеет значение «DRIV» (is derived from — произведен из) и которые связывают данный экземпляр с другими экземплярами класса Observation. Например, для определения производного исследования «среднее содержание гемоглобина в эритроците» (MCH) надо связать исследование MCH с исследованием концентрации гемоглобина (Hb) и абсолютным содержанием эритроцитов (RBC). В этом случае производное выражение должно быть задано формулой MCH = Hb / RBC.

**Формальное ограничение**

Производное выражение представляют в виде строки символов.

**Открытый вопрос**

Синтаксис такого выражения полностью еще не определен. Необходимо отслеживать состояние его разработки.

A.3.1.2.8 Act.title:: ED (0..1)

**Определение**

Слово или фраза, под которыми данный экземпляр класса Act известен людям.

**Примечания к использованию**

Этот идентификатор не является формальным. Его надо скорее рассматривать как человекочитаемое общее название. Однако он похож на атрибут id тем, что относится к конкретному действию, а не к типу действия. (В экземпляре класса Act, у которого атрибут moodCode имеет значение «DEF» (definition — определение), атрибут title относится к данному конкретному определению, а не к широкой категории, которая может быть задана значением атрибута code.)

***Пример — Название научного исследования (например, «Scandinavian Simvastatin Study»), название судебного дела (например, «Brown v. Board of Education»), наименование другого типа рабочего проекта или действия. Для действий, представляющих документы, в этом атрибуте передается название документа.***

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

**Формальное ограничение**

До версии 2.05 модели RIM этот атрибут имел тип данных ST. Начиная с версии 2.05, его тип данных был расширен до ограниченного типа данных ED. Наложенные ограничения идентичны тем, что присущи типу данных ST, только атрибут типа среды (mediaType) должен иметь значение «text/x-hl7-title+xml». Это было сделано, чтобы обеспечить возможность использования достаточного ассортимента разметки, позволяющего передать семантику научных фраз, например, названия химических компонентов. Такая разметка не должна использоваться для указания простых предпочтений отображения. По умолчанию атрибут mediaType должен иметь значение «text/plain».

A.3.1.2.9 Act.text:: ED (0..1)

Текстовое или мультимедийное описание действия (либо ссылка на такое описание), которое может быть предоставлено читателю, отслеживающему это действие.

**Примечания к использованию**

Содержание этого описания не считается частью функциональной информации, передаваемой между компьютерными системами. Если предполагается, что информация, переданная в экземпляре класса Act, предназначена в том числе для людей (читателей и исполнителей), то автоматизированная система должна показать пользователю поле Act.text либо каким-то образом уведомить его о том, что такое поле существует, и показать его по требованию.

Описания в форме свободного текста используются, чтобы человек мог интерпретировать содержание и контекст действия, но вся информация, необходимая для автоматизированных функций, должна быть передана, используя надлежащие атрибуты и ассоциированные объекты.

Читатель должен быть способен воспринять значение атрибута text само по себе, не привлекая какую-либо кодируемую информацию и не рискуя неправильной интерпретацией или неполным пониманием содержания действия. Например, значения компонентов II.root или CD.codeSystem обычно не выводятся читателю и не должны быть частью значения атрибута text.

Ожидается, что представление читателю будет включать в себя «моментальный снимок» всех «зависимых» экземпляров классов ActRelationship и Participation, а также рекурсивно отслеживаемых дочерних экземпляров классов Act. Однако некоторые элементы данных не обязаны включаться в это представление. К ним относятся:

- разделы компонентов (ActRelationship=COMP, classCode <= DOCSECT);
- атрибут title;
- все, что присоединено к экземпляру отношения ActRelationship типа «XFRM»;
- предыдущие версии (экземпляры отношения ActRelationship типа «RPLC»),

Атрибут text МОЖЕТ включать в себя информацию, отсутствующую в других атрибутах или отношениях, но при этот он ДОЛЖЕН включать в себя информацию, которая в них присутствует, за исключением вышеуказанной.

Атрибут text НЕ ДОЛЖЕН использоваться для передачи машинообрабатываемой информации. Такая информация должна передаваться в дискретных атрибутах. Любая информация, переданная в атрибуте text, но не содержащаяся в каких-либо кодируемых атрибутах, должна оставаться закрытой компьютерным системам. Поэтому атрибут text не должен содержать сведения, которые отрицают или существенно модифицируют информацию, передаваемую в кодируемых атрибутах.

Для передачи «дополнительного текста» следует создать отношение (например, типа «компонент» или «субъект действия») с другим экземпляром класса Act, в котором вся информация содержится в атрибуте text, при этом его атрибут code может обозначать «аннотацию» или аналогичное понятие.

**Примечания к использованию**

Следует уточнить силу утверждения «атрибут text НЕ ДОЛЖЕН использоваться для передачи машинообрабатываемой информации» — должно ли оно быть формальным ограничением?

***Пример — В экземпляре класса Act, у которого атрибут moodCode имеет значение «DEF» (определение), атрибут Act.text может содержать справочную информацию о действии. В экземпляре класса Act, у которого атрибут moodCode имеет значение «RQO» (направление), атрибут Act.text будет содержать специфические инструкции, применяемые только к этому направлению.***

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.10 Act.statusCode:: CS (0..1)

Словарный домен: ActStatus

**Определение**

Код, описывающий состояние действия.

**Примечания к использованию**

Атрибут statusCode отражает состояние действия. Для класса Observation он означает состояние действия исследования (например, «новое», «завершено», «отменено», а не состояние того, что исследуется (например, состояние заболевания, наличие аллергии на пенициллин). Для передачи информации о состоянии обследуемого субъекта следует указывать ее в атрибутах code или value класса Observation или в связанном экземпляре класса Observation.

**Обоснование**

В исходной версии модели RIM этот атрибут был определен как повторяющийся, чтобы можно было отразить наличие вложенных подсостояний, указанных на диаграмме перехода состояний. На практике, однако, необходимость передачи нескольких значений состояния никогда не возникала. Поэтому комитетам рекомендуется ограничить кратность этого атрибута до 1 во всех конструкциях сообщений.

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.11 Act.effectiveTime:: GTS (0..1)

**Определение**

Выражение, указывающее «эффективное время» — клинически значимое или оперативное время действия за вычетом административной деятельности.

**Примечания к использованию**

Понятие «эффективного времени» называется также «основным» временем (в стандарте Arden Syntax) или «биологически значимым временем» (в стандарте HL7 v2.x). Этот атрибут надо отличать от атрибута activityTime, описывающего время деятельности.

Для исследований время деятельности может быть много позже времени исследуемого свойства. Например, при анализе газов, содержащихся в артериальной крови, результат всегда будет получен через несколько минут после взятия биоматериала, а за это время физиологическое состояние пациента может значительно измениться.

Для существенно физической деятельности (хирургические процедуры, транспортировка и т. д.) эффективным временем является время целевого действия, например, если целью транспортировки является доставка

груза из пункта А в пункт Б, то эффективным временем является время в пути. Однако действие обычно также включает в себя дополнительную работу, которая необходима для достижения цели действия, но не является существенной для этой цели.

Например, время, необходимое для приезда на место погрузки А и последующего возвращения на свою автобазу из пункта разгрузки Б, включается в физическую деятельность, но не включается во время транспортировки полезного груза. Другой пример — рабочие часы, считающиеся эффективным временем, могут быть с 8:00 до 17:00 независимо от того, тратит ли человек на дорогу 10 минут или 2 часа. Приход на работу необходим, но на рабочие часы не влияет.

***Примеры***

***1 Для клинических исследований атрибут effectiveTime содержит время проведения исследования (эффективное время) пациента.***

***2 Для контрактов атрибут effectiveTime указывает срок действия контракта.***

***3 Для информированных согласий атрибут effectiveTime указывает срок действия согласия.***

***4 Для лекарственных назначений атрибут effectiveTime указывает время, в течение которого надо принимать лекарство, включая режим приема (например, трижды в день в течение 10 дней).***

***5 Для хирургической процедуры (операции) атрибут effectiveTime указывает время, значимое для пациента, например, время между разрезом и последним швом.***

***6 Для действий транспортировки атрибут effectiveTime указывает время в пути.***

***7 Для посещений пациентов атрибут effectiveTime указывает «административное время», то есть даты начала и завершения посещения, определяемые в соответствии с административным регламентом, в отличие от фактического времени оказания медицинской помощи при данном посещении.***

A.3.1.2.12 Act.activityTime:: GTS (0..1)

**Определение**

Выражение, указывающее время деятельности, то есть когда происходило действие, описанное в экземпляре класса Act, либо в зависимости от значения атрибута Act.moodCode, когда оно должно было происходить по предварительному плану, по текущему плану, когда оно могло происходить и т. д. Время деятельности указывает длительность действия, описанного в экземпляре класса Act. Атрибут activityTime содержит полное время деятельности, в том числе время, затрачиваемое на ее подготовку и на необходимые действия после ее завершения. В этих случаях значения атрибутов activityTime могут использоваться в функциях управления/планирования процедур и лекарственных назначений, поскольку более полно описывают затраты времени по сравнению с длительностью самого действия.

**Примечания к использованию**

Атрибут activityTime в большей степени предназначен для использования в административных, а не клинических целях. Клинически релевантное время передается в атрибуте effectiveTime. Если передается результат исследования ранее выявленного симптома, то атрибут activityTime содержит время исследования симптома, в отличие от атрибута effectiveTime, в котором должно передаваться время появления симптома. Поэтому значение атрибута activityTime может существенно отличаться от значения атрибута effectiveTime того же самого действия. Но даже без учета клинических аспектов разработчики должны рассматривать значение атрибута effectiveTime как наиболее релевантное время действия.

Значение атрибута activityTime указывает время деятельности, описанной в экземпляре класса Act, а не время регистрации деятельности. Во многих приложениях учитывается время регистрации результата исследования, а не время, в течение которого осуществлялось это исследование. В этих случаях должно использоваться время, передаваемое в атрибуте Participation.time (например, время, указанное автором). Регистрируемые результаты могут относиться к исследованиям, выполненным в процессе посещения пациента, и время посещения нередко является достаточно информативным, поэтому значение атрибута activityTime не является клинически релевантным.

Атрибут activityTime является описательным: подобно атрибуту effectiveTime, он всегда описывает событие действия, которое имеет место или могло иметь место. Например, если затребована процедура, то атрибут activityTime описывает общее требуемое время процедуры, которое может отличаться от времени, фактически затраченного на ее выполнение. Напротив, время участия автора требования Participation.time является инертным, поскольку оно указывает, когда это требование было написано, и не имеет никакого отношения к времени выполнения требования.

A.3.1.2.13 Act.availabilityTime:: TS (0..1)

**Определение**

Момент времени, когда информация об экземпляре класса Act (в зависимости от значения атрибута Act.moodCode) впервые становится доступной системе, воспроизводящей информацию о действии, описанном в этом экземпляре. Значение атрибута availabilityTime представляет собой метаданные, описывающие факт регистрации, а не само действие.

**Примечания к использованию**

Значение атрибута availabilityTime — субъективная вторичная часть информации, добавленная (или измененная) системой, информирующей о действии. Она не может быть приписана автору действия (и не должна

включаться в информацию, которую автор засвидетельствует своей подписью). Система, информирующая о действии, нередко отличается от системы, зарегистрировавшей действие, то есть получает информацию о действии из другой системы. Время получения этой информации должно быть присвоено атрибуту availabilityTime, поскольку, начиная с этого момента, пользователи системы-получателя должны иметь возможность узнать об этом экземпляре класса Act.

Система оценивает значение атрибута availabilityTime по получению (или созданию) информации и должна быть способна изменить его в том и только в том случае, если она передает эту информацию дальше.

**Обоснование**

В экземпляре класса Act может передаваться информация о том, что три часа назад у пациента произошел инфаркт миокарда правого желудочка (см. описание атрибута Act.effectiveTime), но информация об этом необычном случае поступила только несколько минут назад (значение атрибута Act.availabilityTime). Соответственно, с момента Act.effectiveTime до момента Act.availabilityTime любые вмешательства, скорее всего, предпринимались, исходя из предположения о более распространенном инфаркте миокарда левого желудочка. Новое знание может объяснить, почему эти вмешательства (например, назначение нитрата) могли оказаться не подходящими.

**Примечания к конструированию**

Следует уточнить, должно ли быть указано новое время доступности при каждой передаче информации о действии? Указывает ли это значение на принадлежность к системе? Или же оно всегда определено как время доступности для передающей системы в контексте сообщения, любая следующая передача или не включает его, или перезаписывает его, а при необходимости предыдущие времена передачи передаются в отдельных экземплярах класса Observation.

В удаленном тексте сказано, что время доступности «приписано автору, который описал действие или ссылается на действие». Непонятно, почему для этого атрибута должны требоваться особые правила распространения, отличающиеся от тех, которым подчиняются другие атрибуты?

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.14 Act.priorityCode:: SET<CE> (0..*)

Словарный домен: ActPriority

**Определение**

Код или последовательность кодов (описывающих, например, обычную или экстренную срочность действия), указывающих срочность, с которой действие случилось, может случиться, происходит, планируется или требуется.

**Примечания к использованию**

Этот атрибут используется в направлениях для обозначения требуемой срочности действия, а в документации о совершенном действии он указывает фактическую срочность действия. В экземплярах класса Act, у которых атрибут moodCode имеет значение «DEF», в этом атрибуте указаны разрешенные значения срочности.

***Примеры — Обычный, необязательный, экстренный.***

A.3.1.2.15 Act.confidentialityCode:: SET<CE> (0..*)

Словарный домен: Confidentiality

**Определение**

Код, контролирующий раскрытие информации о данном действии независимо от значения атрибута moodCode.

**Примечания к использованию**

Важно отметить, что необходимая конфиденциальность медицинской карты не может быть достигнута только с помощью кодов конфиденциальности, предназначенных для маскировки отдельных частей этой карты от определенных типов пользователей. При обеспечении конфиденциальности на основе кодов конфиденциальности, присваиваемых отдельным частям карты, возникают две проблемы: одна связана с возможностью логического вывода, а другая — с риском отсутствия доступа к информации, которая может быть критичной в определенных случаях оказания медицинской помощи. Возможность вывода означает, что фильтрованная чувствительная информация может быть выведена из другой, не отфильтрованной информации. Простейшей формой вывода может служить примером, когда из факта наличия направления на анализ иммуноблота или на анализ количества Т4- и Т8-клеток с большой вероятностью можно сделать вывод, что пациент ВИЧ-инфицирован, даже если результат анализа неизвестен. Очень часто диагнозы могут быть выведены из лекарственных назначений, например, назначение зидовудина свидетельствует о лечении ВИЧ. Проблема сокрытия отдельных частей медицинской карты становится особенно трудной при наличии текущих лекарственных назначений, поскольку должно быть обеспечено продолжение приема лекарств.

Чтобы ослабить некоторые риски вывода, следует считать, что агрегированные данные имеют уровень конфиденциальности самого конфиденциального действия в агрегате.

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.16 Act.repeatNumber:: IVL<INT> (0..1)

**Определение**

Диапазон целых чисел, задающий минимальное и максимальное число повторений действия.

**Примечания к использованию**

Этот атрибут принадлежит к совокупности атрибутов управления рабочим процессом.

Число повторений дополнительно ограничивается временем. Действие будет повторяться по меньшей мере минимальное число раз и самое большее максимальное число раз. Повторения также закончатся, если время превысит максимальное значение атрибута Act.effectiveTime, если это случится раньше.

В экземпляре класса Act, у которого атрибут moodCode имеет значение «EVN» (event — событие), значение атрибута repeatNumber обычно равно 1. Если оно превышает 1, то этот экземпляр представляет собой сводную информацию о нескольких событиях, произошедших в течение интервала времени, указанного в атрибуте effectiveTime.

Чтобы различать экземпляры однотипных повторяющихся действий, используйте атрибут ActRelationship. sequenceNumber.

***Пример — После удаления зуба хирург-стоматолог может дать следующий совет пациенту: «Меняйте тампон каждый час от одного до трех раз, пока кровотечение не остановится полностью». Этот совет преобразуется в значение атрибута repeatNumber с нижней границей 1 и верхней границей 3.***

A.3.1.2.17 Act.interruptibleInd:: BL (0..1)

**Определение**

Признак, указывающий, может ли действие прерываться асинхронными событиями.

**Примечания к использованию**

Этот атрибут принадлежит к совокупности атрибутов управления рабочим процессом. Активные действия могут быть прерваны разными способами. Различаются следующие события прерывания: 1) когда получено прямое требование прекращения действия; 2) когда истекло время, выделенное для выполнения этого действия (таймаут); 3) когда «условие» выполнения действия перестает быть истинным (см. описание атрибута ActRelationship. checkpointCode); 4) когда экземпляр класса Act является компонентом, у которого атрибут joinCode имеет значение «K» (kill — прекращение) и все другие компоненты этой же группы завершены (см. описание атрибута Act.joinCode), и 5) когда прерывается объемлющий экземпляр класса Act.

Если действие получило прерывание и само оно может быть прервано, но в настоящее время имеет активные компоненты, которые не могут быть прерваны, то действие будет прервано тогда, когда все его активные непрерываемые компоненты будут завершены.

A.3.1.2.18 Act.levelCode:: CE (0..1)

Словарный домен: ActContextLevel

**Определение**

Код, определяющий уровень в иерархической структуре составного действия и тип контекста составных действий («контейнеров»), распространяемый на компоненты действия в пределах этих контейнеров. Значение атрибута levelCode обозначает положение в этой иерархии включения и применяемые ограничения.

**Ограничение использования**

Ограничения, применимые к специфическому уровню, могут включать разные требования к участникам (например, к пациенту, организации-источнику, автору или другому лицу, подписывающему данные), к ассоциациям или включениям других экземпляров класса Act, документам или использованию шаблонов. Ограничения, применимые к уровню, могут также определить допустимые уровни экземпляров, которые могут быть компонентами этого уровня. Несколько вложенных уровней с тем же самым значением атрибута levelCode могут быть допустимыми, запрещенными (или ограниченными). Экземпляры класса Act следующего подчиненного уровня обычно разрешены на каждом уровне, но некоторые уровни могут быть опущены в модели, и допускается пропустить несколько уровней.

**Примечания к использованию**

Понятия уровня определены в целях удовлетворения специфичных требований к передаче медицинских карт. Хотя эти понятия и применимы к некоторым другим типам транзакций, они не образуют полностью закрытый список. Существуют варианты других наборов ортогональных уровней, которые должны удовлетворять деловым требованиям (например, сообщения о нескольких пациентах можно подразделить с помощью вышестоящего уровня предметных областей).

**Примечания к конструированию**

Этот признак может быть объявлен «устаревающим»: непохоже, чтобы им пользовались.

***Пример — Экземпляры класса Act, находящиеся на «уровне выписки из медицинской карты» (значение атрибута Act.levelCode равно «EXTRACT») и «уровне папки» (значение атрибута Act.levelCode равно «FOLDER»), должны содержать данные о единственном лице, в то время как на «уровне нескольких субъектов» эти экземпляры могут содержать данные более чем об одном лице. В то время как «выписка из медицинской карты» может быть сделана из нескольких источников, «папка» должна содержать данные из одного источника. Уровень «композиции» (значение атрибута Act.levelCode равно «COMPOSITION») обычно имеет единственного автора.***

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.19 Act.independentInd:: BL (0..1)

**Определение**

Признак, указывающий, можно ли управлять данным экземпляром класса Act независимо от других экземпляров или управление этим экземпляром возможно только из вышестоящего составного действия, для которого данный экземпляр является компонентом.

**Примечания к использованию**

По умолчанию атрибут independentInd должен иметь значение «true». Определению действия иногда присваивается значение атрибута independentInd, равное «false», если деловые правила не разрешают назначать это действие, не назначая группу действий, которая его содержит.

***Пример — Назначение может иметь компонент, который нельзя прервать независимо от других компонентов.***

A.3.1.2.20 Act.uncertaintyCode:: CE (0..1)

Словарный домен: ActUncertainty

**Определение**

Код, указывающий, было ли в целом утверждение, передаваемое в экземпляре класса Act, объявлено как недостаточно точное.

**Примечания к использованию**

Отсутствие точности, объявленное с помощью этого атрибута, относится к объединенному смыслу утверждения, передаваемого в экземпляре класса Act с помощью всех описательных атрибутов (например, Act.code, Act.effectiveTime, Observation.value, SubstanceAdministration.doseQuantity и т. д.), и к смыслу всех компонентов. Этот атрибут не предназначен для замены или конкуренции с отсутствием точности значения атрибута Observation.value или других отдельных атрибутов класса. Такие точечные указания отсутствия точности должны быть определены с помощью расширения типов данных PPD, UVP или UVN, применяемых к конкретному атрибуту. В частности, если отсутствие точности относится к значению количественного измерения, то его надо указать с помощью присваивания этому значению типа данных PPD<PQ>, а не с помощью атрибута uncertaintyCode. Если, к примеру, дифференциальные диагнозы перенумерованы или им присвоены вероятностные веса, то надо использовать типы данных UVP<CD> или UVN<CD>, а не атрибут uncertaintyCode. Использование атрибута uncertaintyCode возможно только в том случае, если точность всего действия и зависящих от него действий подвергаются сомнению.

Между атрибутом неопределенности uncertaintyCode и атрибутом отрицания negationInd нет никакой связи. Можно быть очень неуверенным в том, что событие имело место, но это не означает уверенности в его отрицании.

***Пример — Пациенту могли в прошлом сделать операцию холецистэктомии (однако он в этом не уверен): имеет место неопределенность. Пациент отрицает, что в прошлом ему делали холецистэктомию: неопределенности нет.***

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.21 Act.reasonCode:: SET<CE> (0..*)

Словарный домен: ActReason

**Определение**

Код, указывающий мотивацию, причину или логическое обоснование действия, если такое обоснование не было представлено с помощью ассоциации ActRelationship, у которой атрибут typeCode имеет значение «RSON» (has reason — имеет причину) и которая связывает данное действие с другим.

**Примечания к использованию**

Большинство причин действий могут быть четко описаны с помощью связывания нового действия с предшествующим, используя ассоциацию ActRelationship, у которой атрибут typeCode имеет значение «RSON». Такая связь означает, что предшествующее действие служит причиной для нового (см. описание класса ActRelationship). Это предшествующее действие может быть специфичным существующим действием или текстовым разъяснением. Такой подход пригоден для большинства случаев, и чем более специфична причина, тем более следует использовать ассоциацию ActRelationship, а не атрибут reasonCode.

Атрибут reasonCode остается как место для указания общих причин, которые не связаны с предшествующим действием или любыми другими условиями, выраженными с помощью экземпляров класса Act. Примером могут служить указания, что таковы требования закона или что причиной послужил запрос пациента и т. д. Но если требуется более точно сослаться на конкретную статью закона, правила, контракта или запроса пациента, то надо их представить в форме экземпляра класса Act (обычно так и делается) и не использовать атрибут reasonCode.

***Пример — Примерами причин, которые могли бы заслуживать передачи в этом поле, служат «обычное назначение», «требование сообщить об инфекционном заболевании», «по запросу пациента», «требование закона».***

A.3.1.2.22 Act.languageCode:: CE (0..1)

Словарный домен: HumanLanguage

**Определение**

Основной язык, на котором описано данное действие, в особенности язык значения атрибута Act.text.

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.1.2.23 Переходы состояний экземпляра класса Act (атрибутом состояния является statusCode)

Диаграмма перехода состояний класса Act приведена на рисунке А.5. Действие может иметь следующие состояния:

- aborted (прервано) — подсостояние состояния normal: активный объект услуги был неожиданно завершен;
- active (активно) — подсостояние состояния normal: объект услуги активен;
- cancelled (отменено) — подсостояние состояния normal: объект услуги был отменен до того, как стал активным;
- completed (завершено) — подсостояние состояния normal: объект услуги завершен;
- held (отложено) — подсостояние состояния normal: объект услуги, все еще находящийся на подготовительной стадии. Он не может стать активным, пока не будет выведен из этого состояния;
- new (новое) — подсостояние состояния normal: объект услуги, который готовится стать активным;
- normal (нормальное) — охватывает все ожидаемые состояния объекта услуги, за исключением nullified и obsolete, которые представляют необычные терминальные состояния жизненного цикла;
- nullified (аннулировано) — объект услуги не должен был создаваться, поэтому он аннулирован;
- obsolete (устарело) — объект услуги заменен новым объектом;
- suspended (приостановлено) — подсостояние состояния normal: объект активной услуги временно приостановлен.

Между состояниями действия возможны следующие переходы:

- abort (прервать) — из состояния active в состояние aborted;
- revise (пересмотреть) — из состояния active в состояние active;
- complete (завершить) — из состояния active в состояние completed;
- suspend (приостановить) — из состояния active в состояние suspended;
- reactivate (активировать заново) — из состояния completed в состояние active;
- revise (пересмотреть) — из состояния completed в состояние completed;
- cancel (отменить) — из состояния held в состояние cancelled;
- revise (пересмотреть) — из состояния held в состояние held;
- release (освободить) — из состояния held в состояние new;
- activate (активировать) — из состояния new в состояние active;
- cancel (отменить) — из состояния new в состояние cancelled;
- complete (завершить) — из состояния new в состояние completed;
- hold (задержать) — из состояния new в состояние held;
- revise (пересмотреть) — из состояния new в состояние new;
- nullify (аннулировать) — из состояния normal в состояние nullified;
- obsolete (сделать устаревшим) — из состояния normal в состояние obsolete;
- activate (активировать) — из начального (пустого) состояния в состояние active;
- complete (завершить) — из начального (пустого) состояния в состояние completed;
- create (создать) — из начального (пустого) состояния в состояние new;
- jump (перейти) — из начального (пустого) состояния в состояние normal;
- abort (прервать) — из состояния suspended в состояние aborted;
- resume (возобновить) — из состояния suspended в состояние active;
- complete (завершить) — из состояния suspended в состояние completed;
- revise (пересмотреть) — из состояния suspended в состояние suspended.

A.3.1.3 Класс: ActRelationship (в предметной области Acts)

Атрибуты класса ActRelationship:

- typeCode:: CS,
- inversionInd:: BL,
- contextControlCode:: CS,
- contextConductionInd:: BL,
- sequenceNumber:: INT,
- priorityNumber:: INT,
- pauseQuantity:: PQ,
- checkpointCode:: CS,
- splitCode:: CS,
- joinCode:: CS,
- negationInd:: BL,

- conjunctionCode:: CS,
- localVariableName:: ST,
- seperatableInd:: BL,
- subsetCode :: CS,
- uncertaintyCode :: CE.

Ассоциации класса ActRelationship:

- source :: (1..1) Act :: outboundRelationship :: (0..*) (ассоциация с классом Act, роль outboundRelationship — исходящая связь),
- target :: (1..1) Act :: inboundRelationship :: (0..*) (ассоциация с классом Act, роль inboundRelationship — входящая связь).

Класс ActRelationship является специализацией класса InfrastructureRoot.

**Определение класса ActRelationship**

Класс ActRelationship моделирует направленную ассоциацию между классом-источником Act и классом-целью Act.

**Примечания к использованию**

Класс ActRelationship используется для представления планов действий и клинических выводов или суждений о связи действий. Предшествующие действия могут быть связаны с вновь появившимися в качестве их причин. Известные факты могут быть связаны с текущими клиническими гипотезами в качестве их доказательства. Списки диагнозов и другие совокупности связанных суждений о клинических событиях могут быть образованы с помощью экземпляров класса ActRelationship.

Каждый экземпляр класса ActRelationship можно рассматривать как стрелку с наконечником (упирающимся в целевой объект) и хвостовиком (упирающимся в объект-источник). Функции (иногда называемые «ролями»), которые выполняют экземпляры класса Act при таком связывании, определяются для каждого типа экземпляров класса ActRelationship по-разному. Когда экземпляр класса ActRelationship обеспечивает связь «композиция», то источник является составным объектом, а цель — его компонентом. Когда экземпляр класса ActRelationship обеспечивает причинно-следственную связь, то источником может быть любой экземпляр класса Act, а целью — экземпляр класса Act, описывающий причину появления источника.

Связи, ассоциированные с экземпляром класса Act, следует рассматривать как свойства экземпляра-источника. Это означает, что автор экземпляра класса Act считается автором всех связей с этим экземпляром, имеющих его в качестве источника. Из этого правила нет исключений.

Более подробные сведения о различных типах экземпляров класса ActRelationship см. в описании атрибута ActRelationship.typeCode.

***Примеры — отношений «имеет компонент», «выполняется по направлению», «имеет причину».***

***1 Компонентами панели электролитов являются натрий, калий, pH и бикарбонат. Тогда экземпляр класса Act, описывающий панель электролитов, имел бы четыре «исходящие» ассоциации с экземплярами классов ActRelationship, у которых атрибут ActRelationship.typeCode имеет значение «COMP» (has component — имеет компонент).***

***2 Панель электролитов исследуется по направлению на лабораторный анализ. Тогда экземпляр класса Act, содержащий результат исследования, имел бы «исходящую» ассоциацию с экземпляром класса ActRelationship, у которого атрибут ActRelationship.typeCode имеет значение «FLFS» (fulfills — выполняет) и который имеет «входящую» ассоциацию с экземпляром класса Act, содержащим направление на анализ.***

***3 Операция «холецисэктомия» выполнена в связи с результатом исследования «желчекаменная болезнь». Тогда экземпляр класса Act, содержащий сведения об этой операции, имел бы «исходящую» ассоциацию с экземпляром класса ActRelationship, у которого атрибут ActRelationship.typeCode имеет значение «RSON» (has reason — имеет причину) и который имеет «входящую» ассоциацию с экземпляром класса Observation, в котором указан диагноз «желчекаменная болезнь».***

**Атрибуты класса ActRelationship**

A.3.1.3.1 ActRelationship.typeCode:: CS (1..1) Mandatory

Словарный домен: ActRelationshipType

**Определение**

Код, указывающий смысл и назначение каждого экземпляра класса ActRelationship.

**Примечания к использованию**

Класс ActRelationship используется для представления множества семантических структур, включая панели исследований, планы действий и представления клинических выводов или суждений. Предшествующие действия могут быть связаны с вновь появившимися в качестве их причин. Известные факты могут быть связаны с текущими клиническими гипотезами в качестве их доказательства. Списки диагнозов и другие совокупности связанных суждений о клинических событиях могут быть образованы с помощью экземпляров класса ActRelationship. С помощью атрибута typeCode можно указать специфичные ограничения типов экземпляров класса Act, которые могут быть связаны подобным образом.

Типы экземпляров класса ActRelationship попадают в одну из шести категорий:

1) Композиция или декомпозиция, с составным объектом (источником) и его компонентом (цель). Одним из наиболее часто используемых типов экземпляров класса ActRelationship является «has component» (имеет компоненты), описывающий композицию и декомпозицию действий, представленных экземплярами класса Act. Такой тип отношения позволяет описывать действия с разной степенью детализации.

Связь композиции («COMP») позволяет группировать действия в «панели», например, панели LYTES, CHEM12 или CBC, с помощью которых можно сделать групповой заказ нескольких рутинных лабораторных анализов. Некоторые группировки, к примеру CHEM12, представляются более случайными, другие — более обоснованными, например, измерение артериального давления естественным образом состоит из систолического и диастолического давления.

С помощью отношений композиции детали действия могут быть представлены на разных уровнях для разных целей, не требуя переупорядочения структуры иерархии классов Act. Это позволяет отображать разные точки зрения на один и тот же процесс деятельности. Например, с позиции платежной системы панель лабораторных анализов может считаться простой оплачиваемой услугой. С клинической позиции та же самая панель лабораторных анализов является совокупностью отдельных анализов независимо от того, как они были заказаны. Точка зрения лаборатории на эту панель может быть более детальной и включать в себя такие шаги, которые никогда не сообщаются клиницисту (центрифугирование, отбор, аликвотирование, использование определенных устройств и т. д.). Лабораторная точка зрения обеспечивает исчерпывающую спецификацию планов действий (которые могут быть автоматизированы). При составлении такой спецификации будут описаны все более и более вложенные подчиненные действия. Тем не менее действие остается тем же самым, только его детали скрываются с помощью декомпозиции до необходимого уровня.

2) Продолжение, например, долечивание, выполнение назначения, конкретизация, замена, преобразование и т. д., имеющие то общее, что источником и целью связи являются экземпляры класса Act принципиально того же типа, но отличающиеся значениями атрибута moodCode и других атрибутов, а также то, что цель связи существует до появления источника, и источник содержит ссылку на цель, а цель содержит обратную ссылку на этот источник.

3) Предусловие, триггер, причина, противопоказание, при которых условно выполняемое действие («дать аспирин») является источником связи, а условие или причина («если температура превысит определенный порог») — целью.

4) Постусловие, результат, цель или риск, при которых действие-источник имеет связь с результатом или целью.

5) Рабочий процесс. Отношения композиции и следования могут быть организованы в последовательности, чтобы формировать временные и условные (не временные) ряды планов действий (например, план лечения, критичные пути, протоколы клинических испытаний, протоколы медикаментозной терапии). Как в классе Act, так и в классе ActRelationship есть группа атрибутов, называемая «комплексом атрибутов рабочих процессов», с помощью которых можно формировать детальные спецификации планов выполняемых действий. Этими атрибутами являются:

- Act.repeatNumber,
- Act.interrubtibleInd,
- ActRelationship.sequenceNumber,
- ActRelationship.pauseQuantity,
- ActRelationship.checkpointCode,
- ActRelationship.splitCode,
- ActRelationship.joinCode.

С помощью атрибута ActRelationship.sequenceNumber можно упорядочить компоненты экземпляра класса Act в форме последовательной или параллельной коллекции, выражая логические ветвления, а также параллельные задачи (задачи, выполняемые в одно и то же время). С помощью атрибута ActRelationship.splitCode и ActRelationship.joinCode можно описывать выбор ветвей или параллельное выполнение задач.

С помощью атрибутов Act.activityTime и ActRelationship.pauseQty можно явным образом задавать время выполнения планируемых действий. С помощью атрибута Act.repeatNumber можно задать повторение действия (цикл), а с помощью атрибута Act.interruptibleInd можно указать, может ли данное действие быть прервано связанными с ним действиями.

С помощью экземпляров класса ActRelationship, у которых атрибут ActRelationship.typeCode имеет значение «PRCN» (has precondition — имеет предусловие), можно задавать условия выполнения шагов плана в зависимости от состояния или результата предшествующих действий. С помощью атрибута ActRelationhsip.checkpointCode можно указать, когда проверяется предусловие действия в процессе передачи управления. Дополнительную информацию см. в описании отдельных атрибутов данной модели.

С помощью экземпляров класса ActRelationship, описывающих отношения композиции, можно организовывать многие уровни вложения, позволяющие полностью обеспечить управление рабочими процессами. Такое вложение и такое использование атрибутов рабочих процессов сконструировано по аналогии с конструкциями языков структурного программирования, поддерживающих параллелизм (ветвление, соединение, прерывания) и не требующих применения операторов перехода GOTO. Важно отметить, что все планы описываются с помощью

упорядоченных компонентов (шагов) составного действия (блока) в соответствии с принципами структурного программирования.

6) Множество функциональных связей, например поддержка, причина, производное, обобщаемых понятием «соответствие».

A.3.1.3.2 ActRelationship.inversionInd:: BL (0..1)

**Определение**

Признак, указывающий, что значение атрибута ActRelationship.typeCode должно быть интерпретировано таким образом, как если бы роли источника и целевых действий поменялись местами. Этот признак реверса используется, когда смысл значения атрибута ActRelationship.typeCode должен быть обращен.

**Примечания к конструированию**

Следует указать в аннотации значение по умолчанию и объяснить, почему для инверсии источника и цели целесообразно использовать признак.

A.3.1.3.3 ActRelationship.contextControlCode :: CS (0..1)

Словарный домен: ContextControl

**Определение**

Код, указывающий, какой вклад вносит данный экземпляр класса ActRelationship в контекст текущего экземпляра класса Act, и будет ли этот контекст распространяться на действия-потомки, чьи связи разрешают такое распространение (см. описание атрибута ActRelationship.contextConductionInd).

**Примечания к использованию**

С помощью этого атрибута можно точно указать, вносит ли ассоциация дополнение к контексту, связанному с конкретным элементом (например, добавляя дополнительного автора), или заменяет (отменяет) часть контекста, связанного с конкретным элементом (например, указывая единственного автора независимо от прежнего содержания элемента). С его помощью можно также указать, применяется ли ассоциация только к этому действию (не распространяемый контекст) или может применяться также и к производным действиям (распространяемый контекст).

Рассматриваемый атрибут тесно связан с атрибутом ActRelationship.contextConductionInd, указывающим, действительно ли информация будет распространяться на дочернее действие тем способом, который указан в атрибуте contextControlCode.

Если этот атрибут не имеет значения (например, у него пустое значение) или у него нет значения по умолчанию, то о контексте нельзя сделать никаких выводов. В системах могут использоваться собственные допущения на основе передаваемых данных. По этой причине подкомитетам комитета HL7 рекомендуется в своих спецификациях указывать для этого атрибута значение по умолчанию или фиксированное значение, чтобы гарантировать его согласованную интерпретацию.

**Обоснование**

При интерпретации информации люди нередко склонны полагаться на ее контекст. Например, читая документ, извлеченный из папки, содержащей медицинскую карту пациента, пользователь сделает вывод, что этот документ относится к данному пациенту, даже если в нем нет прямого указания на принадлежность к этому пациенту. Однако другие части информации, например автор папки (больница, в которой ведется эта медицинская карта), могут иногда применяться к содержанию папки (например, документ, составленный врачом этой больницы), а могут не применяться (например, в папку вложена копия документа, полученного из другого медицинского учреждения). Люди вполне хорошо делают необходимые выводы о том, какая часть контекста должна распространяться на элемент данных, а какая — нет. Однако случаются и неправильные выводы (например, в медицинскую карту пациента вложен документ с информацией о его родственнике). Более того, автоматизация подобных выводов представляет собой очень сложную задачу, даже если такие выводы необходимы для систем обеспечения принятия решений.

**Примечания к конструированию**

В приведенном ниже примере возникает впечатление, что у события аптечного заказа тот же автор, что и у комплексного назначения.

***Примеры***

***1 Пусть экземпляр класса Observation имеет исходящую ассоциацию к экземпляру класса ActRelationship, содержащему информацию об участии пациента в исследовании, и при этом атрибут ActRelationship.contextControlCode имеет значение «AP» (additive, propagating — аддитивная, распространяемая). Пусть этот же экземпляр класса Observation имеет связи с компонентами исследования, описанные с помощью экземпляров класса ActRelationship, которые маркированы как распространяющие контекст. Это означает, что участие пациента, описанное для родительского экземпляра класса Observation, распространяется и на эти компоненты исследования.***

***2 Пусть создано комплексное назначение (1), содержащее заказ в аптеку (2), а также заказ нескольких лабораторных анализов (3). Это комплексное назначение имеет связи участия с пациентом (4) и автором назначения (5), а также связь с диагнозом (6), и каждая из этих связей маркирована как «аддитивная, распространяемая». Пусть связь «является компонентом» (7) между комплексным назначением (1) и входящим в него заказом в аптеку (2) маркирована как распространяющая связь (ее атрибут***

***contextConductionInd имеет значение «true»). При этом заказ в аптеку имеет связь участия с автором (8), маркированную как «AN» (additive, non-propagating — аддитивная, не распространяемая), и причинно-следственную связь с диагнозом (9), маркированную как «OP» (overriding, propagating — замещающая, распространяемая). Кроме того, заказ в аптеку (2) имеет связь с информацией об отпуске лекарства (10), а также связь с протоколом медикаментозной терапии (11), которая маркирована как не распространяемая (ее атрибут contextConductionInd имеет значение «false»).***

***Такая совокупность объектов и связей трактуется следующим образом: заказ в аптеку (2) интерпретируется как наследующий пациента (4) от комплексного назначения (1). Он имеет двух авторов (одного, унаследованного от комплексного назначения, и другого, явно указанного автора заказа в аптеку). Диагнозом, в связи с которым (9) сделан заказ в аптеку, будет считаться только тот, что связан с заказом в аптеку (2), а не тот (6), что связан с комплексным назначением. Событие отпуска (10) унаследует пациента (4) от комплексного назначения и диагноз заказа в аптеку (9), но не авторов. Протокол медикаментозной терапии (11) не будет связан ни с пациентом, ни с диагнозом, ни с автором.***

A.3.1.3.4 ActRelationship.contextConductionInd:: BL (0..1)

**Определение**

Если этот атрибут имеет значение TRUE, то связи родительского действия распространяются через данный экземпляр класса ActRelationship на дочернее действие.

**Примечания к использованию**

Обоснование и примеры указаны в описании атрибута ActRelationship.contextControlCode.

A.3.1.3.5 ActRelationship.sequenceNumber:: INT (0..1)

**Определение**

Целое значение, указывающее относительное положение данной связи среди других связей похожих типов, у которых источником является один и тот же экземпляр класса Act.

**Примечания к использованию**

Этот атрибут принадлежит к группе атрибутов управления рабочим процессом. План действия представляет собой составное действие, связанное с действиями-компонентами. В упорядоченном плане каждый экземпляр класса ActRelationship, связывающий составное действие с компонентом, имеет атрибут sequenceNumber, значение которого определяет порядок шагов плана. Если у нескольких компонентов значение атрибута sequenceNumber одинаково, то эти компоненты являются ветвями. Ветви могут быть исключающими (как в переключателе case) или могут допускать параллельное выполнение, что можно указать с помощью атрибута splitCode.

Если атрибут имеет пустое значение, то относительная позиция действия, описываемого целевым экземпляром класса Act, не задана (то есть оно может быть на любом месте).

Чтобы указать относительную очередность связанных действий, вместо атрибута sequenceNumber следует использовать атрибут priorityNumber.

A.3.1.3.6 ActRelationship.priorityNumber:: INT (0..1)

**Определение**

Целое значение, указывающее относительный приоритет данной связи среди других связей похожих типов, у которых источником является один и тот же экземпляр класса Act. Связи с меньшими значениями атрибута priorityNumber рассматриваются раньше и выше тех, что имеют более высокие значения.

**Примечания к использованию**

Если указано несколько критериев, то с помощью этого атрибута можно указать, какой из них должен быть рассмотрен раньше других. Если связи с компонентами имеют одно и то же значение атрибута sequenceNumber, то атрибут приоритета позволяет указать, какая из них должна быть рассмотрена раньше других. Если альтернативы или варианты выбираются людьми, то значение priorityNumber указывает предпочтение.

Упорядочение может быть полным, при котором все значения приоритета уникальны, или частичным, при котором один и тот же приоритет назначается нескольким связям.

A.3.1.3.7 ActRelationship.pauseQuantity:: PQ (0..1)

**Определение**

Промежуток времени между готовностью действия к выполнению и фактическим началом его выполнения.

**Примечания к использованию**

Этот атрибут принадлежит к группе атрибутов управления рабочим процессом. План действия представляет собой составное действие, связанное с действиями-компонентами. В упорядоченном плане каждый экземпляр класса ActRelationship, связывающий составное действие с компонентом, имеет атрибут sequenceNumber, значение которого определяет порядок шагов плана. Если у шага есть предусловия, то его выполнение инициируется в том случае, когда они удовлетворяются. В этот момент запускается таймер со значением pauseQuantity, и действие начинает выполняться, когда пройдет время, указанное в атрибуте pauseQuantity.

В качестве предусловия (например, применить за 3 часа до хирургической операции) значение атрибута pauseQuantity может быть отрицательным при условии, что можно предсказать событие, описанное целевым условием.

**Формальное ограничение**

Единицами измерения должны быть единицы времени.

A.3.1.3.8 ActRelationship.checkpointCode:: CS (0..1)

Словарный домен: ActRelationshipCheckpoint

**Определение**

Код, указывающий моменты проверки выполнения предусловия действия, (например, перед тем как действие начнется впервые, после каждого повторения действия, но не перед первым, или в процессе всего времени действия).

**Примечания к использованию**

Этот атрибут принадлежит к группе атрибутов управления рабочим процессом. План действия представляет собой составное действие, связанное с действиями-компонентами. В упорядоченном плане каждый экземпляр класса ActRelationship, связывающий составное действие с компонентом, имеет атрибут sequenceNumber, значение которого определяет порядок шагов плана. Если у шага есть предусловия, то его выполнение инициируется в том случае, когда они удовлетворяются. С помощью атрибута repeatNumber можно указать, что выполнение действия может повторяться. А с помощью атрибута checkpointCode можно указать, когда проверяется предусловие, что аналогично различным условным операторам и циклам в языках программирования: while-do по сравнению с do-while или repeat-until по сравнению с loop-exit.

Для всех значений атрибута checkpointCode, кроме «E» (end — конец), предусловия проверяются в момент завершения предшествующего шага плана при условии, что данный шаг является следующим согласно значению атрибута sequenceNumber.

Если атрибут checkpointCode критерия повторяющегося действия имеет значение «E» (end — конец), то этот критерий проверяется только в конце каждого повторения действия. Если критерий повторения удовлетворен, то следующее повторение действия готово к выполнению.

Если атрибут checkpointCode имеет значение «S» (entry — вход), то критерий проверяется в начале каждого повторения (если таковые имеются), при этом «начало» означает, что критерий проверяется однократно при старте «циклического» повторения.

Если атрибут checkpointCode имеет значение «T» (through — в течение), то оно задает особый случай, когда критерий проверяется в процессе повторения. Как только критерий перестал выполняться, должно быть инициировано событие прерывания действия (см. описание атрибута Act.interruptibleInd) и в принципе действие должно быть завершено.

Атрибут checkpointCode со значением «X» (exit — выход) используется только в специальном шаге плана, представляющем выход из цикла. С его помощью можно обеспечить завершение плана действий в связи с выполнением определенного условия, проверяемого при выполнении этого плана. Такие критерии выхода упорядочены относительно других компонентов плана с помощью атрибута ActRelationship.sequenceNumber.

A.3.1.3.9 ActRelationship.splitCode:: CS (0..1)

Словарный домен: ActRelationshipSplit

**Определение**

Код, указывающий, какие ветви плана действия выбираются среди других ветвей.

**Примечания к использованию**

Этот атрибут принадлежит к группе атрибутов управления рабочим процессом. План действия представляет собой составное действие, связанное с действиями-компонентами. В упорядоченном плане каждый экземпляр класса ActRelationship, связывающий составное действие с компонентом, имеет атрибут sequenceNumber, значение которого определяет порядок шагов плана. Если для нескольких компонентов значение атрибута sequenceNumber одинаково, то эти компоненты являются ветвями. Атрибут splitCode указывает, является ли ветвь исключающей (как в переключателе case) или включающей, то есть может допускать параллельное выполнение других ветвей.

В дополнение к исключающему и включающему ветвлению с помощью атрибута splitCode можно указать, как проверяется предусловие (называемое также «сторожевым условием») ветви. Сторожевое условие может проверяться однократно при переходе к ветви, и если оно не выполнено, то ветвь отвергается. В другом варианте выполнение ветви может быть отложено, пока это условие не будет выполнено.

При исключающем ветвлении с ожиданием первая ветвь, у которой ее условие ветвления станет выполненным, начнет выполняться, а все остальные ветви будут отвергнуты. При включающем ветвлении с ожиданием некоторые ветви могут уже выполняться, в то время как другие все еще будут ожидать, пока их сторожевое условие не будет выполнено.

***Пример — исключающее ветвление с ожиданием, включающее ветвление с ожиданием, исключающее ветвление с однократной проверкой.***

A.3.1.3.10 ActRelationship.joinCode:: CS (0..1)

Словарный домен: ActRelationshipJoin

**Определение**

Код, указывающий способ восстановления синхронизации параллельно выполняемых действий.

**Примечания к использованию**

Этот атрибут принадлежит к группе атрибутов управления рабочим процессом. План действия представляет собой составное действие, связанное с действиями-компонентами. В упорядоченном плане каждый экземпляр класса ActRelationship, связывающий составное действие с компонентом, имеет атрибут sequenceNumber, значение которого определяет порядок шагов плана. Если для нескольких компонентов значение атрибута sequenceNumber одинаково, то эти компоненты являются ветвями. Ветви могут выполняться параллельно, если атрибут splitCode указывает, что в одно и то же время может исполняться более одной ветви. В этом случае с помощью атрибута joinCode можно указать, будет ли восстанавливаться синхронизация ветвей, и если да, то каким образом.

Основные способы восстановления синхронизации следующие:

- поток управления ждет, пока выполнение каждой ветви не завершится (ожидание ветвей);
- как только выполнится одна ветвь, выполнение остальных ветвей прекращается (прекращение ветвей);
- синхронизация ветвей не восстанавливается, и они продолжают выполняться (отложенные ветви).

Прекращение ветвей происходит только в том случае, когда имеется как минимум одна активная ожидающая (или исключающая ожидающая) ветвь. Если активных ожидающих ветвей нет, то процесс прекращения ветвей не инициируется (а не прекращается после инициации). Поскольку отложенная ветвь не связана с другими ветвями, наличие активных отложенных ветвей не мешает прекращению другой ветви.

***Пример — отложенная ветвь, прекращение ветвей, исключающая ожидающая ветвь.***

A.3.1.3.11 ActRelationship.negationInd:: BL (0..1)

**Определение**

Признак, указывающий, что к значению связи применяется отрицание.

**Примечания к использованию**

Данный атрибут используется главным образом в разъяснительных целях. Как показывают примеры, его использование довольно ограничено, особенно по сравнению с атрибутом Act.negationInd, с помощью которого фактически можно указать, что описанное действие не существует, не выполняется и т. д., в то время как с помощью атрибута ActRelationship.negationInd можно лишь отрицать данную связь между действием-источником и действием — целью действия, а не изменить смысл каждого действия.

Следует учитывать также различие между отрицанием и противоположностью. Противопоказание является противоположностью показания (причины). То обстоятельство, что наличие боли в пояснице не является причиной назначения антибиотиков, не означает, что боль в пояснице является противопоказанием для применения антибиотиков.

***Пример — Если исходная связь указывает, что у действия A есть компонент B, то ее отрицание означает, что действие B не является компонентом действия A. Если действие B описывает причину действия A, то отрицание означает, что действие B не является причиной действия A. Если действие B описывает предусловие действия A, то отрицание означает, что действие B не является предварительным условием действия A.***

A.3.1.3.12 ActRelationship.conjunctionCode:: CS (0..1)

Словарный домен: RelationshipConjunction

**Определение**

Код, указывающий логическое соединение критериев, описанных условными связями между действиями [например, AND («и»), OR («или»), XOR («исключающее или»)].

**Примечания к использованию**

Этот атрибут используется для передачи критериев, обычно применяемых к действиям определения или цели.

Все критерии, соединенные с помощью AND, должны выполняться. Если в одну группу входят критерии, соединяемые с помощью OR и AND, то должны выполняться хотя бы один из критериев OR и все критерии AND. Если в одну группу входят критерии, соединяемые с помощью XOR, OR и AND, то должны выполняться ровно один критерий XOR, по крайней мере один из критериев OR и все критерии AND. Другими словами, множества критериев AND, OR и XOR, в свою очередь, объединены логическим оператором AND (все критерии AND, по крайней мере один из критериев OR и ровно один критерий XOR). Если требуется иное, то можно воспользоваться вложенными критериями.

A.3.1.3.13 ActRelationship.localVariableName:: ST (0..1)

**Определение**

Строка символов, указывающая имя входного параметра, по значению которого экземпляр класса Act, который служит источником ассоциации с данным экземпляром класса ActRelationship, может вычислить некоторые из своих атрибутов. Областью действия имени локальной переменной является значение атрибута Act.derivationExpr, который может содержать это имя в качестве входного параметра.

A.3.1.3.14 ActRelationship.seperatableInd:: BL (0..1)

**Определение**

Этот признак указывает, должно ли действие — источник связи интерпретироваться независимо от целевого действия связи.

**Примечания к использованию**

Значение этого признака не может препятствовать человеку или приложению обрабатывать действия независимо друг от друга, оно лишь обозначает желание и готовность автора заверить содержание действия-источника, если оно будет отделено от содержания целевого действия. Следует иметь в виду, что типичным значением этого атрибута по умолчанию будет TRUE. Следует также учесть, что этот атрибут ортогонален механизму наследования контекста и никак с ним не связан. Если контекст действия распространен на вложенные действия, то тем самым предполагается, что эти вложенные действия не могут интерпретироваться без распространенного контекста.

A.3.1.3.15 ActRelationship.subsetCode:: CS (0..1)

Словарный домен: ActRelationshipSubset

**Определение**

Код, указывающий, что целевым действием связи является фильтрованное подмножество общего связанного множества целевых действий.

**Примечания к использованию**

Используется для ограничения числа компонентов до подмножества, состоящего из первого, последнего или следующего компонента, а также из суммарного множества, усредненного множества или иным образом отфильтрованного либо вычисленного подмножества.

***Пример — первый, максимальный, суммарный.***

A.3.1.3.16 ActRelationship.uncertaintyCode:: CE (0..1)

Словарный домен: ActUncertainty

**Определение**

Код, указывающий, является ли связь между действием-источником и целевым действием неопределенной.

**Примечания к использованию**

Неопределенность, указываемая значением этого атрибута, относится только к связи между двумя действиями. Степень определенности самих действий должна задаваться с помощью атрибута Act.uncertaintyCode.

***Пример — Подозревается конкретное событие воздействия, но при этом нет уверенности, что оно вызвало конкретный симптом. Связь между ними описывается как неопределенная.***

A.3.1.4 Класс ControlAct (в предметной области Acts)

Ассоциации класса ControlAct:

- payload::(0..*) Message::controlAct::(0..1) (ассоциация с классом Message, роль controlAct — управляющее действие),
- queryEvent::(0..1) QueryEvent::controlAct::(1..1) (ассоциация с классом QueryEvent, роль controlAct — управляющее действие).

Класс ControlAct является специализацией класса Act.

**Определение класса ControlAct**

Действие, приводящее к изменению состояния другого класса, пользовательское событие (например, запрос) или системное событие (например, выполняемое по таймеру).

**Примечания к использованию**

Этот класс воплощает понятие «события, инициирующего взаимодействие» и поэтому должен присутствовать при каждом обмене сообщением (в силу кратности «1..1» между событием и взаимодействием). Однако управляющие действия могут также присутствовать в информации, передаваемой в сообщении. Примером может служить последовательность управляющих действий, связанных с лабораторным заказом и идентифицирующих события его обработки (вначале создание, затем пересмотр, приостановка, возобновление и, наконец, завершение).

***Примеры***

***1 Выписка пациента [состояние класса Encounter (контакт с пациентом) изменяется с Active (активный) на Completed (завершенный)].***

***2 Прекращение применения лекарственного средства [состояние класса SubstanceAdministration (применение лекарства) изменяется с Active (активное) на Aborted (прекращенное)].***

***3 Передача ежедневной сводки (событие, выполняемое по таймеру).***

A.3.1.5 Класс DeviceTask (в предметной области Acts)

**Свойства класса DeviceTask**

Атрибуты класса DeviceTask:

- parameterValue :: LIST<ANY>.

Класс DeviceTask является специализацией класса Act.

**Определение класса DeviceTask**

Активность автоматизированной системы.

**Примечания к использованию**

Такая активность возникает или по внешней команде или самопроизвольно планируется и выполняется устройством (например, регулярная калибровка или промывка). У экземпляра класса, содержащего команду

устройству, атрибут moodCode должен иметь значение «ORD» (order — заказ), у экземпляра с описанием выполнямого или выполненного действия атрибут moodCode должен иметь значение «EVN» (event — событие), у экземпляра с описанием автоматически инициируемой задачи атрибут moodCode должен иметь значение «APT» (выполнение по графику).

**Атрибуты класса DeviceTask**

A.3.1.5.1 DeviceTask.parameterValue :: LIST<ANY> (0..*)

**Определение**

Список параметров задачи, передаваемых устройству при выдаче команды (или используемых для планирования самопроизвольно выполняемых задач).

**Ограничение использования**

Список параметров может содержать любые значения, интерпретируемые устройством. Они должны иметь тип данных из числа определенных в стандарте HL7 (например, коды перечислимых значений, типы данных REAL и INT для чисел, TS для моментов времени, PQ для размерных физических величин и т. д.). Однако помимо типизации значений иные характеристики параметров являются закрытыми для стандартов HL7.

**Обоснование**

Некоторые параметры задач уникальны для конкретной модели оборудования. Наиболее критичные аргументы задачи (например, привести контейнер в движение, позиционировать его, выполнять действия по графику) указаны в структуре, стандартизованной комитетом HL7, и данный список параметров не должен использоваться для их передачи. Он нужен только в тех случаях, когда передаются не стандартизованные параметры, уникальные для конкретной модели оборудования.

Примечание — Тем самым следует, что семантика и интерпретация списка parameterValue могут быть осуществлены, только исходя из содержания спецификации данного устройства или его документации. Это содержание не является частью сообщения.

**Примечания к конструированию**

Понятие структуры, определенной или стандартизованной комитетом HL7, должно быть определено здесь или в словаре со ссылкой на него.

A.3.1.6 Класс DiagnosticImage (в предметной области Acts)

**Свойства класса DiagnosticImage**

Атрибуты класса DiagnosticImage:

- subjectOrientationCode :: CE.

Класс DiagnosticImage является специализацией класса Observation.

**Определение класса DiagnosticImage**

Исследование, результатом которого является пространственное представление физического субъекта, пригодное для визуализации.

**Атрибуты класса DiagnosticImage**

A.3.1.6.1 DiagnosticImage.subjectOrientationCode :: CE (0..1)

Словарный домен: ImagingSubjectOrientation

**Определение**

Код, характеризующий пространственное расположение исследуемого объекта по отношению к рентгеновской пленке или детектору.

A.3.1.7 Класс Diet (в предметной области Acts)

**Свойства класса Diet**

Атрибуты класса Diet:

- energyQuantity :: PQ,
- carbohydrateQuantity :: PQ.

Класс Diet является специализацией класса Supply.

**Определение класса Diet**

Действие кормления субъекта или предоставления ему питания.

**Примечания к использованию**

Детали диеты передаются в экземпляре класса Material, ассоциированного с классом Diet с помощью экземпляра класса Participation, у которого атрибут typeCode имеет значение «PRD» (product — товар). Номер диеты, имеющий медицинское значение, можно передать в атрибуте Diet.code, однако детали пищи, входящей в диету, и различные сочетания блюд должны передаваться в форме экземпляров класса Material.

**Примечания к конструированию**

Во введении должно быть оговорено, как используются документы или ограничения атрибутов, унаследованные от родительского класса, например, Diet.code.

***Примеры — Безглютеновая диета, бессолевая диета.***

**Атрибуты класса Diet**

A.3.1.7.1 Diet.energyQuantity:: PQ (0..1)

**Определение**

Предоставляемая биологическая энергия (калории) в день.

**Примечания к использованию**

Единицами измерения ДОЛЖНЫ быть «калории» (4,184 Дж), а не «большие калории», указываемые в этикетках на продуктах питания. Лучше использовать малую калорию, являющуюся 1/1000 большой калории. Это различие строго проводится в таблицах единиц измерения, включенных в стандарт HL7.

A.3.1.7.2 Diet.carbohydrateQuantity:: PQ (0..1)

**Определение**

Предоставляемое количество углеводов (грамм) в день.

**Примечания к использованию**

Для диабетической диеты типично ограничение количества усваиваемых углеводов в день (например, 240 г/д). Это ограничение может быть передано как значение атрибута carbohydrateQuantity.

**Примечания к конструированию**

В определении были указаны граммы (г), но это не должно быть ограничением для типа данных PQ.

A.3.1.8 Класс Exposure (в предметной области Acts)

**Свойства класса Exposure**

Атрибуты класса Exposure:

- routeCode :: CE,
- exposureLevel :: CE,
- exposureModeCode :: CE.

Класс Exposure является специализацией класса Act.

**Определение класса Exposure**

Воздействие одной сущности на другую, предполагающее возможность передачи физического, химического или биологического агента от сущности-источника к целевой сущности.

**Примечания к использованию**

Этот класс имеет дело только с возможностью, а не с результатом воздействия, то есть не все стороны, подвергшиеся воздействию, получат от него вред или пользу.

Воздействие отличается от применения лекарственного средства тем, что в нем отсутствует участие исполнителя.

Для различения специфичных сущностей ДОЛЖНЫ использоваться следующие виды участий:

- сущность, на которую оказано воздействие, должна участвовать в нем как «цель воздействия» [атрибут Participation.typeCode имеет значение «EXPTRGT» (exposure target)];

- сущность, которая переносит воздействующий агент, участвует в воздействии как «источник воздействия» [атрибут Participation.typeCode имеет значение «EXSRC» (exposure target)], например:

1) лицо или животное, являющееся переносчиком инфекционного заболевания (атрибут Participation. typeCode имеет значение «EXSRC») другому лицу или животному (атрибут Participation.typeCode имеет значение «EXPTRGT»);

2) место или иная окружающая среда (атрибут Participation.typeCode имеет значение «EXSRC») оказывает воздействие на лицо или животное (атрибут Participation.typeCode имеет значение «EXPTRGT»);

- если не установлено, является ли участвующая сущность источником или целью воздействующего агента, то атрибуту Participation.typeCode присваивается значение «EXPART» (exposure participant — участник воздействия);

- физическая (включая энергию), химическая или биологическая субстанция, участвующая в воздействии, характеризуется значением атрибута Participation.typeCode «EXPAGNT» (exposure agent — воздействующий агент). Существует по меньшей мере три сценария:

- исполнитель роли, участвующий как воздействующий агент, является химической или биологической субстанцией, которая смешивается или переносится контролирующей сущностью роли (например, исполняет роль ингредиента);

- исполнитель роли, участвующий как воздействующий агент, является смесью, содержащей химическую, радиационную или биологическую субстанцию (предмет интереса);

- исполнитель роли, участвующий как воздействующий агент, является переносчиком агента (например, потенциально зараженным предметом, насекомым и т. д.).

Значение атрибута Exposure.statusCode следует интерпретировать как состояние объекта, участвующего в воздействии (например, активный, прекращенный, завершенный), а не как клинический статус воздействия (например, вероятный или подтвержденный). Клиническое состояния воздействия должно ассоциироваться с ним с помощью класса Observation, относящегося к субъекту воздействия.

**Примечания к конструированию**

В примечаниях к использованию необходим четкий критерий, с помощью которого можно определить, является ли действие воздействием либо применением потенциально вредоносной субстанции, неопределенностью фактического переноса или относится к иной категории.

Чтобы отнести действие к категории применения субстанции, необходимо наличие исполнителя, но есть примеры, когда такой критерий оказывается под вопросом (например, первый пример, относящийся к ошибке дозировки).

*Примеры*

*Следующие примеры приведены, чтобы показать, какие взаимодействия считаются воздействиями, а не другими типами действий.*

*1 Вследствие ошибки дозировки пациент случайно получил дозу лекарства, в три раза превышающую рекомендованную.*

*Это применение субстанции. Органы санитарно-эпидемиологического надзора или фармаконадзора могут быть также заинтересованы в документировании этого факта и результата воздействия.*

*2 Пациенту случайно отпустили неправильное лекарство (например, кломифен вместо кломипрамина). Прежде чем ошибка была обнаружена, он принял несколько доз. Тем самым на него было оказано «воздействие», не имевшее терапевтических показаний к приему.*

*В этом примере несколько применений субстанции. Органы санитарно-эпидемиологического надзора или фармаконадзора могут быть также заинтересованы в документировании этого факта и результата воздействия.*

*3 В тесной палате пациенту проводили химиотерапию лимфомы. К несчастью, инфузионный мешок разорвался и цитотоксичное лекарство разбрызгалось на самого пациента и на его соседа.*

*В этом примере применение субстанции. Первое желательное (внутривенная инфузия) с его ассоциированным (подразумеваемым) воздействием. Оно дополняется инцидентом, при котором это же лекарство разбрызгалось на пациента и оказало на него еще одно воздействие. Кроме того, в результате этого же инцидента лекарство разбрызгалось на соседнего пациента и тоже оказало на него воздействие. Уязвимые пациенты (с пониженным иммунитетом), находившиеся рядом с ним в приемной, могли подвергнуться воздействию бацилл туберкулеза, и должны быть идентифицированы для последующего обследования.*

*4 Пациент, являющийся беженцем из охваченной войной африканской страны, прибыл в переполненное городское скоропомощное отделение. У него кашель с кровохарканием. Не понимая правил регистрации и сортировки, он просидел в приемной несколько часов, прежде чем на него обратили внимание. Как только им занялись, возникло подозрение на наличие у него туберкулеза.*

*Это воздействие (возможно, несколько воздействий), имевшее место в приемной, распространяется на беженца и всех, кто находился в приемной в этот период. При этом также могло происходить несколько известных или предположительных применений субстанций (в результате кашля) через несколько различных путей. Применение субстанций является лишь гипотетическим, пока не будет подтверждено последующими исследованиями.*

*5 Пациент, которому выполнили плановую операцию по полному эндопротезированию тазобедренного сустава, продолжительно пребывает в стационаре из-за инфекции места хирургической раны метициллин-резистентным золотистый стафилококком.*

*Это воздействие стафиллококком. Хотя это можно рассматривать как некоторый вид применения субстанции, может оказаться, что точный механизм попадания стафиллококка в рану не будет идентифицирован.*

*6 При плановом техническом обслуживании рентгеновских аппаратов в местной больнице было выявлено серьезное нарушение экранирования одного из аппаратов. Довольно вероятно, что пациенты, проходившие в последнем месяце исследование на этом аппарате, получили значительно более высокую дозу облучения и должны быть проконтролированы не предмет возможных побочных эффектов.*

*Это воздействие на каждого пациента, проходившего исследование на этом аппарате в течение последних 30 дней. Для некоторых пациентов могло иметь место применение субстанции.*

*7 Новый работник был принят в прачечную небольшой сельской больницы и неправильно прочел инструкции, в результате чего при стирке больничного постельного белья была использована концентрация моющего средства, в 50 раз превышающая обычную. В результате несколько пациентов подверглись воздействию моющих средств, оставшихся в «чистом» белье, и испытали дерматологическую реакцию.*

*Здесь имело место несколько воздействий на нескольких пациентов. Хотя к кожным покровам пациентов была применена субстанция, такое применение субстанции не было непосредственно документировано.*

*8 У семерых пациентов психиатрической клиники для престарелых выявились респираторные проблемы. После нескольких месяцев, в течение которых им выполнялись различные анализы и назначались различные лекарственные средства, выяснилось, что причиной проблем оказалась их «чувствительность» к новому фунгициду, которым была обработана штукатурка стен их палаты.*

*Пациенты подвергались непрерывному воздействию фунгицида. Хотя имело место применение субстанции (через органы дыхания), это не было документировано как таковое.*

*9 Пациенту с остеоартритом колен проводилось симптоматическое лечение, используя обезболивание парацетамолом (ацетаминофеном) 1 г до четырех раз в день. Но его врач общей практики не знал, что 20 лет назад (будучи в колледже) пациент сильно злоупотреблял алкоголем и хотя теперь он полностью контролирует себя, его печень серьезно пострадала, в результате чего парацетамол оказывает на нее токсичное влияние. Через некоторое время у пациента развилась желтуха. Парацетамол был немедленно отменен и для обезболивания остеоартрита было предложено другое лечение. С помощью консервативного лечения желтуха значительно ослабла, но для консультирования и контроля требуется направление к гастроэнтерологу.*

*Это применение субстанции с ассоциированным воздействием. Компонент воздействия связан с дозировкой субстанции, относительно токсичной для пациента с пониженной функцией печени.*

*10 Пациент, выписанный из местной больницы после срочной аппендэктомии, обратился к своему врачу общей практики с жалобой на абдоминальную боль. Врач не нашел ничего необычного и предположил, что это послеоперационная боль, которая должна будет пройти. Через две недели пациент снова обратился к врачу, тот прописал ему обезболивающее, но при этом решил направить его на консультацию к хирургу. По результатам консультации медрегистратор оформил пациенту направление на ультразвуковое исследование, которое было выполнено три недели спустя и выявило наличие непонятной небольшой массы в абдоминальной полости. В дневном стационаре пациенту сделали лапароскопию и извлекли из абдоминальной полости кусочек хирургического тампона. К счастью, пациент полностью восстановился.*

*Это осложнение процедуры. Оно может быть также зарегистрировано как инцидент.*

*11 Пациент немного опоздал на регулярную проверку батарейки имплантированного ему водителя ритма, выполняемую в отделении кардиологии местной больницы. Он спешил спуститься в коридор второго этажа. Внезапный летний шквал пронесся над больницей, струи дождя попали через открытое коридорное окно на пол, пациент поскользнулся на лужице и упал так сильно, что его пришлось отнести в отделение скорой и неотложной помощи, где было диагностировано небольшое растяжение крестообразной связки левого колена.*

*Это не воздействие, а инцидент.*

A.3.1.9 Класс FinancialContract (в предметной области Acts)

**Свойства класса FinancialContract**

Атрибуты класса FinancialContract:

- paymentTermsCode:: CE.

Класс FinancialContract является специализацией класса Act.

**Определение класса FinancialContract**

Контракт, имеющий денежное выражение.

***Пример — Договор страхования, соглашение о закупках.***

**Атрибуты класса FinancialContract**

A.3.1.9.1 FinancialContract.paymentTermsCode:: CE (0..1)

Словарный домен: PaymentTerms

**Определение**

Условия платежа по контракту или обязательствам.

***Пример — «В течение 30 дней с даты выставления счета», «по получении счета», «по завершении услуги».***

A.3.1.10 Класс FinancialTransaction (в предметной области Acts)

**Свойства класса FinancialTransaction**

Атрибуты класса FinancialTransaction:

- amt:: MO,
- creditExchangeRateQuantity:: REAL,
- debitExchangeRateQuantity:: REAL.

Класс FinancialTransaction является специализацией класса Act.

**Определение класса FinancialTransaction**

Действие, представляющее движение денежных сумм между двумя счетами.

**Примечания к использованию**

Финансовые операции всегда осуществляются между двумя счетами (дебет и кредит), но могут быть случаи, когда один или оба счета определяются более общей моделью или наследуются от нее.

Если у экземпляра класса FinancialTransaction атрибут moodCode имеет значение «ORD» (order — заказ), то этот экземпляр трактуется как требование выполнения операции.

Если атрибут moodCode имеет значение «EVN» (event — событие), то этот экземпляр класса FinancialTransaction содержит информацию об уже выполненной операции.

***Примеры — Затраты на услугу; оплата услуги; оплата счета.***

**Атрибуты класса FinancialTransaction**

A.3.1.10.1 FinancialTransaction.amt:: MO (0..1)

**Определение**

Указывает денежную сумму, перемещаемую с одного счета (например, кредита счета) на другой счет (например, дебет счета).

**Примечания к использованию**

Если денежные единицы суммы отличаются от денежных единиц дебета или кредита счета, то должен быть указан используемый обменный курс.

**Примечания к конструированию**

Обычно запись в бухгалтерской книге идентифицирует как дебит, так и кредит счета, а также текстовое описание транзакции. Если существуют другие документы, указывающие, как идентифицируются дебит и кредит счета или как аннотируются транзакции, то они должны быть упомянуты в подразделе примечаний к использованию.

A.3.1.10.2 FinancialTransaction.creditExchangeRateQuantity:: REAL (0..1)

**Определение**

Десятичное число, указывающее обменный курс между валютой кредита счета и валютой операции.

***Пример — При покупке услуг, оцененных в мексиканских песо (MXP) и оплаченных в долларах США (USD) со счета, который ведется в канадских долларах (CAD), в качестве обменного курса валюты кредита должно быть передано десятичное число r, для которого «y (USD) · r = x (CAD)».***

A.3.1.10.3 FinancialTransaction.debitExchangeRateQuantity:: REAL (0..1)

**Определение**

Десятичное число, указывающее обменный курс между валютой дебета счета и валютой операции.

***Пример — При покупке услуг, оцененных в мексиканских песо (MXP) и оплаченных в долларах США (USD) со счета, который ведется в канадских долларах (CAD), в качестве обменного курса валюты дебита должно быть передано десятичное число r, для которого «y (USD) · r = x (MXP)».***

A.3.1.11 Класс InvoiceElement (в предметной области Acts)

**Свойства класса InvoiceElement**

Атрибуты класса InvoiceElement:

- modifierCode:: SET<CE>,
- unitQuantity:: RTO<PQ,PQ>,
- unitPriceAmt:: RTO<MO,PQ>,
- netAmt:: MO,
- factorNumber:: REAL,
- pointsNumber:: REAL.

Класс InvoiceElement является специализацией класса Act.

**Определение класса InvoiceElement**

Действие, представляющее объявление и расшифровку «причитающейся суммы».

**Примечания к использованию**

Эта информация является частью «расшифровки» счета. Она часто объединяется с информацией о финансовой операции, представляющей сумму, подлежащую оплате, сумму, согласованную к оплате, или фактически оплаченную сумму. Чтобы разбить один элемент счета-фактуры, передаваемый в экземпляре класса InvoiceElement, на несколько составляющих элементов, можно использовать рекурсивные отношения. Если у экземпляра класса InvoiceElement атрибут moodCode имеет значение «DEF» (definition — определение), то этот экземпляр описывает «возможное» согласование строки счета-фактуры при его будущем рассмотрении. Если атрибут moodCode имеет значение «RQO» (request — требование), то этот экземпляр класса InvoiceElement содержит запрос на определение причитающейся суммы. Если атрибут moodCode имеет значение «EVN» (event — событие), то этот экземпляр класса InvoiceElement содержит информацию о сумме, которую должен оплатить конкретный получатель.

**Атрибуты класса InvoiceElement**

A.3.1.11.1 InvoiceElement.modifierCode:: SET<CE> (0..*)

Словарный домен: InvoiceElementModifier

**Определение**

Указывает модификатор кода, передаваемого в атрибуте code, для представления дополнительной информации об элементе счета-фактуры.

**Обоснование**

Этот модификатор не рассматривается как часть прекоординированной классификации с кодом, передаваемым в атрибуте code, поскольку система кодирования значений атрибута modifierCode не обязательно специально является детализацией системы кодирования атрибута code. Это не соответствует смыслу имени modifier (модификатор), поскольку обычно словарный домен модификатора должен быть определен как часть базового системы кодирования или должен быть разработан специально для нее.

**Примечания к конструированию**

К обоснованию: причина, по которой значения атрибута code не ограничены, состоит в том, чтобы он мог использовать словарные домены без ограничений. Где может быть ограничен атрибут modifierCode аналогичным образом? Если это ограничение относится к модификатору типа данных CD, оно должно подразумеваться в кодированном атрибуте.

***Пример — Отдаленная территория, внеурочное обслуживание.***

A.3.1.11.2 InvoiceElement.unitQuantity:: RTO<PQ,PQ> (0..1)

**Определение**

Описание количества товара или услуги, которое включено в счет или подлежит включению в счет.

**Примечания к использованию**

Указание исчисляемых единиц может быть выполнено с помощью следующих методов:

1) определить исчисляемую единицу в атрибуте InvoiceElement.code. В этом случае конкретное значение атрибута InvoiceElement.code будет содержать указание, что предмет, описанный в данном экземпляре класса InvoiceElement, является упаковкой, содержащей 20 элементов. Если значение InvoiceElement.code соответствует упаковке, содержащей 20 элементов, а значение атрибута количества InvoiceElement.unitQuantity = 2 (штуки), то данный экземпляр класса InvoiceElement описывает две упаковки по 20 элементов в каждой, то есть всего 40 элементов;

2) если требуется указать больше деталей (например, описать состав, упаковку, производителя товара), то эта информация должна быть описана как экземпляр класса Entity, связанный с данным экземпляром класса InvoiceElement с помощью экземпляров классов Role и Participation, при этом атрибут Participation.typeCode должен иметь значение «PRD» (product — товар).

Каждый экземпляр класса InvoiceElement, описывающий элемент, включенный в счет-фактуру или подлежащий включению в счет, идентифицируется кодом товара или услуги, передаваемым в атрибуте InvoiceElement.code. В некоторых случаях этот код берется из прекоординированной классификации и идентифицирует контейнер (например, универсальный код продукта (УКП), присвоенный контейнеру, содержащему 1000 таблеток, и другой УКП-код для контейнера, содержащего 100 таких же таблеток). УКП-код используется при выставлении счетов, но при его применении возникает необходимость указать в форме дроби, что только часть контейнера (например, флакона) подлежит оплате или включению в счет. Например, пусть подлежат оплате 15 таблеток из контейнера, содержащего 1000 таблеток. В этом случае числитель дроби может быть указан как «15 {таблетка}» или просто «15», а знаменатель как «1000 {флакон}» или просто «1000» (см. обсуждение, следующее за обоснованием использования описательного текста для исчисляемых величин). Если товар, информация о котором передается в экземпляре класса InvoiceElement, не является контейнером, то знаменатель дроби не указывается.

**Примечания к конструированию**

Следует утвердить реорганизацию. Ссылка на спецификацию типов данных в документе Data Types Part II Unabridged Specification, Appendix A: Unified Code for Units of Measure тупиковая. Спецификацию UCUM (Unified Code for Units of Measure — унифицированные коды единиц измерения) можно найти в документе по типам данных, поэтому обновленная ссылка могла бы помочь, однако она не помогает с ограничением представления «безразмерностных» единиц, не запрещенных этой спецификацией. Замечания к отображению: страница института Regenstrief с описанием UCUM предоставлена в одном фрейме с голосуемым содержанием HL7 Версии 3. Даже если на этот счет существует соглашение о взаимопонимании, этот контекст вводит читателей в заблуждение. Его следует представлять в отдельном окне. Рекомендация: следует добавить ссылку на спецификацию UCUM в разделы введения, содержащиеся в сопутствующих документах, и убрать ее из описаний атрибутов.

***Примеры — 4 часа, 4 мг, 4 коробки, 15 штук из контейнера, содержащего 1000 штук, и т. д.***

**Формальное ограничение**

Единицы товара или услуги ДОЛЖНЫ быть ограничены такими измеряемыми единицами, как литры, миллиграммы или часы. Не измеряемые, но исчисляемые единицы, например короба, пакеты, посещения, таблетки и контейнеры, НЕ ДОЛЖНЫ указываться в компоненте единиц измерения типа данных PQ иначе как в аннотации.

A.3.1.11.3 InvoiceElement.unitPriceAmt:: RTO<MO,PQ> (0..1)

**Определение**

Стоимость единицы товара или услуги.

***Примеры — $0.20/мг; $250/день; $50.***

**Формальное ограничение**

При указании отношения числитель должен иметь тип данных MO, а знаменатель — тип данных PQ. Детали указания отношения см. в описании атрибута unitQuantity.

A.3.1.11.4 InvoiceElement.netAmt:: MO (0..1)

**Определение**

Указывает полную стоимость элемента счета-фактуры, включая суммы его компонентов.

**Примечания к использованию**

Для листовых элементов счета-фактуры эта сумма вычисляется по формуле «unitQuantity · unitPriceAmt[ · factorNumber][ · pointsNumber]». Для группировок элементов счета-фактуры эта сумма вычисляется как сумма значений атрибутов netAmt всех элементов группы.

A.3.1.11.5 InvoiceElement.factorNumber:: REAL (0..1)

**Определение**

Указывает коэффициент, используемый при определении полной стоимости оказанных услуг и/или полученных товаров.

**Примечания к использованию**

Простейшая формула для вычисления полной суммы такова: «unitQuantity · unitPriceAmount = netAmt».

С помощью коэффициента можно учесть скидки или доплаты, применяемые к полной сумме. Например, с учетом коэффициента формула для вычисления полной суммы примет следующий вид: «unitQuantity · unitPrice (цена единицы или балла) · factorNumber = netAmt». Это понятие часто используется в Европе, чтобы устанавливать разные цены на услуги для обязательного и добровольного медицинского страхования.

См. аналогичное примечание к описанию атрибута InvoiceElement.pointsNumber, когда стоимость вычисляется с помощью баллов. Семантически баллы и коэффициент являются множителями цены и различаются только сценариями использования. Они могут применяться одновременно.

**Примечания к конструированию**

Следует утвердить примечания к использованию.

***Пример — 10 (число услуг в качестве единиц) · $3.00 (стоимость единицы) · 1.5 (коэффициент) = $45.00 (сумма).***

A.3.1.11.6 InvoiceElement.pointsNumber:: REAL (0..1)

**Определение**

Этот атрибут используется в ситуации, когда количество товара или услуги выражается в «баллах», позволяющих задать весовой коэффициент к стоимости товара или услуги (основанный на трудоемкости, стоимости и/или интенсивности ресурса).

**Примечание к использованию**

Баллы (points) используются в ситуации, когда услугам присваиваются относительные цены или единицы трудоемкости, а баллу назначается фиксированная цена. Коррекция всех цен, назначенных организацией, может осуществляться с помощью увеличения или снижения цены балла для отражения инфляции, накладных расходов и т. д. При одновременном применении коэффициента и баллов формула вычисления полной суммы примет следующий вид: «unitQuantity · unitPriceAmt · pointsNumber · factorNumber = netAmt».

См. соответствующее примечание к описанию атрибута factorNumber.

**Обоснование**

Понятие баллов может использоваться для расценки услуг и/или товаров, при которой количество услуг или товара измеряется в баллах, а одному баллу назначается определенная стоимость, например, в долларах.

***Пример — Стоимость процедуры, определяемая трудоемкостью, определяется выражением «5 (число единиц трудоемкости) · 3 (число баллов, присвоенных одной единице трудоемкости) · $20.00 (стоимость балла) = $300.00 (сумма).***

A.3.1.12 Класс ManagedParticipation (в предметной области Acts)

**Свойства класса ManagedParticipation**

Атрибуты класса ManagedParticipation:

- id:: SET<II>,
- statusCode:: CS>.

Класс ManagedParticipation является специализацией класса Participation.

Переходы состояний класса ManagedParticipation описаны в подразделе A.3.1.12.3.

**Определение класса ManagedParticipation**

Участие, которое может меняться с течением времени, в связи с чем его состоянием и идентичностью надо управлять.

**Обоснование**

Класс ManagedParticipation определен как подкласс класса Participation, чтобы явно указать, что не все участия не имеют состояния. В общем случае, если о подзадаче, реализуемой с помощью участия, надо иметь больше информации, и этой подзадачей надо управлять, то вместо применения парадигмы участия НАДО моделировать эту подзадачу как компонент основного действия, описываемого классом Act.

Однако в некоторых случаях представление о том, в чем именно состоят эти подзадачи и что именно выполняют участники, является не очень определенным и поэтому их моделирование в форме компонентов оказывается неоднозначным или затруднительным.

Для таких случаев предназначен класс ManagedParticipation, который расширяет базовый класс Participation двумя атрибутами: идентификатором id и кодом состояния statusCode. Классы ManagedParticipation

должны применяться с чрезвычайными предосторожностями, чтобы избежать путаницы с действиями, моделируемыми в виде классов Act, и не создавать для участий инфраструктуру управления, подобную той, что уже существует для действий.

**Примечания к конструированию**

Для этого класса в модели RIM опубликована машина перехода состояний, предусматривающая перечисляемые значения атрибута statusCode. Во введении должна быть проведена граница между кодированными атрибутами, специфичными для предметной области или сферы действия (повторение перечислимых значений таких атрибутов следует по возможности избегать), и структурными кодами, характеризующими машину перехода состояний и соответствующим образом перечисленными в модели.

***Пример — Лечащий врач госпитализированного пациента может уйти в отпуск, поэтому важно знать, когда его участие возобновится.***

**Атрибуты класса ManagedParticipation**

A.3.1.12.1 ManagedParticipation.id:: SET<II> (0..*)

**Определение**

Уникальный идентификатор, с помощью которого можно идентифицировать конкретный экземпляр класса ManagedParticipation среди всех экземпляров этого класса, имеющих ассоциации с тем же самым экземпляром класса Act и с тем же самым экземпляром класса Role.

A.3.1.12.2 ManagedParticipation.statusCode:: CS (0..1)

Словарный домен: ManagedParticipationStatus

**Определение**

Код, указывающий состояние экземпляра класса ManagedParticipation.

**Примечания к использованию**

В исходной модели RIM этот атрибут был определен как повторяющийся, чтобы можно было отразить наличие вложенных подсостояний, указанных на диаграмме перехода состояний. На практике, однако, необходимость передачи нескольких значений состояния никогда не возникала. Поэтому комитетам рекомендуется ограничить кратность этого атрибута до 1 во всех конструкциях сообщений.

***Пример — Готовящееся, активное, завершенное или отмененное участие.***

A.3.1.12.3 Переходы состояний класса ManagedParticipation

Диаграмма перехода состояний класса ManagedParticipation приведена на рисунке А.7. Управляемое участие может иметь следующие состояния:

- active (активно) — подсостояние состояния normal: это состояние отражает тот факт, что участие продолжается;
- cancelled (отменено) — подсостояние состояния normal: участие было отменено до того, как стало активным;
- completed (завершено) — подсостояние состояния normal: участие успешно завершено;
- normal (нормальное) — «типичное» состояние. Исключает состояние nullified, которое указывает, что экземпляр участия был создан по ошибке;
- nullified (аннулировано) — это состояние является терминальным состоянием экземпляра участия, созданного по ошибке;
- pending (готовящееся) — подсостояние состояния normal: это состояние отражает тот факт, что участие еще не стало активным.

Между состояниями действия возможны следующие переходы:

- revise (пересмотреть) — из состояния active в состояние active;
- complete (завершить) — из состояния active в состояние completed;
- reactivate (активизировать заново) — из состояния completed в состояние active;
- revise (пересмотреть) — из состояния completed в состояние completed;
- nullify (аннулировать) — из состояния normal в состояние nullified;
- create (создать) — из начального (пустого) состояния в состояние active;
- create (создать) — из начального (пустого) состояния в состояние completed;
- create (создать) — из начального (пустого) состояния в состояние pending;
- activate (активизировать) — из состояния pending в состояние active;
- cancel (отменить) — из состояния pending в состояние cancelled;
- revise (пересмотреть) — из состояния pending в состояние pending.

A.3.1.13 Класс Observation (в предметной области Acts)

**Свойства класса Observation**

Атрибуты класса Observation:

- value :: ANY,
- valueNegationInd :: BL,
- interpretationCode :: SET<CE>,

- methodCode :: SET<CE>,
- targetSiteCode :: SET<CD>.

Класс Observation является специализацией класса Act.

Класс Observation является обобщением следующих классов:

- DiagnosticImage;
- PublicHealthCase.

**Определение класса Observation**

Действие исследования, то есть получение новой информации о субъекте.

**Примечания к использованию**

Основное отличие класса Observation от других специализаций класса Act состоит в том, что класс Observation имеет атрибут value (значение). Для определения семантики исследования должно рассматриваться сочетание атрибутов Observation.code и Observation.value.

Структура многих результатов исследований может быть представлена в форме пар «имя — значение», при этом атрибут Observation.code (унаследованный от класса Act) — это имя свойства, а атрибут Observation.value — значение свойства. Такая конструкция часто называется «переменной» (то есть именованное свойство, которому может быть присвоено значение). Таким образом, класс Observation всегда используется для передачи общих пар «имя — значение», то есть переменных, даже если вычисление переменной и не является результатом сложного метода исследования. Оно может быть просто ответом на вопрос, или утверждением, или присваиванием значения параметру.

Как и все другие специализации класса Act, класс Observation используется для описания того, что сделано, и в случае класса Observation такое описание указывает, что было в действительности выявлено («результаты» или «ответы»), и эти «результаты» или «ответы» являются частью исследования и не расщепляются на объекты других типов.

Метод действия передается в атрибуте classCode класса Observation или его специализаций на самом верхнем уровне детализации, в атрибуте code на среднем уровне детализации и в атрибуте methodCode на самом высоком уровне детализации. Информация о методе в целом или частично может появляться в атрибуте value, если его значение представлено кодируемыми типами данных. Релевантные аспекты методик исследования могут также повторяться в атрибуте value, если результаты исследования сами по себе подразумевают методику.

Исследование может состоять из нескольких других исследований (компонентов), у каждого из которых свои значения атрибутов Observation.code и Observation.value. В этом случае составное исследование может не иметь собственного атрибута Observation.value. Например, дифференцированный подсчет белых клеток крови включает в себя отдельные результаты подсчета гранулоцитов, лейкоцитов и других нормальных или аномальных клеток крови (к примеру, незрелых клеток). Поэтому экземпляр класса Observation, описывающий составное исследование (подсчет белых клеток крови), может не иметь собственного значения (хотя его и можно получить, просуммировав все счетчики-компоненты). Таким образом, всякое действие, которое по своей природе является действием получения информации о субъекте и сообщения этой информации, представляется в форме экземпляра класса Observation независимо от того, имеет ли оно собственное простое значение или состоит из нескольких других исследований.

Хотя исследования являются профессиональными действиями (см. описание класса Act) и как таковые являются намеренными, это не требует, чтобы все возможные исходы исследований планировались заранее. Например, результаты дифференцированного подсчета белых клеток крови редко показывают наличие незрелых клеток, но, если таковые обнаружены, они включаются в результаты исследования вне зависимости от того, определены ли незрелые клетки в структуре нормального подсчета.

Клинические документы обычно имеют разделы измеренных значений и их клинической интерпретации. Каждый из этих разделов представляет собой отдельный вид исследования. Кроме того, клинические документы обычно содержат раздел заключения, который также представляет собой вид исследования. Таким образом, поставленный диагноз тоже относится к категории исследования.

**Примечания к конструированию**

В подразделе примечаний к использованию необходимо упомянуть исследования, которые не следуют семантической парадигме «имя — значение», например, те, что идентифицируют патологии.

***Примеры***

***1 Регистрация семейного анамнеза.***

***2 Лабораторный анализ и его результаты.***

***3 Физикальное исследование и его результаты.***

***4 Температура устройства.***

***5 Содержание свинца в почве.***

***6 Клиническое утверждение, например, перелом левого бедра.***

**Атрибуты класса Observation**

A.3.1.13.1 Observation.value:: ANY (0..*)

**Определение**

Информация, которая создана или определена действием исследования.

**Примечания к использованию**

Тип данных, который может принимать значение атрибута Observation.value, зависит от вида исследования и обычно указывается в описании исследования или определяется с помощью простого правила по виду исследования, передаваемому в атрибуте Observation.code.

Ниже приведены общие рекомендации по выбору подходящего типа данных:

1) в количественных результатах в основном используется тип данных физической величины (PQ). Этот тип данных, по существу, представляет собой вещественное число с единицей измерения. Это общие предпочтения для всех числовых значений. Некоторые исключения описаны далее.

Числовые значения НЕ ДОЛЖНЫ передаваться в форме простой строки символов (тип данных ST);

2) для титра (например, 1:64) и очень немногих других отношений используется тип данных RTO. У титров числителем и знаменателем отношения являются целые числа (например, 1:128). В других отношениях могут использоваться разные количественные типы данных, например, «цена», определенная как физическая величина с типом данных MO.

Иногда по местным соглашениям для титров передаются только знаменатели (например, 32 вместо 1/32). Такие соглашения могут вносить путаницу, и в сообщениях стандарта HL7 вместо них ДОЛЖНЫ использоваться правильные отношения;

3) для передачи значений индексов (чисел без единицы измерения) используется вещественный тип данных (REAL). Если величина не имеет подходящей единицы измерения, то ее можно передать как вещественное число. В качестве альтернативы можно использовать тип данных PQ с безразмерностной единицей (например, 1 или %). Целое число можно передавать только в том случае, если согласно определению исследования результатами могут являться только целые числа, что является редким случаем, например, если результат исследования имеет перечислимые значения (см. ниже);

4) диапазоны (например, < 3; 12–20) должны быть представлены в форме диапазонов физических количеств (тип данных IVL<PQ>) или в форме интервалов значений других количественных типов данных.

Иногда такие интервалы используются, чтобы указать неопределенность измеренного значения. Однако на этот случай имеются специальные расширения типов данных;

5) для передачи перечислимых значений (например +, ++, +++; или I, IIa, IIb, III, IV) используется тип данных CO (coded ordinal — кодированное перечислимое значение);

6) номинальные результаты («таксоны», например, тип организма) передаются с помощью любого из кодируемых типов данных (CD, CE), которые указывают по меньшей мере код, систему кодирования и необязательный исходный текст, преобразование в другую систему кодирования, а иногда еще и квалификаторы;

7) для представления мультимедийных данных используется тип данных ED (Encapsulated Data — инкапсулированные данные). С его помощью можно передать изображение (например, рентгенограмму грудной клетки) или видеозапись (например, результат коронарной ангиографии или эхокардиограмму) в виде вложенных двоичных данных или ссылки на внешние адреса, откуда они могут быть загружены по запросу;

8) записи биосигналов можно передавать, используя шаблоны коррелируемых последовательностей результатов (Correlated Observation Sequences), обеспечивающих передачу всей необходимой информации в структуре, предложенной в стандарте HL7. Эту информацию можно также передать, вложив ее в значение типа ED. Это позволяет использовать другие форматы цифрового кодирования биосигналов, кроме предложенных в стандарте HL7, а также передавать ссылки на внешние адреса, откуда оцифрованные записи биосигналов могут быть загружены по требованию;

9) иногда для передачи формализованных выражений, для которых не подходит ни один из указанных выше типов данных, может использовать строковый тип данных. Однако строковый тип данных НЕ ДОЛЖЕН использоваться, если значение может быть представлено одним из существующих типов данных;

10) временные метки не должны передаваться как значения результатов исследования, если для них можно найти более подходящие места, например, атрибут Act.effectiveTime некоторого действия (к примеру, «время получения биоматериала лабораторией» можно передать в атрибуте effectiveTime действия, описывающего транспорт биоматериала в лабораторию, а не как результат исследования);

11) множества значений любого типа данных, перечисляемые данные, а также интервалы часто используются в экземплярах класса Observation, описывающих критерии (то есть в экземплярах, у которых атрибут moodCode имеет значение «EVN.CRT»), чтобы указать «референтные пределы» или «пороговые значения» (для тревожных сигналов) и т. д.;

12) для последовательностей результатов (повторяемые измерения того же свойства в течение относительно короткого времени) используется тип данных LIST (list — список). Дополнительные детали см. в спецификации коррелируемых последовательностей результатов (Correlated Observation Sequences);

13) степень неопределенности значений можно указать, используя расширения типа данных вероятности и распределения вероятности (UVP, PPD). Для передачи статистики абсолютных частот категорий вполне пригоден тип мультимножества BAG.

A.3.1.13.2 Observation.valueNegationInd:: BL (0..1)

**Определение**

Указывает, что исследование состоялось, но результат, переданный в атрибуте value, не имеет места.

**Примечания к использованию**

Этот атрибут должен использоваться только в том случае, если терминология, использованная для представления значения атрибута value, не позволяет выражать отрицательные результаты (например, МКБ-9).

A.3.1.13.3 Observation.interpretationCode:: SET<CE> (0..*)

Словарный домен: ObservationInterpretation

**Определение**

Качественная интерпретация исследования.

**Примечания к использованию**

Эти коды интерпретации иногда называют «флагами аномалий», однако оценка нормы представляет собой только один пример грубой интерпретации и часто неприменима. Например, оценка чувствительности микроорганизмов не имеет никакого отношения к «норме», и при любом исследовании патологического состояния нет смысла говорить о норме, так как патологические состояния никогда не считаются «нормальными».

***Примеры — Норма, патология, ниже нормы, возрастает, устойчивый.***

A.3.1.13.4 Observation.methodCode:: SET<CE> (0..*)

Словарный домен: ObservationMethod

**Определение**

Информация о методике или способе исследования.

**Примечания к использованию**

При всех исследованиях метод их выполнения частично определяется по виду исследования (атрибут Observation.code), и эту косвенную информацию о методе не надо явным образом указывать в атрибуте Observation.methodCode. Например, если в атрибуте Observation.code передается код из классификации LOINC, то метод может быть уже известен с определенной степенью точности: в классификации LOINC многим видам исследований присвоены разные коды, если методики исследования различаются и это может практически повлиять на интерпретацию результатов исследования. Например, в этой классификации проводится различие между исследованием чувствительности к антибиотикам методом «минимальной ингибирующей концентрации» (МИК) и «методом диффузии в агаре» (Кирби — Бауэр), так что этим видам исследований специально присвоены разные коды. Следовательно, значение атрибута methodCode может лишь служить дополнительным квалификатором, указывающим то, что не выводится непосредственно из значения атрибута Act.code.

Кроме того, некоторые вариации методов могут быть связаны с конкретным используемым устройством. Но значение атрибута methodCode не должно использоваться для указания конкретного устройства или использованного диагностикума. Такую информацию надо связывать с данным экземпляром класса Observation, используя экземпляр класса Participation, у которого атрибут typeCode имеет значение «DEV» (device — устройство).

***Примеры — Способы измерения давления крови (артериальная пункция, сфигмоманометр (Riva-Rocci), сидя, лежа на спине и т. д.).***

A.3.1.13.5 Observation.targetSiteCode:: SET<CD> (0..*)

Словарный домен: ActSite

**Определение**

Код, указывающий анатомическую локализацию или систему организма, являющуюся предметом исследования.

**Примечания к использованию**

В большинстве случаев исследуемая анатомическая локализация вытекает из определения исследования, значения атрибута Observation.code или значения атрибута Observation.value. Например, «шумы сердца» заведомо относятся к сердцу. Атрибут targetSiteCode используется только в тех случаях, когда анатомическая локализация должна быть детализирована, например, чтобы различать правую и левую сторону и т. д. Если предметом исследования не является человек или животное, то этот атрибут используется аналогичным образом для указания структурного ориентира предмета исследования. Например, если предмет — озеро, то исследуемой локализацией может быть приток или исток, и т. д. Если предметом является лимфоузел, то в качестве локализации можно указать «ворота», «периферию» или иное место узла.

***Примеры — Сердце, ворота лимфатического узла, приток озера.***

**Формальное ограничение**

Если значение атрибута targetSiteCode присутствует, то оно НЕ ДОЛЖНО противоречить той информации об анатомической локализации или системе организма, которая вытекает из определения исследования или из значения атрибута Observation.code.

A.3.1.14 Класс Participation (в предметной области Acts)

**Свойства класса Participation**

Атрибуты класса Participation:

- typeCode :: CS.

- functionCode :: CD,
- contextControlCode :: CS,
- sequenceNumber :: INT,
- negationInd :: BL,
- noteText :: ED,
- time :: IVL<TS>,
- modeCode :: CE,
- awarenessCode :: CE,
- signatureCode :: CE,
- signatureText :: ED,
- performInd :: BL,
- substitutionConditionCode :: CE,
- subsetCode :: CS.

Ассоциации класса Participation:

- act::(1..1)Act::participation::(0..*) (ассоциация с классом Act, роль participation — участие),
- role::(1..1)Role::participation::(0..*) (ассоциация с классом Role, роль participation — участие).

Класс Participation является специализацией класса InfrastructureRoot.

Класс Participation является обобщением следующего класса:

- ManagedParticipation (управляемое участие).

**Определение класса Participation**

Ассоциация между классом действия Act и классом роли Role. Сущность Entity, принимающим участие в действии, описанном классом Act, выполняет в этом участии роль, описанную классом Role.

**Примечания к использованию**

Каждая пара «сущность — роль» связана с экземпляром класса Act с помощью одного экземпляра класса Participation. Тип вовлечения этой пары в действие, описываемое экземпляром класса Act, задается с помощью атрибута Participation.typeCode.

Класс роли Role, ассоциированный с классом сущности Entity, является посредником между классами Entity и Participation. Экземпляры класса Participation описывают выполняемую функцию, в то время как экземпляры класса Role описывают компетенцию. Экземпляры класса Participation описывают фактическое участие сущности в определенном действии, и следовательно, вид участия зависит от специфики этого действия и очень редко — от этой сущности. Напротив, роль описывает компетенцию сущности (например, какое принципиальное участие она может принимать в действиях) независимо от конкретного действия.

Например, профессиональные полномочия лица (описанные экземпляром класса Role) могут значительно отличаться от его фактических действий (описанных экземпляром класса Participation). Типичным примером может служить выполнение интерном или ординатором анестезии либо хирургической операции под присмотром (более или менее тщательным) ведущего хирурга: роль интерна не задает характер его участия.

Один и тот же вид участия в действии могут принимать несколько сущностей, что бывает при коллективном сотрудничестве или групповых действиях. Понятие кратных участий МОЖЕТ быть также представлено в виде подчиненных действий (компонентов действия): участие нескольких действующих лиц может быть представлено как составное действие, составленное из нескольких действий, каждое из которых выполняется только одним лицом, либо как одно действие, выполняемое этими участниками.

Например, в протокол хирургической операции могли быть внесены три лица:

a) лицо, давшее согласие на проведение операции;

b) ведущий хирург;

c) анестезиолог.

Эти лица выполняют разные задачи, которые можно представить в виде трех связанных действий:

a) предоставление согласия;

b) собственно операция;

c) анестезия, выполняемая параллельно с операцией.

Если используются три таких компонента, то тип действующего лица у согласившегося на операцию, хирурга и анестезиолога будет одним и тем же, а именно «исполнитель». Чем более детальные компоненты действий используются, тем меньше требуется градаций действующих лиц. И наоборот, чем меньше компонентов у действия, тем больше требуется градаций действующих лиц (и экземпляров класса Participation).

Как эмпирическое правило использование подзадач вместо нескольких действующих лиц полезно в тех случаях, когда каждая подзадача требует специального планирования или отдельной оплаты или если полная ответственность за выполнение подзадач различна. Однако в большинстве случаев ресурсы персонала планируются в форме бригад (а не отдельных сотрудников), методы расчетов по оплате лечения имеют тенденцию группировки мелких подзадач в одну строку счета, а полная ответственность часто возлагается на одного лечащего врача, старшую медсестру или заведующего отделением. В то время как с помощью класса ActRelationship можно обеспечить детальную декомпозицию действия, применение класса Participation позволяет объединять мелкие подзадачи в одно действие, выполняемое несколькими лицами.

***Примеры***

***1 Исполнители действий (хирурги, исследователи, врачи общей практики).***

***2 Субъекты действий, пациент, устройства.***

***3 Местонахождения.***

***4 Автор, поручитель, свидетель, информатор.***

***5 Адресат, получатель информации.***

**Атрибуты класса Participation**

A.3.1.14.1 Participation.typeCode:: CS (1..1) Mandatory

Словарный домен: ParticipationType

**Определение**

Код, указывающий вид участия (вовлечения) сущности, выполняющей определенную роль, связанную с этим участием, в действии, описанном в ассоциированном экземпляре класса Act.

A.3.1.14.2 Participation.functionCode:: CD (0..1)

Словарный домен: ParticipationFunction

**Определение**

Необязательный код, указывающий дополнительную информацию о функции данного участия в действии, если она однозначно не вытекает из значения атрибута Participation.typeCode.

**Ограничения использования**

Не будет написано ни одной спецификации стандарта HL7, технически зависящей от атрибута functionCode. Если возникнет потребность в дополнительных понятиях, то они должны быть определены для значений атрибута Participation.typeCode.

**Примечания к использованию**

Этот код может указывать множество функций по сравнению с теми, что можно выразить с помощью жестко контролируемого атрибута typeCode. Количество и виды применимых функций зависят от конкретного вида действия, например, для каждого типа операций может требоваться свое число ассистирующих хирургов и операционных сестер.

Поскольку функции участия характеризуют то, что именно люди выполняют в процессе действия, то в действительности они эквивалентны подзадачам, которые могут выполняться параллельно. Если нужны дополнительные сведения об этих подзадачах, кроме списка их участников, то надо использовать компоненты действия.

***Примеры — Ведущий хирург, второй хирург (или первый ассистент хирурга, стоящий напротив ведущего хирурга), второй ассистент (часто стоящий рядом с ведущим хирургом), потенциально третий ассистент, старшая операционная сестра, операционная сестра, медрегистратор, ведущий анестезиолог, анестезиолог, сестра-анестезистка, санитар, укладывающий пациента, сестра послеоперационного мониторинга, ассистенты, акушерки, студенты и т. д.***

**Формальное ограничение**

Если этот код указан, его значение НЕ ДОЛЖНО конфликтовать со значением атрибута Participation.typeCode.

A.3.1.14.3 Participation.contextControlCode:: CS (0..1)

Словарный домен: ContextControl

**Определение**

Код, указывающий, какой вклад вносит данный экземпляр класса Participation в контекст текущего экземпляра класса Act, и будет ли этот контекст распространяться на действия-потомки, чьи связи разрешают такое распространение (см. описание атрибута ActRelationship.contextConductionInd).

**Примечания к использованию**

Обоснование, обсуждение и примеры см. в описании атрибута ActRelationship.contextControlCode.

A.3.1.14.4 Participation.sequenceNumber:: INT (0..1)

**Определение**

Целое значение, указывающее относительный порядок данного участия среди других участий в том же самом действии.

***Примеры — Указание порядка участия выгодоприобретателей, необходимое для координации возмещений по счету на оплату страхового случая.***

A.3.1.14.5 Participation.negationInd:: BL (0..1)

**Определение**

Значение «true» этого атрибута указывает, что участие, описанное в экземпляре класса Participation, не имело, не имеет или не должно иметь места (в зависимости от значения атрибута moodCode).

**Примечания к использованию**

В случае конфликта экземпляр класса Participation, у которого атрибут negationInd имеет значение «true», обладает преимуществом по отношению к экземпляру класса Participation.

**Обоснование**

Этот атрибут имеет два основных приложения:

1) чтобы указать, что конкретная роль не имела или не должна иметь участие в действии;

2) чтобы удалить участника из контекста, распространяемого на другие действия.

***Примеры — Доктор Смит не участвовал; пациент Джонс не подписывал согласия.***

A.3.1.14.6 Participation.noteText:: ED (0..1)

**Определение**

Текстовое или мультимедийное представление комментария к данному участию.

**Примечания к использованию**

Этот комментарий относится только к непосредственному участнику.

A.3.1.14.7 Participation.time:: IVL<TS> (0..1)

**Определение**

Интервал времени, в течение которого сущность, связанная с действием с помощью конкретного экземпляра класса Participation, была вовлечена в это действие.

**Примечания к использованию**

Указание времени участия необходимо, если сущность принимала участие в действии не на протяжении всего времени выполнения действия. Время участия используется для указания времени, в течение которого выполнялись определенные общие подзадачи, не заслуживающие выделения в действии, но подразумеваемые типом участия.

***Примеры***

***1 Время ввода данных в систему-источник передается в атрибуте Participation.time экземпляра класса Participation, связывающем действие с ролью «ввод данных».***

***2 Концом времени участия автора считается время подписи данных, передаваемых в экземпляре класса Act.***

***3 Период времени, в течение которого др. Джонс нес ответственность за лечение пациента.***

A.3.1.14.8 Participation.modeCode:: CE (0..1)

Словарный домен: ParticipationMode

**Определение**

Код, указывающий модальность участия сущности в действии, выполняя определенную роль.

**Примечания к использованию**

Этот атрибут часто используется для участников «автор» («создатель»), чтобы определить, обеспечивалась ли информация, представленная действием, первоначально устно, письменно (рукописно) или в электронном виде.

***Примеры — Физическое присутствие, по телефону, письменная коммуникация.***

A.3.1.14.9 Participation.awarenessCode:: CE (0..1)

Словарный домен: TargetAwareness

**Определение**

Код, указывающий степень осведомленности сущности о действии, в которой она участвует в определенной роли.

**Примечания к использованию**

В процессе диагностики пациент, члены его семьи или другие участники могут быть не осведомлены о наличии у пациента смертельного заболевания. Поскольку указание этого атрибута нередко означает недостаточную степень осведомленности, то оно обычно относится к целевому экземпляру класса Participation (например, к пациенту). Если степень осведомленности, отказ, отсутствие сознания и т. д. — предмет медицинского рассмотрения (например, часть списка проблем), то нужно передавать эту информацию в экземплярах класса Observation, а не полагаться исключительно на этот простой атрибут класса Participation, который не может представить информацию, достаточную для обеспечения принятий медицинского решения.

***Примеры — Полностью осведомлен, неспособен воспринять, не информирован.***

A.3.1.14.10 Participation.signatureCode:: CE (0..1)

Словарный домен: ParticipationSignature

**Определение**

Код, указывающий, заверила ли сущность свое участие подписью, а также указывающий необходимость такой подписи.

**Примечания к использованию**

См. также описание атрибута Participation.signatureText.

***Примеры — Протокол хирургической операции (представленный в форме экземпляра класса Procedure) должен быть подписан ответственным оперировавшим хирургом и, возможно, другими участниками операции; участник намерен поставить подпись.***

A.3.1.14.11 Participation.signatureText:: ED (0..1)

**Определение**

Текстовое или мультимедийное представление подписи, подтверждающей определенный вид участия сущности в действии (указанный в атрибуте Participation.typeCode) и ее согласие принять на себя связанную с этим ответственность.

**Примечания к использованию**

Подпись может быть представлена многими разными способами либо по значению, либо по ссылке в соответствии с типом данных ED. Типичные случаи таковы:

1) подписи бумажных документов: значение, имеющее тип данных ED, может содержать ссылку на некоторый документ или файл, который может быть найден с помощью электронного интерфейса к архиву бумажных документов;

2) электронная подпись: с помощью типа данных ED можно представить фактически любую схему электронной подписи;

3) квалифицированная электронная подпись: в частности, с помощью типа данных ED можно представить квалифицированные электронные подписи, например, с помощью ссылки на блок данных подписи, сконструированный в соответствии со стандартом электронной подписи, например, XML-DSIG, PKCS#7, PGP и т. д.

***Примеры***

***1 Участник «автор» берет на себя ответственность за достоверность содержания экземпляра класса Act в меру его осведомленности.***

***2 Получатель информации подтверждает лишь факт ее получения.***

A.3.1.14.12 Participation.performInd:: BL (0..1)

**Определение**

Указывает, должен ли быть зарезервирован ресурс, необходимый для этого участия, до начала участия (то есть предоставление ресурса управляется расписанием).

**Примечания к использованию**

Этот признак используется для очень специфичных нужд в контексте планирования ресурсов. В большинстве описаний участия он не требуется. Чаще всего он применяется при описании участия, для которого требуется конкретное помещение или определенное оборудование, использование которого контролируется расписанием.

A.3.1.14.13 Participation.substitutionConditionCode:: CE (0..1)

Словарный домен: SubstitutionCondition

**Определение**

Указывает условия, при которых один участвующий предмет может быть заменен другим.

A.3.1.14.14 Participation.subsetCode:: CS (0..1)

Словарный домен: ParticipationSubset

**Определение**

Указывает, что данное участие представляет собой фильтрованное подмножество всех участий данного типа, ассоциированных с одним экземпляром класса Act.

**Примечания к использованию**

Этот атрибут используется, если необходимо ограничить участие первым элементом, последним элементом, следующим элементом или некоторым отфильтрованным подмножеством.

A.3.1.15 Класс PatientEncounter (в предметной области Acts)

**Свойства класса PatientEncounter**

Атрибуты класса PatientEncounter:

- admissionReferralSourceCode:: CE,
- lengthOfStayQuantity:: PQ,
- dischargeDispositionCode:: CE,
- acuityLevelCode :: CE,
- preAdmitTestInd:: BL,
- specialCourtesiesCode:: SET<CE>,
- specialArrangementCode:: SET<CE>.

Класс PatientEncounter является специализацией класса Act.

**Определение класса PatientEncounter**

Взаимодействие между пациентом и поставщиком (поставщиками) медицинской помощи с целью предоставления одной или нескольких услуг в сфере здравоохранения.

**Примечания к использованию**

К таким услугам относится в том числе оценка состояния здоровья.

***Примеры — Амбулаторное посещение нескольких отделений, медицинская помощь на дому (включая физиотерапию), госпитализация, посещение травмпункта, полевая медицинская помощь (например, при дорожном происшествии), посещение кабинета врачебного приема, производственная терапия, телефонный звонок.***

**Атрибуты класса PatientEncounter**

A.3.1.15.1 PatientEncounter.admissionReferralSourceCode:: CE (0..1)

Словарный домен: EncounterReferralSource

**Определение**

Тип места или организации, отвечавшей за лечение пациента непосредственно перед данным посещением.

A.3.1.15.2 PatientEncounter.lengthOfStayQuantity:: PQ (0..1)

**Определение**

Общее время, которое субъект проведет или провел в медицинской организации как часть посещения.

**Примечания к использованию**

Фактическое число дней пребывания не может быть получено вычитанием даты выписки из даты поступления из-за возможных отпусков (на выходные).

A.3.1.15.3 PatientEncounter.dischargeDispositionCode:: CE (0..1)

Словарный домен: EncounterDischargeDisposition

**Определение**

Код, указывающий место отправки пациента при выписке.

**Примечания к использованию**

Если посещение все еще «активно» (то есть у него еще нет даты завершения), то этот атрибут должен рассматриваться как ожидаемое место отправки. Если посещение уже «завершено», то в этом атрибуте передается фактическое место отправки.

***Примеры — Выписан домой, умер, выписался вопреки рекомендациям врача.***

A.3.1.15.4 PatientEncounter.acuityLevelCode :: CE (0..1)

Словарный домен: EncounterAcuity

A.3.1.15.5 PatientEncounter.preAdmitTestInd:: BL (0..1)

**Определение**

Признак необходимости предварительного выполнения лабораторных анализов перед данным посещением пациента.

A.3.1.15.6 PatientEncounter.specialCourtesiesCode:: SET<CE> (0..*)

Словарный домен: EncounterSpecialCourtesy

**Определение**

Код, указывающий признак особого обслуживания пациента в течение посещения.

***Примеры — Профессиональное обслуживание, обслуживание очень важного лица, обычное обслуживание.***

A.3.1.15.7 PatientEncounter.specialArrangementCode:: SET<CE> (0..*)

Словарный домен: SpecialArrangement

**Определение**

Код, указывающий тип специальных мер, которые должны быть предприняты в контексте посещения пациента.

**Примечания к использованию**

Если атрибут moodCode экземпляра класса PatientEncounter указывает будущее действие, то этот экземпляр может использоваться для указания специальных мер, которые надо предпринять перед ожидаемым поступлением пациента. Это не связано с событием размещения пациента.

***Примеры — Каталка, носилки, переводчик, дежурный врач, собака-поводырь.***

A.3.1.16 Класс Procedure (в предметной области Acts)

**Свойства класса Procedure**

Атрибуты класса Procedure:

- methodCode:: SET<CD>,
- approachSiteCode:: SET<CD>,
- targetSiteCode:: SET<CD>.

Класс Procedure является специализацией класса Act.

Специализации класса Procedure:

- SubstanceAdministration.

**Определение класса Procedure**

Непосредственным и основным результатом процедуры (постусловие) является измененное физическое состояние субъекта.

**Примечания к использованию**

Применительно к клинической медицине процедура — один из нескольких типов клинических действий, например, исследование, лекарственные назначения и коммуникативные воздействия (к примеру, обучение, консультация, психотерапия, представленные в форме экземпляров класса Act без специальных атрибутов). Понятие

процедуры не поглощает эти другие понятия и, в свою очередь, не поглощается ими. Примечательно, что оно не включает в себя все действия, целью которых является вмешательство или лечение. Будет ли физическое изменение полезным субъекту или предполагается полезным, не имеет значения, существенно лишь то, что целью действия является изменение физического состояния субъекта.

Выбор между способами представлениями реальных действий в форме экземпляров классов основан на том, применимы ли специфические свойства процедуры и будет ли физическое изменение необходимым постусловием деятельности или ее части. Например, получение рентгеновского изображения иногда называется «процедурой», но в модели RIM оно не отображается на экземпляр класса Procedure, поскольку целью рентгеновского снимка не является изменение физического состояния тела.

При многих видах клинической деятельности отдельные действия, которые по своей природе являются исследованиями и процедурами, объединяются в одно составное действие. Например, при выполнении процедур лечебной радиологии (к примеру, интраартериальный тромболиз) производятся как исследования, так и лечение, и большинство хирургических процедур включает в себя намеренные и документированные шаги исследований. Такие клинические воздействия лучше всего представлять в виде нескольких действий-компонентов, каждый из которых имеет соответствующий тип.

***Примеры — Изменения физического состояния могут включать в себя нарушение целостности некоторой поверхности тела (например, разрез при хирургической операции), выполнение консервативных процедур, например, вправление вывихнутого сустава, манипуляции хиропрактика, массаж, бальнеотерапию, иглоукалывание, шиацу и т. д. За пределами клинической медицины можно привести такие примеры процедур, как изменение окружающей среды (например, спрямление рек, осушение болот, возведение дамб), ремонт или модернизация машин и т. д.***

**Атрибуты класса Procedure**

A.3.1.16.1 Procedure.methodCode:: SET<CD> (0..*)

Словарный домен: ProcedureMethod

**Определение**

Указывает средства или методику выполнения процедуры.

**Обоснование**

У любой процедуры может быть несколько разных методов. Хотя они обеспечивают в основном те же самые результаты, для более тщательной интерпретации протокола процедуры может требоваться указание, какой именно метод был применен (например, открытая или лапароскопическая холецистэктомия). Понятия метода могут быть «прекоординированы» в определении действия. Поскольку существует много возможных методов, существенно зависящих от конкретного вида процедуры, конструирование словарного домена, который бы охватывал все эти методы, представляется затруднительным. Однако для конкретного типа процедуры такую систему кодирования сконструировать можно. Таким образом, пользователь, направляющий пациента на процедуру, может указать один из нескольких вариантов ее выполнения, указав код метода. Возможные варианты методов выполнения могут быть описаны в справочнике процедур. В записях такого справочника, передаваемых с помощью экземпляров класса Procedure, у которых атрибут moodCode имеет значение «DEF» (definition — определение), атрибут methodCode может содержать список методов выполнения процедуры, используемый для выбора нужного метода при оформлении направления на процедуру или для проверки полученного протокола процедуры.

При лекарственных назначениях метод применения нередко описывается с помощью атрибута routeCode. В этом случае атрибут methodCode нужен только при условии, что значение атрибута routeCode должно быть дополнительно уточнено. Например, если атрибут routeCode имеет значение «перорально», то никакая другая информация о методе не требуется. Если же атрибут routeCode имеет значение «внутривенно» или «внутримышечно», то в атрибуте methodCode может быть указан точный метод применения, например, «медленная инъекция ударной дозы» или «вертикальная инъекция».

При лекарственных назначениях путь применения (передаваемый в атрибуте routeCode), место применения (передаваемое в атрибуте approachSiteCode) и метод применения (передаваемый в атрибуте methodCode) тесно связаны между собой. Значения всех этих трех атрибутов (если присутствуют) должны быть хорошо скоординированы и согласованы. В некоторых случаях система кодирования, использованная для указания значения одного из атрибутов, может быть прекоординированной для одной или двух других.

A.3.1.16.2 Procedure.approachSiteCode:: SET<CD> (0..*)

Словарный домен: ActSite

**Определение**

Анатомическая локализация или система организма, через которую процедура достигает места своего применения.

**Примечания к использованию**

Если субъектом процедуры является не человек и не животное, то этот атрибут используется аналогичным образом, чтобы определить структурный ориентир на предмете действия.

Некоторые списки мест доступа могут быть «прекоординированы» с описанием процедуры, чтобы исключить неправильный выбор анатомической локализации. Одна и та же информационная структура может использоваться в обоих подходах: прекоординированном и посткоординированном.

При лекарственных назначениях путь применения (передаваемый в атрибуте routeCode), место применения (передаваемое в атрибуте approachSiteCode), метод применения (передаваемый в атрибуте methodCode) и изделие, используемое для применения, тесно связаны между собой. Значения всех этих четырех атрибутов (если присутствуют) должны быть хорошо скоординированы и согласованы. В некоторых случаях система кодирования, использованная для указания значения одного из атрибутов, может быть прекоординированной для одной или нескольких других.

***Примеры***

***1 При нефрэктомии доступ к почке может быть трансабдоминальным или главным образом ретроперитонеальным.***

***2 Целью катетеризации легочной артерии является легочная артерия, но при этом местом доступа обычно является внутренняя яремная вена или подключичная вена, в шее или, соответственно, в подключичной ямке.***

***3 При применении лекарственного средства в атрибуте approachSiteCode передается точная анатомическая локализация, в которое оно введено или к которому оно применено.***

***4 Для неинвазивных процедур, например иглоукалывания, место доступа — прокалываемое место кожи.***

A.3.1.16.3 Procedure.targetSiteCode:: SET<CD> (0..*)

Словарный домен: ActSite

**Определение**

Анатомическая локализация или система организма, являющаяся местом применения процедуры.

**Примечания к использованию**

Если субъектом процедуры является не человек и не животное, то этот атрибут используется аналогичным образом, чтобы определить структурный ориентир на предмете действия.

Списки мест применения могут быть «прекоординированы» с описанием процедуры, чтобы исключить неправильный выбор анатомической локализации. Одна и та же информационная структура может использоваться в обоих подходах: прекоординированном и посткоординированном.

***Примеры***

***1 Местом применения нефрэктомии является правая или левая почка.***

***2 Местом применения катетеризации легочной артерии является легочная артерия.***

***3 Для неинвазивных процедур, например иглоукалывания, местом применения является орган/система, на которые рассчитано воздействие (например, «печень»).***

A.3.1.17 Класс PublicHealthCase (в предметной области Acts)

**Свойства класса PublicHealthCase**

Атрибуты класса PublicHealthCase:

- detectionMethodCode:: CE,
- transmissionModeCode:: CE,
- diseaseImportedCode:: CE.

Класс PublicHealthCase является специализацией класса Observation.

**Определение класса PublicHealthCase**

Описание события, условия либо комплекса событий или условий, имеющих определенное значение для общественного здоровья, представляется как специализация класса Observation.

**Примечания к использованию**

Обычно в экземпляре класса PublicHealthCase передается информация об одном или нескольких случаях инфекционного заболевания или другого события, подлежащего регистрации. Описание события общественного здоровья может включать в себя сведения о состоянии здоровья одного лица или сведения о нескольких случаях того же самого заболевания или условия, которые рассматриваются как угроза общественному здоровью. Определение события общественного здоровья [передаваемое в экземпляре класса PublicHealthCase, у которого атрибут moodCode имеет значение «DEF» (definition — определение)] включает в себя описание клинических, лабораторных и эпидемиологических показателей, связанных с заболеванием или условием, влияющим на общественное здоровье. Есть определения событий для условий, которые подлежат регистрации, а также для тех, которые не подлежат. Есть также определения событий вспышек заболеваний. Определение события общественного здоровья используется здравоохранением для статистики событий и не должно использоваться в качестве клинических рекомендаций, предназначенных для лечения.

***Примеры — ВИЧ, синдром токсического шока, сальмонеллез (и ассоциированные признаки, используемые при определении угрозы).***

**Атрибуты класса PublicHealthCase**

A.3.1.17.1 PublicHealthCase.detectionMethodCode:: CE (0..1)

Словарный домен: CaseDetectionMethod

**Определение**

Код, указывающий метод получения санитарно-эпидемиологической службой информации о событии общественного здоровья.

***Примеры — Сообщение поставщика медицинской помощи, сообщение лаборатории, отчет о результате исследования угрозы или вспышки.***

A.3.1.17.2 PublicHealthCase.transmissionModeCode:: CE (0..1)

Словарный домен: CaseTransmissionMode

**Определение**

Код, указывающий механизм приобретения заболевания живым субъектом, вовлеченным в событие общественного здоровья.

***Примеры — Половой путь, воздушно-капельный, передача через переносчиков.***

A.3.1.17.3 PublicHealthCase.diseaseImportedCode:: CE (0..1)

Словарный домен: CaseDiseaseImported

**Определение**

Код, указывающий, была ли болезнь, вероятно, приобретена за пределами территории юрисдикции санитарно-эпидемиологической службы, и какова природа межтерриториальных отношений.

***Примеры — Не занесенная, занесенная из другой страны, недостаточно информации для определения.***

A.3.1.18 Класс SubstanceAdministration (в предметной области Acts)

**Свойства класса SubstanceAdministration**

Атрибуты класса SubstanceAdministration:

- routeCode:: CE,
- doseQuantity:: IVL<PQ>,
- rateQuantity:: IVL<PQ>,
- doseCheckQuantity:: SET<RTO>,
- maxDoseQuantity:: SET<RTO>,
- administrationUnitCode :: CE.

Класс SubstanceAdministration является специализацией класса Act.

**Определение класса SubstanceAdministration**

Вид процедуры, в процессе которой ее исполнитель вводит субстанцию в субъект или иным образом применяет субстанцию к субъекту.

**Примечания к использованию**

Применение субстанции отличается от воздействия наличием исполнителя.

При применении субстанции исполнитель физически взаимодействует с субъектом или иным образом «приближается» к субъекту в процессе применения.

Детальная информация о примененной субстанции представляется с помощью экземпляра класса Entity или одной из его специализаций.

Исполнителем применения субстанции может быть другая сущность, например, физическое лицо, устройство, растение (к примеру, ядовитый плющ), насекомое (например, комариный укус) либо сам субъект, как это имеет место при самостоятельном применении.

Для целей настоящего определения фотоны и другие виды излучения или световой энергии считаются субстанциями.

К субстанциям могут быть отнесены живые существа, например, живые вирусные вакцины и другие вещества, содержащие инфекционные агенты, например, слюна, продукты крови и т. д.

Примечание — Если инфекционный агент является предметом применения субстанции, то информация о нем моделируется с помощью класса LivingSubject.

Если атрибут moodCode имеет значение «INT» (intent — намерение), то экземпляр класса SubstanceAdministration описывает план применения данной субстанции. К таким планам относятся рецепты (но не только), которые могут также ассоциироваться с требованием поставки.

Если атрибут moodCode имеет значение «EVN» (event — событие), то экземпляр класса SubstanceAdministration описывает состоявшееся применение субстанции.

***Примеры***

***Класс SubstanceAdministration может использоваться для представления следующих применений:***

***1 Применение измеряемой величины внешней силы (например, лучевая терапия).***

***2 Применение измеряемой величины субстанции или силы как часть процесса исследования (например, применение глюкозы при проведении пробы на толерантность к глюкозе.***

***3 Химиотерапевтические протоколы (многократные применения субстанций).***

***4 Рецепт.***

***5 Регистрация вакцинации.***

***6 Питание через трубку.***

***7 Опыление сельскохозяйственных полей.***

***8 Смазывание машины.***

***9 Медикация стада на откормочной площадке с помощью пищевых добавок.***

**Атрибуты класса SubstanceAdministration**

A.3.1.18.1 SubstanceAdministration.routeCode:: CE (0..1)

Словарный домен: RouteOfAdministration

**Определение**

Физиологический путь или маршрут применения терапевтического материала в субъекте или на его поверхности.

**Ограничение использования**

Путь применения (передаваемый в атрибуте routeCode), место применения (передаваемое в атрибуте administrationSiteCode), метод применения (передаваемый в атрибуте methodCode) и изделие, используемое для применения, тесно связаны между собой. Значения всех этих четырех атрибутов (если присутствуют) должны быть хорошо скоординированы и согласованы. В некоторых случаях система кодирования, использованная для указания значения одного из атрибутов, может быть прекоординированной для одной или нескольких других.

**Примечания к использованию**

Если путь применения требует дальнейшей детализации, то могут использоваться как место применения (передаваемое в атрибуте administrationSiteCode), так и метод применения (передаваемый в атрибуте methodCode). К примеру, если атрибут routeCode имеет значение «внутривенно» или «внутримышечно», то может оказаться необходимым указать с помощью атрибута approachSiteCode точное место введения (например, правое предплечье или, соответственно, левая дельтовидная мышца) и передать в атрибуте methodCode точный метод применения (например, «медленная инъекция ударной дозы» или, соответственно, «вертикальная инъекция»). Когда лекарственное средство применяется к окружающей среде или к некоторому физическому месту, то код способа введения указывает это место в его «форме».

***Примеры — Перорально, ректально, внутривенно.***

A.3.1.18.2 SubstanceAdministration.doseQuantity:: IVL<PQ> (0..1)

**Определение**

Количество терапевтического агента или другой субстанции, введенное за один прием.

**Ограничение использования**

Не измеряемые, но исчисляемые единицы, например таблетки и капсулы, не должны указываться в компоненте единиц измерения типа данных PQ иначе как в аннотации, указанной в фигурных скобках ({xxx}).

**Примечания к использованию**

Доза может определяться или как физическая величина активного ингредиента (например, 200 мг), или как количество единиц назначения (например, таблетки, капсулы, «штуки»). Какой выбрать подход, зависит от исполнителя «потребляемого» участия (который идентифицирует вводимое лекарство). Если потребляемое лекарство имеет неисчисляемую форму дозировки (например, измеряется в миллиграммах или в литрах), тогда доза должна выражаться в этих единицах. Если потребляемое лекарство имеет исчисляемую форму дозировки (таблетки, капсулы, «штуки»), тогда доза должна быть выражена как безразмерностная величина (то есть никакая другая единица измерения не указывается).

**Примечания к конструированию**

Предшествующее ограничение гласило, что количество единиц назначения неприемлемо, пока не будет указано, как их использовать. Теперь это снято. Следует задаться вопросом, будет ли здесь уместным ограничение на тип данных PQ (а не атрибут)?

A.3.1.18.3 SubstanceAdministration.rateQuantity:: IVL<PQ> (0..1)

**Определение**

Указывает скорость введения субстанции в субъект. Выражается как физическая (экстенсивная) величина, введенная за единицу времени.

**Примечания к использованию**

Это понятие применимо к непрерывно делимым формам дозировки (например, жидкости, газы). Если значение указанного атрибута является интервалом, то скорость должна быть в этом интервале.

***Примеры — 100 мл/ч, 1 г/сут, 40 ммоль/ч.***

A.3.1.18.4 SubstanceAdministration.doseCheckQuantity:: SET<RTO> (0..*)

**Определение**

Отношение количества потребляемой субстанции к периоду времени, в течение которого она должна быть потреблена.

**Примечания к использованию**

Этот атрибут используется как критерий проверки значений, переданных в других атрибутах. Обычно он не используется; этот атрибут предназначен только для специальных целей. В некоторых странах, в частности в Японии, есть норма, требующая указывать общую суточную дозу в рецепте и сопутствующей документации. Цель этого требования в том, чтобы обеспечить и облегчить контроль общей назначенной дозы во избежание передозировки (или слишком малой дозировки).

***Примеры***

***1 При назначении эритромицина в дозировке 250 мг (1 таблетка 3 раза в день) общую суточную дозу можно рассчитать по формуле «doseCheckQuantity = doseQuantity (1) · Ingredient.quantity (250 мг) · effectiveTime (3 / сут) = 750 мг/сут».***

***1 При внутривенном введении ожидаемое количество субстанции в час можно рассчитать по формуле «doseCheckQuantity = doseQuantity (100 мл) · Ingredient.quantity (5 мг/л) / rateQuantity (1 час) = 0.5 мг/ч», что можно преобразовать в суточную дозу по формуле «doseCheckQuantity = 0.5 мг/ч · 24 ч/сут = 12 мг/сут».***

**Формальное ограничение**

Числитель должен быть в единицах, сопоставимых с единицами измерения значения doseQuantity, а знаменатель должен быть мерой времени.

A.3.1.18.5 SubstanceAdministration.maxDoseQuantity:: SET<RTO> (0..*)

**Определение**

Указывает максимальное общее количество субстанции, которое может быть применено к субъекту за период времени.

**Примечания к использованию**

Этот атрибут особенно полезен, если разрешенная дозировка задается в форме диапазона либо режим введения является переменным или по мере необходимости. С его помощью можно указать предел общего количества субстанции, которое может быть применено за период времени. Чтобы указать разные пределы для разных периодов времени, можно указать несколько значений атрибута maxDoseQuantity.

**Примечания к конструированию**

«Инвариантная» форма ограничения была удалена. Если она должна использоваться как способ формального описания ограничения, то следует упомянуть это во введении и применять во всех формальных ограничениях.

***Примеры — 500 мг/сут; 1200 мг/неделя.***

**Формальное ограничение**

Числитель должен выражаться в единицах, сопоставимых с единицами измерения значения doseQuantity, а знаменатель должен быть мерой времени.

A.3.1.18.6 SubstanceAdministration.administrationUnitCode :: CE (0..1)

Словарный домен: AdministrableDrugForm

**Определение**

Часть применяемой субстанции.

**Ограничение использования**

1) этот атрибут должен использоваться только в том случае, если вещество, указанное в качестве «исполнителя» в экземпляре класса Role, связанном с классом Psrticipation, описывающим потребляемое участие, представляет собой не готовую форму, применяемую целиком, а большее целое, упаковку и т. д.;

2) если вещество, представленное экземпляром класса SubstanceAdministration, является готовой формой, применяемой целиком, то атрибут administrationUnitCode должен иметь пустое значение («неприменим»);

3) если это вещество представляет собой аморфную субстанцию (жидкость, газ, порошок и т. д.), измеряемую объемом, массой и т. д., то атрибут administrationUnitCode также должен иметь пустое значение («неприменим»);

4) если экземпляр класса SubstanceAdministration означает контейнер, содержание которого измеряется объемом, массой и т. д., то атрибут administrationUnitCode должен иметь значение, соответствующее «отмеренной порции».

**Обоснование**

Если атрибут administrationUnitCode отсутствует, то в приведенном ниже примере значение атрибута doseQuantity = 1 означало бы, что весь флакон ингалятора должен быть опустошен за один акт применения. С помощью атрибута administrationUnitCode, содержащего код «дозы аэрозоля» (или «вдоха»), можно указать, что значение атрибута doseQuantity относится не ко всему количеству лекарственного средства, а только к его части.

***Пример — В системе ввода заказов может быть присвоен код только «Дозированному аэрозольному ингалятору с будесонидом», а доза должна измеряться «числом доз аэрозоля».***

A.3.1.19 Класс Supply (в предметной области Acts)

**Свойства класса Supply**

Атрибуты класса Supply:

- quantity:: PQ,
- expectedUseTime:: IVL<TS>.

Класс Supply является специализацией класса Act.

Класс Supply является обобщением класса Diet.

**Определение класса Supply**

Действие, включающее в себя передачу товара от одной сущности к другой.

**Примечания к использованию**

Информация о передаваемом товаре связывается с экземпляром класса Supply с помощью экземпляра класса Participation, у которого атрибут typeCode имеет значение «PRD» (product — товар). При этом важна точная идентификация товара (производитель, серийный номер и т. д.). Большая часть детальной информации о товаре должна передаваться в экземпляре класса Material. Если требуется отдельно описать планирование доставки, доставку и оплату товара, то с экземпляром класса Supply можно связать экземпляр класса Transportation. Для описания услуги отпуска лекарственного средства используется экземпляр класса Supply, связанный с экземпляром класса SubstanceAdministration. В этом случае экземпляр класса SubstanceAdministration описывает применение лекарства, а экземпляр класса Supply — отпуск.

***Примеры — Заказ простыней, отпуск лекарства, отпуск медицинских расходных материалов со склада.***

**Атрибуты класса Supply**

A.3.1.19.1 Supply.quantity:: PQ (0..1)

**Определение**

Количество товара, которое было предоставлено или должно быть предоставлено.

**Примечания к использованию**

Этот атрибут может использоваться как альтернатива атрибуту expectedUseTime или вместе с этим атрибутом. Если оба эти атрибута указаны, то значение, указанное в атрибуте quantity, представляет собой количество товара, которое предполагается израсходовать в течение времени, указанного в атрибуте expectedUseTime.

Единицы измерения должны быть ограничены такими измеряемыми единицами как литры, миллиграммы. Не измеряемые, но исчисляемые единицы, например таблетки и капсулы, не должны указываться в компоненте единиц измерения типа данных PQ иначе как в аннотации, указанной в фигурных скобках ({xxx}). Тип «исчисляемой» информации определен информацией сущности «продукта».

**Примечание к конструированию**

Удаленное ограничение на исчисляемые единицы.

A.3.1.19.2 Supply.expectedUseTime:: IVL<TS> (0..1)

**Определение**

Период времени, в течение которое предполагается использовать предоставляемый товар.

**Примечания к использованию**

В некоторых случаях этот атрибут МОЖЕТ использоваться вместо атрибута Supply.quantity для косвенного указания предоставляемого количества в форме продолжительности предоставления. Например, запас лекарства на 90 дней лечения (вычисление количества основано на дозировке лекарства), количество реактивного топлива на 10 часов полета и т. д. По возможности количество лучше передавать в атрибуте Supply.quantity, поскольку это точнее. Значение атрибута Supply.expectedUseTime всегда будет оценкой количества, подверженной влиянию внешних факторов.

A.3.1.20 Класс WorkingList (в предметной области Acts)

**Свойства класса WorkingList**

Атрибуты класса WorkingList:

- ownershipLevelCode:: CE.

Класс WorkingList является специализацией класса Act.

**Определение класса WorkingList**

Динамический список отдельных экземпляров класса Act, составленный для отражения потребностей отдельного сотрудника, бригады или организации по просмотру группы действий, выполняемых в клинических или административных целях.

**Примечания к использованию**

Группируемые действия связываются с экземпляром класса WorkingList с помощью экземпляров класса ActRelationship, у которых атрибут typeCode имеет значение «COMP» (component — компонент). Класс WorkingList имеет лишь один атрибут сверх тех, что наследуются от класса Act. В процессе гармонизации модели HL7 RIM использование этого атрибута в конструировании статических моделей, основанных на модели RIM, было запрещено с ноября 2005 года. Основанием послужила рекомендация технического комитета Patient Care. Как только этот атрибут будет отменен, то весь класс WorkingList будет удален из модели RIM. Тем не менее его использование с унаследованным атрибутом classCode, имеющим значение «LIST» (список), вполне приемлемо, пока в нем используются только атрибуты, унаследованные от класса Act.

*Примеры — Список жалоб, список целей лечения, список аллергий, листы назначений.*

**Атрибуты класса WorkingList**

A.3.1.20.1 WorkingList.ownershipLevelCode:: CE (0..1)

Словарный домен: ListOwnershipLevel

**A.3.2 Классы предметной области Entities**

A.3.2.1 Класс Container (в предметной области Entities)

**Свойства класса Container**

Атрибуты класса Container:

- capacityQuantity:: PQ,
- heightQuantity:: PQ,
- diameterQuantity:: PQ,
- capTypeCode:: CE,
- separatorTypeCode:: CE,
- barrierDeltaQuantity:: PQ,
- bottomDeltaQuantity:: PQ.

Класс Container является специализацией класса ManufacturedMaterial.

**Определение класса Container**

Сущность, содержащая другие сущности.

**Примечания к использованию**

Экземпляр класса Container связан с информацией о веществе, содержащемся в контейнере, с помощью экземпляра класса Role, у которого атрибут classCode имеет значение «CONT» (content — содержание).

**Обоснование**

Спецификация этого класса появилась в результате сотрудничества комитета HL7 и организации NCCLS. Многие из определений атрибутов позаимствованы из стандарта NCCLS или ссылаются на него. Для хранения аморфных субстанций (жидкости, газы) необходим контейнер. Однако его содержание всегда можно отличить от самого контейнера и относительно легко отделяется от него, в отличие от содержания (ингредиента) микстуры.

**Атрибуты класса Container**

A.3.2.1.1 Container.capacityQuantity:: PQ (0..1)

**Определение**

Функциональная емкость контейнера.

A.3.2.1.2 Container.heightQuantity:: PQ (0..1)

**Определение**

Высота контейнера.

A.3.2.1.3 Container.diameterQuantity:: PQ (0..1)

**Определение**

Внешний диаметр контейнера.

A.3.2.1.4 Container.capTypeCode:: CE (0..1)

Словарный домен: ContainerCap

**Определение**

Тип крышки контейнера, с которой могут осуществляться такие автоматизированные манипуляции, как открытие, прокалывание и т. д.

A.3.2.1.5 Container.separatorTypeCode:: CE (0..1)

Словарный домен: ContainerSeparator

**Определение**

Материал, добавляемый в контейнер для обеспечения и облегчения физического разделения компонентов образца различной плотности.

**Обоснование**

Состав или тип разделяющего вещества может оказывать воздействие на выполняемый анализ. Знание разделяющего материала помогает интерпретировать результаты.

***Пример — В пробирки для сбора крови добавляется гель, который при последующем центрифугировании создает физический барьер между клетками крови и сывороткой или плазмой.***

A.3.2.1.6 Container.barrierDeltaQuantity:: PQ (0..1)

**Определение**

Расстояние от контрольной точки до разделяющего материала (барьера) внутри контейнера.

**Примечания к использованию**

Это расстояние может передаваться автоматизированной лабораторной системой измерительному инструменту и/или устройству обработки биоматериала, чтобы пробоотборник, вставляемый в образец, не касался разделителя. См. определение контрольной точки (Point of Reference) в стандарте NCCLS AUTO5.

A.3.2.1.7 Container.bottomDeltaQuantity:: PQ (0..1)

**Определение**

Расстояние от контрольной точки до внешнего дна контейнера.

**Примечания к использованию**

См. определение контрольной точки (Point of Reference) в стандарте NCCLS AUTO5.

A.3.2.2 Класс Device (в предметной области Entities)

**Свойства класса Device**

Атрибуты класса Device:

- manufacturerModelName:: SC,
- softwareName:: SC,
- localRemoteControlStateCode:: CE,
- alertLevelCode:: CE,
- lastCalibrationTime:: TS.

Класс Device является специализацией класса ManufacturedMaterial.

**Определение класса Device**

Класс Device является специализацией класса ManufacturedMaterial, предназначенной для описания сущности, которая используется в деятельности, не претерпевая существенных изменений.

**Примечания к использованию**

Понятие устройства включает в себя как долговечное медицинское изделие (многократно используемое), так и одноразовое медицинское изделие. Тип устройства указывается в атрибуте code, унаследованном от класса Entity.

**Атрибуты класса Device**

A.3.2.2.1 Device.manufacturerModelName:: SC (0..1)

Словарный домен: ManufacturerModelName

**Определение**

Человекочитаемое наименование модели устройства, присвоенное производителем.

***Пример — Масс-спектрометр на основе индуктивно связанной плазмы Perkin Elmer 400.***

A.3.2.2.2 Device.softwareName:: SC (0..1)

Словарный домен: SoftwareName

**Определение**

Наименование, версия и выпуск программного обеспечения, управляющего устройством, присвоенные производителем или разработчиком программного обеспечения.

***Пример — Agilent Technologies Chemstation A.08.xx.***

A.3.2.2.3 Device.localRemoteControlStateCode:: CE (0..1)

Словарный домен: LocalRemoteControlState

**Определение**

Значение, указывающее текущее состояние управления устройством.

**Обоснование**

Устройство может работать автономно или управляться другой системой. Состояние управления устройством может быть передано другому устройству до дистанционной передачи команд управления. Если состояние управления устройством не имеет значение «R» (remote — дистанционное), то внешние команды будут игнорироваться.

***Пример — Местное, дистанционное.***

A.3.2.2.4 Device.alertLevelCode:: CE (0..1)

Словарный домен: DeviceAlertLevel

**Определение**

Значение, указывающее текущее состояние функционирования автоматизированного устройства.

**Примечания к использованию**

Допустимые значения атрибута зависят от устройства.

***Примеры — Нормальное, предупреждение, критическое.***

A.3.2.2.5 Device.lastCalibrationTime:: TS (0..1)

**Определение**

Дата и время последней калибровки устройства.

**Обоснование**

Чтобы гарантировать, что устройства работают в соответствии со спецификациями, их надо с определенной периодичностью калибровать. Допустимый интервал между калибровками зависит от регламентов. Следовательно, для обеспечения правильности результатов точные время и дату последней калибровки надо рассматривать как критичный параметр.

A.3.2.3 Класс Entity (в предметной области Entities)

**Свойства класса Entity**

Атрибуты класса Entity:

- classCode:: CS,
- determinerCode:: CS,
- id:: SET<II>,
- code:: CD,
- quantity:: SET<PQ>,
- name:: BAG<EN>,
- desc:: ED,
- statusCode:: CS,
- existenceTime:: IVL<TS>,
- telecom:: BAG<TEL>,
- riskCode:: SET<CE>,
- handlingCode :: CE.

Ассоциации класса Entity:

- communicationFunction::(0..*) CommunicationFunction::entity::(1..*) (ассоциация с классом Communication Function, роль entity — сущность);
- languageCommunication::(0..*) LanguageCommunication::entity::(1..1) (ассоциация с классом Language Communication, роль entity — сущность);
- playedRole::(0..*) Role::player::(0..1) (ассоциация с классом Role, роль player — исполнитель);
- scopedRole::(0..*) Role::scoper::(0..1) (ассоциация с классом Role, роль scoper — контролер).

Класс Entity является обобщением следующих классов:

- LivingSubject;
- Material;
- Organization;
- Place;
- EntityHeir.

Описание переходов состояния класса Entity приведено в подразделе A.3.2.3.13.

**Определение класса Entity**

Сущность (физический предмет, группа физических предметов или организация), способная участвовать в действиях, выполняя некоторую роль.

**Примечания к использованию**

Сущность — физический объект, который существует, существовал или будет существовать. Единственное исключение — организация, которая, не имея физического облика, обладает другими характеристиками класса Entity. Этот класс моделирует сам объект, а не роли, которые объект может выполнять. Например, роль пациента выполняется сущностью, описывающей физическое лицо.

***Примеры — Живые существа (включая людей), организации, материал, места и их специализации.***

**Атрибуты класса Entity**

A.3.2.3.1 Entity.classCode:: CS (1..1) Mandatory

Словарный домен: EntityClass

**Определение**

Код, определяемый стандартом HL7 и указывающий вид или категорию экземпляра класса Entity.

**Обоснование**

Вследствие чрезвычайно большого числа потенциальных значений, которые должна охватывать система кодирования, представляющая все физические объекты в мире, атрибут classCode указывает высокоуровневый классификатор экземпляра класса Entity, предназначенный для помещения этого экземпляра в определенный контекст. Его можно использовать для ограничения допустимых областей значений атрибута Entity.code.

***Примеры — Человек, животное, химическая субстанция, группа, организация.***

A.3.2.3.2 Entity.determinerCode:: CS (1..1) Mandatory

Словарный домен: EntityDeterminer

**Определение**

Код, определяемый стандартом HL7 и указывающий, представляет ли экземпляр класса Entity вид сущностей («KIND») или конкретный экземпляр сущности («INSTANCE»).

**Обоснование**

Экземпляр класса Entity может представлять информацию о конкретном экземпляре сущности (наиболее общий случай), о перечисляемой группе сущностей, обладающих общими характеристиками, или об общем виде сущностей.

***Примеры — 1 человек (экземпляр), 3 шприца (вид сущностей, количество которых определено) или население Индианаполиса (вид группы).***

A.3.2.3.3 Entity.id:: SET<II> (0..*)

**Определение**

Уникальный идентификатор экземпляра класса Entity.

**Примечания к использованию**

Идентификатор экземпляра сущности представляет собой чистый идентификатор, а не классификатор. Для представления серийных номеров, присваиваемых производителями, или артикулов в каталогах поставщиков, или инвентарных номеров материалов и товаров, информация о которых передается в экземплярах класса Material, могут использоваться атрибуты Role.id, что позволяет более ясно отразить тот факт, что такие номера или артикулы присвоены определенной стороной, связанной с этим материалом или товаром.

**Обоснование**

Для успешной передачи данных требуется лишь наличие у сущности уникального идентификатора. Но поскольку различные системы ведут различные базы данных, то одной и той же сущности в них могут быть присвоены разные идентификаторы.

A.3.2.3.4 Entity.code:: CD (0..1)

Словарный домен: EntityCode

**Определение**

Значение, указывающее определенный вид сущности, представляемой в форме экземпляра класса Entity.

**Примечания к использованию**

Система кодирования значений этого атрибута зависит от значения атрибута Entity.classCode, например, своя система для кодирования живых субъектов (таксономии животного и растительного мира), своя для химических субстанций (например, код IUPAC), своя для организаций (например, номер поставщика медицинской помощи, присвоенный организацией CMS) и т. д. В принципе, система кодирования значений Entity.code может быть до такой степени детализирована, что будет содержать единственное значение. Примером может служить код CDC производителя вакцины, моделируемый как словарь понятий, в котором каждое понятие фактически относится к единственному экземпляру. Граница между понятиями кода и идентификатора является достаточно спорной: в настоящем стандарте разрешается определенная степень гибкости.

***Примеры — Здание больницы, доберман-пинчер, пробирка для сбора крови, биоптат.***

A.3.2.3.5 Entity.quantity:: SET<PQ> (0..*)

**Определение**

Физическая величина, указывающая количество физических предметов, представленных экземпляром класса Entity, в виде числа членов в группе либо в виде иной физической величины. Чтобы явно идентифицировать группу похожих сущностей, конструкция статической модели должна ограничивать тип данных PQ, присвоенный атрибуту quantity, до целочисленного типа INT, тем самым позволяя задать число членов группы.

**Примечания к использованию**

Подобно тому, как физическое лицо может поменять свое именование и даже пол, количество сущности также может претерпевать изменения. Количество вещества или численность популяции может постепенно уменьшаться или разбиваться на меньшие количества той же самой сущности (например, аликвотирование в лаборатории или поставка продукции по частям). В случае разбиения на меньшие количества исходный экземпляр класса Entity, описывающий большее количество, может прекратить свое существование, однако полученные порции все еще могут отслеживаться до исходного экземпляра (до пациента в случае аликвотирования образца или до партии вакцины).

Указание атрибута Entity.quantity часто не является необходимым, поскольку количество можно задавать в отношениях к другим сущностям (например, в атрибуте Role.quantity), а также в действиях, которые потребляют или создают такие сущности (например, в атрибутах SubstanceAdministration.quantity, Supply.quantity).

***Примеры — 1 человек, 2 кота, 500 коров, 20 мл крови, 1 кг дрожжей, 200 предметов исследования.***

**Формальное ограничение**

В атрибуте quantity должен передаваться экстенсивный вид количества (например, число предметов) или делимые величины, например масса (1 кг), объем (1 л), количество субстанции (1 моль) или иная величина, пригодная для описания количества (каталитическая активность).

**Открытые вопросы**

Указание количества в терминах произвольной величины, зависящей от процедуры (например, туберкулиновые единицы), может не обеспечить надежную интерпретацию такой величины.

A.3.2.3.6 Entity.name:: BAG<EN> (0..*)

**Определение**

Неуникальный текстовый идентификатор или псевдоним экземпляра класса Entity.

**Обоснование**

Большинство сущностей имеет общеупотребительное название, которое может использоваться, чтобы отличить их от других сущностей, но не является уникальным идентификатором.

***Примеры — Собственные имена, псевдонимы, юридические фамилии, имена и отчества людей, названия мест или предметов.***

A.3.2.3.7 Entity.desc:: ED (0..1)

**Определение**

Текстовое или мультимедийное описание сущности.

**Примечания к использованию**

Содержание описания не рассматривается в качестве части функциональной информации, передаваемой другой системе. Оно предназначено для восприятия человеком. Вся информация, существенная для автоматизированных функций, должна передаваться в надлежащих атрибутах и ассоциированных объектах.

**Обоснование**

Имена и описания сущностей обычно более информативны для человека, читающего информацию, чем числовые, мнемонические или сокращенные кодированные значения. В атрибуте desc можно передать дополнительный контекст сущности, предназначенный человеку и не влияющий на функциональные компоненты сообщения.

A.3.2.3.8 Entity.statusCode:: CS (0..1)

Словарный домен: EntityStatus

**Определение**

Значение, указывающее, является ли информация, связанная с экземпляром класса Entity, активной или неактивной в отношении возможности участия в действиях.

**Примечания к использованию**

В исходной модели RIM этот атрибут был определен как повторяющийся, чтобы можно было отразить наличие вложенных подсостояний, указанных на диаграмме перехода состояний. На практике, однако, необходимость передачи нескольких значений состояния никогда не возникала. Поэтому комитетам рекомендуется ограничить кратность этого атрибута до 1 во всех конструкциях сообщений.

A.3.2.3.9 Entity.existenceTime:: IVL<TS> (0..1)

**Определение**

Интервал времени физического существования сущности.

**Примечания к использованию**

Физические сущности имеют определенные периоды существования. Люди рождаются, живут и умирают. Оборудование производится, вводится в эксплуатацию, выводится из эксплуатации и списывается. Значение этого атрибута полезно учитывать при планировании и оценке доступности, а также при ретроспективном анализе. Интервал времени может относиться к прошлому, настоящему и будущему.

***Примеры — Дата производства/дата списания.***

A.3.2.3.10 Entity.telecom:: BAG<TEL> (0..*)

**Определение**

Телекоммуникационный адрес сущности.

A.3.2.3.11 Entity.riskCode:: SET<CE> (0..*)

Словарный домен: EntityRisk

**Определение**

Значение, указывающее тип риска или опасности, связанной с сущностью.

***Примеры — Нефтехимические или органические химикаты являются очень огнеопасными, что в определенных условиях увеличивает риск пожара. Материалы с естественными или привнесенными радиоактивными свойствами представляют риск для тех, кто с ними обращается. Предметы, содержащие биоматериал больных людей, представляют повышенный риск инфекции для тех, кто обращается с ними. Вспыльчивые люди или животные могут представлять опасность для медицинского персонала.***

A.3.2.3.12 Entity.handlingCode:: SET<CE> (0..*)

Словарный домен: EntityHandling

**Определение**

Значение, указывающее специальные требования обработки сущности.

**Примечания к использованию**

Этот атрибут используется для описания требования специальной обработки сущности.

***Примеры — Хранить при комнатной температуре; хранить замороженным при температуре ниже 0 °С; хранить в сухом месте; хранить вертикально.***

A.3.2.3.13 Переходы состояний экземпляра класса Entity

Диаграмма перехода состояний класса Entity приведена на рисунке А.6.

Сущность может иметь следующие состояния:

- active (активное) — подсостояние состояния normal: сущность активна;
- inactive (неактивное) — подсостояние состояния normal: сущность более не может быть активным участником событий;

- normal (нормальное) — «типичное состояние», охватывающее все ожидаемые состояния, за исключением nullified, которое представляет терминальные состояния экземпляра класса Entity, созданного по ошибке;
- nullified (аннулированное) — терминальное состояние экземпляра класса Entity, созданного по ошибке.

Между состояниями действия возможны следующие переходы:
- revise (пересмотреть) — из состояния active в состояние active;
- inactivate (сделать неактивным) — из состояния active в состояние inactive;
- reactivate (активировать заново) — из состояния inactive в состояние active;
- revise (пересмотреть) — из состояния inactive в состояние inactive;
- nullify (аннулировать) — из состояния normal в состояние nullified;
- create (создать) — из начального (пустого) состояния в состояние active.

A.3.2.4 Класс LanguageCommunication (в предметной области Entities)

Атрибуты класса LanguageCommunication:
- languageCode:: CE,
- modeCode:: CE,
- proficiencyLevelCode:: CE,
- preferenceInd:: BL.

Ассоциации класса LanguageCommunication:
- entity::(1..1) Entity:: languageCommunication:: (0..*) (ассоциация с классом Entity, роль languageCommunication — язык общения).

Класс LanguageCommunication является специализацией класса InfrastructureRoot.

**Определение класса LanguageCommunication**

Способности сущности к языковому общению.

**Примечания к использованию**

Хотя на первый взгляд использование этого класса должно быть ограничено только подтипами класса LivingSubject (живой организм), но устройства, представляемые классом Device, также могут использовать языковое общение, например, автоматизированные телефонные устройства, перенаправляющие пациента к операторам в зависимости от его нужд, или устройства, автоматически читающие врачам результаты лабораторных анализов.

***Примеры — Пациент, ранее приехавший из Мексики, может иметь беглые языковые возможности чтения и письма на испанском языке, но при этом имеет только самые начальные способности общения на английском языке. Выходец из России может одинаково хорошо владеть разговорным русским, армянским или украинским языком, но предпочитает говорить на армянском языке.***

**Атрибуты класса LanguageCommunication**

A.3.2.4.1 LanguageCommunication.languageCode:: CE (0..1)

Словарный домен: HumanLanguage

**Определение**

Значение, указывающее язык, которым сущность в определенной степени владеет для разговорного или письменного общения.

**Примечания к использованию**

Разговорное или письменное языковое общение присуще не только живым существам. Для устройств, которые взаимодействуют с людьми, используя человеческий язык, также надо указать, на какие языки они рассчитаны. Автоматизированные системы речевого ответа передают сообщения на естественном языке и взаимодействуют с другими устройствами или людьми, используя естественный язык.

**Обоснование**

Многие люди и устройства способны общаться на разных языках, имея разные степени владения языками. Значение атрибута languageCode указывает способность к языковому общению, декларируемую сущностью.

***Примеры — Испанский, итальянский, немецкий, английский, американский язык жестов и т. д.***

A.3.2.4.2 LanguageCommunication.modeCode:: CE (0..1)

Словарный домен: LanguageAbilityMode

**Определение**

Значение, указывающее метод применения языка.

***Примеры — Речевое выражение, письменное выражение, выражение на языке жестов, восприятие речи, восприятие письменного языка, восприятие языка жестов.***

A.3.2.4.3 LanguageCommunication.proficiencyLevelCode:: CE (0..1)

Словарный домен: LanguageAbilityProficiency

**Определение**

Значение, указывающее степень владения языком.

***Примеры — Свободно, хорошо, удовлетворительно, плохо.***

A.3.2.4.4 LanguageCommunication.preferenceInd:: BL (0..1)

**Определение**

Признак, указывающий, является ли язык предпочтительным для метода применения, указанного в атрибуте modeCode.

A.3.2.5 Класс LivingSubject (в предметной области Entities)

**Свойства класса LivingSubject**

Атрибуты класса LivingSubject:

- administrativeGenderCode:: CE,
- birthTime:: TS,
- deceasedInd:: BL,
- deceasedTime:: TS,
- multipleBirthInd:: BL,
- multipleBirthOrderNumber:: INT,
- organDonorInd:: BL.

Класс LivingSubject является обобщением следующих классов:

- NonPersonLivingSubject;
- Person.

Класс LivingSubject является специализацией класса Entity.

**Определение класса LivingSubject**

Организм, живой или нет.

**Обоснование**

Этот класс содержит административные атрибуты, представляющие интерес для медицины и отличающие живые организмы от других сущностей.

***Примеры — Человек, собака, микроорганизм или растение любой таксономической группы.***

**Атрибуты класса LivingSubject**

A.3.2.5.1 LivingSubject.administrativeGenderCode:: CE (0..1)

Словарный домен: AdministrativeGender

**Определение**

Пол живого субъекта (то есть поведенческие, культурные или психологические черты, обычно ассоциируемые с одним полом), определяемый в административных целях.

**Ограничение использования**

Этот код используется в административных целях.

**Примечания к использованию**

Значения этого атрибута не включают в себя термины, относящиеся к клиническому полу. Пол — это сложное физиологическое, генетическое и социологическое понятие, для всестороннего описания которого необходим ряд наблюдений. Цель этого атрибута — обеспечить высокоуровневую классификацию, которая дополнительно может использоваться для соответствующего размещения пациентов на больничные койки.

Эта информация передается в поле FL 15 Универсального счета на оплату лечения (UB FL 15).

***Примеры — Женский, мужской.***

A.3.2.5.2 LivingSubject.birthTime:: TS (0..1)

**Определение**

Дата и время рождения или выведения живого организма.

A.3.2.5.3 LivingSubject.deceasedInd:: BL (0..1)

**Определение**

Признак смерти организма.

A.3.2.5.4 LivingSubject.deceasedTime:: TS (0..1)

**Определение**

Дата и время смерти живого организма.

A.3.2.5.5 LivingSubject.multipleBirthInd:: BL (0..1)

**Определение**

Признак, появился ли живой организм в результате многоплодных родов.

A.3.2.5.6 LivingSubject.multipleBirthOrderNumber:: INT (0..1)

**Определение**

Порядок рождения живого организма при многоплодных родах.

A.3.2.5.7 LivingSubject.organDonorInd:: BL (0..1)

**Определение**

Признак того, что живой организм является кандидатом в доноры органа.

**Примечания к конструированию**

*Примечания к использованию модифицировали определение и были удалены.*

A.3.2.6 Класс ManufacturedMaterial (в предметной области Entities)

**Свойства класса ManufacturedMaterial**

Атрибуты класса ManufacturedMaterial:

- lotNumberText:: ST,
- expirationTime:: IVL<TS>,
- stabilityTime:: IVL<TS>.

Класс ManufacturedMaterial является обобщением следующих классов:

- Container;
- Device.

Класс ManufacturedMaterial является специализацией класса Material.

**Определение класса ManufacturedMaterial**

Сущность или комбинация сущностей, преобразованная в специальных целях с помощью производственного процесса.

**Примечания к использованию**

К понятиям этого класса относятся контейнеры, медицинские изделия, программные модули и средства. Он используется для дальнейшего определения характеристик сущностей, которые созданы из других сущностей. Эти сущности идентифицируются и отслеживаются с помощью ассоциаций и механизмов, уникальных для данного класса, например, с помощью атрибутов lotName, stabilityTime и expirationTime.

***Примеры — Обработанные пищевые продукты, одноразовые шприцы, химический анализатор, физиологический раствор и т. д.***

**Атрибуты класса ManufacturedMaterial**

A.3.2.6.1 ManufacturedMaterial.lotNumberText:: ST (0..1)

**Определение**

Идентификатор конкретной партии изготовленного материала.

**Примечания к использованию**

Номер партии обычно печатается на ярлыке, приложенном к контейнеру, содержащему субстанцию, и/или на упаковке, в которую вложен контейнер. Следует отметить, что номер партии не обязан быть уникальным идентификатором, он имеет смысл только в сочетании с видом продукта и производителем.

A.3.2.6.2 ManufacturedMaterial.expirationTime:: IVL<TS> (0..1)

**Определение**

Дата и время, после которого производитель больше не гарантирует безопасность, качество и/или надлежащее функционирование материала.

**Обоснование**

Во многих ситуациях требуется, чтобы используемые материалы имели конкретное качество или обладали определенной эффективностью либо имели необходимое функциональное состояние. Конечная дата этой гарантии определяется производителем. Хотя после указанной даты материал может все еще обладать теми же самыми свойствами, производитель с этого момента снимает с себя ответственность за надлежащее функционирование материала и возможные последствия его использования.

A.3.2.6.3 ManufacturedMaterial.stabilityTime:: IVL<TS> (0..1)

**Определение**

Период, в течение которого материал считается пригодным к использованию после активирования.

**Примечания к использованию**

Если экземпляр класса ManufacturedMaterial описывает вид материала (его атрибут determinerCode имеет значение «KIND»), то может быть известна только длительность интервала, например, время после открытия контейнера с реагентом, в течение которого вещество реагента считается пригодным для употребления в его стандартном аналитическом применении. Привязка к календарю с помощью штампов даты и времени невозможна, компонент stabilityTime.low типа TS может быть равен нулю, а компонент stabilityTime.high типа TS может содержать скалярную величину длительности интервала.

Для определенного экземпляра реагента (атрибут determinerCode имеет значение «INSTANCE») значение компонента stabilityTime.low типа TS указывает время, когда флакон был открыт (или реагент был активирован иным образом). Добавляя к нему типичное время стабильности («KIND»), можно получить значение stabilityTime.high типа TS, после которого реагент уже не считается пригодным для употребления в его стандартном аналитическом применении.

**Примечания к конструированию**

Следует подтвердить экстраполяцию примечаний к использованию.

***Пример — Два химических реагента, которые должны быть использованы в течение двух часов после смешивания, иначе активность их смеси снижается.***

A.3.2.7 Класс Material (в предметной области Entities)

**Свойства класса Material**

Атрибуты класса Material:

- formCode:: CE.

Класс Material является обобщением класса ManufacturedMaterial.

Класс Material является специализацией класса Entity.

**Определение класса Material**

Специализация класса Entity, описывающая неодушевленную сущность, не зависящую от местонахождения.

**Примечания к использованию**

К классу Material относятся сущности, не являющиеся живыми существами или местонахождениями. Изготовленные или обработанные продукты считаются материалом, даже если они происходят от живой материи. Существует широкое разнообразие физических форм материалов, которые могут принимать различные состояния (например, газ, жидкость, твердое вещество), все еще сохраняя свой физический состав и материальные характеристики.

**Примечания к конструированию**

Следует уточнить смысл характеристики «независимость от местонахождения»; рассмотреть возможность удаления ее и замены на первое предложение из примечаний к использованию.

***Примеры — Фармацевтические субстанции (включая активные вакцины, содержащие ослабленные вирусные клетки), одноразовые предметы, оборудование длительного пользования, имплантанты, продукты питания (включая мясные или растительные продукты), отходы, продаваемые товары.***

**Атрибуты класса Material**

A.3.2.7.1 Material.formCode:: CE (0..1)

Словарный домен: MaterialForm

**Определение**

Значение, указывающее физическое состояние и природу материала.

***Примеры — Твердое, жидкое, газообразное, таблетка, мазь, гель.***

A.3.2.8 Класс NonPersonLivingSubject (в предметной области Entities)

**Свойства класса NonPersonLivingSubject**

Атрибуты класса NonPersonLivingSubject:

- strainText:: ED,
- genderStatusCode:: CE.

Класс NonPersonLivingSubject является специализацией класса LivingSubject.

**Определение класса NonPersonLivingSubject**

Специализация класса LivingSubject, описывающая все живые организмы, кроме человека.

**Обоснование**

Для описания живого организма, не являющегося человеком, может понадобиться дополнительная характерная информация, например генетическая идентификация рода, которая не может быть передана в атрибуте code.

***Примеры — Крупный рогатый скот, птицы, бактерии, растения, грибки и грибы.***

**Атрибуты класса NonPersonLivingSubject**

A.3.2.8.1 NonPersonLivingSubject.strainText:: ED (0..1)

**Определение**

Определенный генотипический или фенотипический вариант организма, информация о котором передается в экземпляре класса NonPersonLivingSubject.

**Обоснование**

Универсальное руководство по именованию или каталогизированию породы, вида или штамма организмов отсутствует. Многие обозначения пород, видов и штаммов создавались, а затем исчезали, а другие по разным причинам приживались в различных областях производства (изготовление вакцин, племенное животноводство и т. д.). И поскольку каждый, кому это нужно, может обозначить организм как «новый» вид, вместо кодированного значения данный атрибут должен содержать описательный текст, в котором можно передать все такие обозначения.

***Примеры — Миннесота 5 (порода поросят), DXL (порода домашней птицы), RB51 (штамм противобруцеллезной вакцины).***

A.3.2.8.2 NonPersonLivingSubject.genderStatusCode:: CE (0..1)

Словарный домен: GenderStatus

**Определение**

Значение, указывающее наличие основных репродуктивных органов у организма, информация о котором передается в экземпляре класса NonPersonLivingSubject.

A.3.2.9 Класс Organization (в предметной области Entities)

**Свойства класса Organization**

Атрибуты класса Organization:

- addr:: BAG<AD>,
- standardIndustryClassCode:: CE.

Класс Organization является специализацией класса Entity.

**Определение класса Organization**

Специализация класса Entity, описывающая формализованную группу сущностей с общей целью (например, административной, юридической, политической) и инфраструктуру, необходимую для осуществления этой цели.

***Примеры — Предприятия и организации, правительственная структура, корпоративный орган, ответственный за управление учреждением, страховая компания.***

**Атрибуты класса Organization**

A.3.2.9.1 Organization.addr:: BAG<AD> (0..*)

**Определение**

Почтовый и/или юридический адрес организации.

A.3.2.9.2 Organization.standardIndustryClassCode:: CE (0..1)

Словарный домен: OrganizationIndustryClass

**Определение**

Значение, указывающее код организации согласно классификатору отраслей народного хозяйства.

***Пример — Коды организации NAICS (к примеру, 11231 — производство куриных яиц, 6211 — амбулаторная помощь, 621511 — медицинские лаборатории).***

A.3.2.10 Класс Person (в предметной области Entities)

**Свойства класса Person**

Атрибуты класса Person:

- addr:: BAG<AD>,
- maritalStatusCode:: CE,
- educationLevelCode:: CE,
- disabilityCode:: SET<CE>,
- livingArrangementCode:: CE,
- religiousAffiliationCode :: CE,
- raceCode:: SET<CE>,
- ethnicGroupCode:: SET<CE>.

Класс Person является специализацией класса LivingSubject.

**Определение класса Person**

Специализация класса LivingSubject, описывающая человека.

**Примечания к использованию**

В зависимости от значений атрибутов determinerCode и quantity экземпляр этого класса может содержать информацию об одном лице, группе лиц или категории лиц.

**Атрибуты класса Person**

A.3.2.10.1 Person.addr:: BAG<AD> (0..*)

**Определение**

Почтовый адрес или адрес места жительства лица.

A.3.2.10.2 Person.maritalStatusCode:: CE (0..1)

Словарный домен: MaritalStatus

**Определение**

Значение, указывающее статус семейного партнерства лица.

**Примечания к использованию**

Эта информация передается в поле FL 16 Универсального счета на оплату лечения (UB).

***Примеры — Женат (замужем), раздельное проживание, в разводе, вдовец (вдова), гражданский брак.***

A.3.2.10.3 Person.educationLevelCode:: CE (0..1)

Словарный домен: EducationLevel

**Определение**

Самый высокий уровень образования, достигнутый лицом.

***Примеры — Начальная школа, средняя школа или законченное среднее образование, колледж или законченное обучение в бакалавриате.***

A.3.2.10.4 Person.disabilityCode:: SET<CE> (0..*)

Словарный домен: PersonDisabilityType

**Определение**

Значение, указывающее инвалидность человека.

***Примеры — Ослабленное зрение, ослабленный слух.***

A.3.2.10.5 Person.livingArrangementCode:: CE (0..1)
Словарный домен: LivingArrangement
**Определение**
Значение, указывающее условия проживания лица.
**Примечания к использованию**
Используется при планировании выписки, оценке необходимости социального обеспечения, психосоциальной оценке.

***Примеры — Собственный дом, организация, интернат, реабилитационный центр, дом престарелых.***

A.3.2.10.6 Person.religiousAffiliationCode:: CE (0..1)
Словарный домен: ReligiousAffiliation
**Определение**
Основное религиозное предпочтение лица.

***Примеры — индуизм, ислам, римско-католическая церковь.***

A.3.2.10.7 Person.raceCode:: SET<CE> (0..*)
Словарный домен: Race
**Определение**
Раса лица.
A.3.2.10.8 Person.ethnicGroupCode:: SET<CE> (0..*)
Словарный домен: Ethnicity
**Определение**
Этническая группа лица.
A.3.2.11 Класс Place (в предметной области Entities)
**Свойства класса Place**
Атрибуты класса Place:
- mobileInd:: BL,
- addr:: BAG<AD>,
- directionsText:: ED,
- positionText:: ED,
- gpsText:: ST.

Класс Place является специализацией класса Entity.
**Определение класса Place**
Специализация класса Entity, описывающая ограниченное физическое место или участок с содержащимися в нем структурами, если таковые имеются.
**Примечания к использованию**
Место может быть естественным или созданным человеком. Географическое положение места может быть непостоянным. Места могут быть производственными зданиями (где выполняются соответствующие действия), домами (где живут люди) или конторами (где люди работают). Места могут содержать другие места (этаж, палата, кабина, койка). Экземпляры класса Place могут содержать информацию о местах, идентифицируемых в контексте оказания медицинской помощи, социальной помощи, санитарно-эпидемиологических мероприятий (например, здания, поляны для пикника, дневные стационары, тюрьмы, округа, штаты и другие места эпидемиологических событий).

***Примеры — Поле, озеро, город, графство, штат, страна, участок (земли), здание, трубопровод, линия электропередачи, детская площадка, судно, грузовик.***

**Атрибуты класса Place**
A.3.2.11.1 Place.mobileInd:: BL (0..1)
**Определение**
Признак того, может ли место свободно перемещаться из одного местоположения в другое.

***Примеры — Перемещаемыми местами оказания медицинской помощи могут быть суда, самолеты и санитарные машины.***

A.3.2.11.2 Place.addr:: AD (0..1)
**Определение**
Физический адрес данного места.
**Ограничение использования**
Значением этого атрибута должен быть адрес, с помощью которого можно определить физическое положение места на карте.
A.3.2.11.3 Place.directionsText:: ED (0..1)
**Определение**
Свободный текст, содержащий информацию о данном месте, которая полезна для сущностей при поиске этого места.

**Примечания к использованию**

В этом атрибуте могут передаваться указания для поиска места, если информация об адресе неадекватна, его координаты в глобальной системе навигации (GPS) недоступны и/или сущность, которая ищет это место, не может непосредственно использовать координаты GPS. В нем можно передавать информацию, полезную для посетителей места.

***Примеры — последний дом справа; если владельца нет, узнайте в следующем доме, где его найти.***

A.3.2.11.4 Place.positionText:: ED (0..1)

**Определение**

Совокупность кодов, задающих расположение места на карте.

***Примеры — Координаты на картах, издаваемых агентством US Geological Survey.***

A.3.2.11.5 Place.gpsText:: ST (0..1)

***Документирование***

**Примечания к использованию**

Значения координат в глобальной системе навигации должны соответствовать стандартам USGS Spatial Data Transmission (передача пространственных данных). К числу этих параметров относятся способы получения значений широты и долготы, ошибки сдвига, проекция.

**Обоснование**

В некоторых полевых условиях физический адрес интересующего места может отсутствовать. Поскольку все места поверхности земли имеют уникальные географические координаты, для точного определения и передачи информации о положении места можно использовать GP-координаты.

**A.3.3 Классы предметной области Roles**

A.3.3.1 Класс Access (в предметной области Roles)

**Свойства класса Access**

Атрибуты класса Access:

- approachSiteCode:: CD,
- targetSiteCode:: CD,
- gaugeQuantity:: PQ.

Класс Access является специализацией класса Role.

**Определение класса Access**

Специализация класса Role, описывающая роль устройства, используемого для введения терапевтических агентов (лекарств и элементов, поддерживающих жизнь) в тело пациента, или изделия, используемого для вывода веществ (например, экссудатов, гноя, мочи, воздуха, крови) из его тела.

**Примечания к использованию**

В общем случае класс Access описывает роль произведенного материала или изделия, иногда специально изготовленного или созданного для определенной цели, например, роль катетера или канюли, помещенной в полый орган тела. Изделия, выполняющие роль доступа к органам тела, обычно используются в наблюдениях поглощения/оттока, а также для введения лекарственных средств. Микробиологические исследования самого материала или жидкостей, выделяемых из дренажа, также являются обычными примерами.

Роль, представляемая классом Access, в основном используется для описания материала, фактически применяемого для доступа, а не нового материала, полученного от изготовителя. Например, для представления заказа коробки катетеров у поставщика не обязательно использовать класс роли Access, так как атрибутов класса Material обычно вполне достаточно для описания и идентификации заказываемого изделия. А экземпляры класса Access используются для передачи данных об эксплуатации, поглощении/оттоке и своевременной замене трубок и дренажа.

**Атрибуты класса Access**

A.3.3.1.1 Access.approachSiteCode:: CD (0..1)

Словарный домен: ActSite

**Определение**

Кодированное представление анатомической локализации места, через которое изделие, участвующее в доступе, впервые проникает в тело и, если применимо, маршрут от места первого входа к целевому месту.

**Обоснование**

Поскольку изделия, обеспечивающие доступ, обычно помещаются в тело на значительный период времени и используются как ресурс для многих действий, то целевое место доступа становится важным идентифицирующим атрибутом самого доступа (в противоположность тому, когда оно является атрибутом процедуры введения).

***Примеры — Катетер легочной артерии предназначен для легочной артерии, но при этом местом доступа обычно является внутренняя яремная вена в шее или подключичная вена в подключичной ямке.***

**Формальное ограничение**

Система кодирования значений атрибута Access.approachSiteCode та же, что у атрибута Procedure. approachSiteCode; действительно, атрибут approachSiteCode был скопирован из класса Procedure в класс Access. Значение атрибута Access.approachSiteCode должно быть идентично значению атрибута Procedure. approachSiteCode соответствующей процедуры введения изделия.

A.3.3.1.2 Access.targetSiteCode:: CD (0..1)

Словарный домен: ActSite

**Определение**

Кодированное указание анатомической локализации места или части тела, куда обеспечивается доступ, то есть часть, в которую вводится материал, или часть, из которой он выводится.

**Обоснование**

Поскольку изделия, обеспечивающие доступ, обычно помещаются в тело на значительный период времени и используются как ресурс для многих действий, то целевое место доступа становится важным идентифицирующим атрибутом самого доступа (в противоположность тому, когда оно является атрибутом процедуры введения). Целевое место является важной информацией, по которой можно установить вид субстанций, которые могут или не могут быть применены в этом месте (например, должны быть предприняты специальные меры во избежание инъекции лекарства в изделие, находящееся в артерии).

***Примеры — Катетер легочной артерии будет иметь целевым местом «легочную артерию».***

**Формальное ограничение**

Система кодирования значений атрибута Access.targetSiteCode та же, что у атрибута Procedure.targetSiteCode; действительно, атрибут targetSiteCode был скопирован из класса Procedure в класс Access. Значение атрибута Access.targetSiteCode должно быть идентично значению атрибута Procedure.targetSiteCode соответствующей процедуры введения устройства.

A.3.3.1.3 Access.gaugeQuantity:: PQ (0..1)

**Определение**

Мера внутреннего диаметра устройства доступа.

***Пример — Просвет трубки.***

A.3.3.2 Класс Employee (в предметной области Roles)

**Свойства класса Employee**

Атрибуты класса Employee:

- jobCode:: CE,
- jobTitleName:: SC,
- jobClassCode:: CE,
- occupationCode :: CE,
- salaryTypeCode:: CE,
- salaryQuantity:: MO,
- hazardExposureText:: ED,
- protectiveEquipmentText:: ED.

Класс Employee является специализацией класса Role.

**Определение класса Employee**

Специализация класса Role, описывающая роль лица, связанного с организацией для получения заработной платы рабочего или служащего.

**Примечания к использованию**

Организация-работодатель имеет роль контролера. Класс Employee предназначен для описания типа трудовых отношений между работником и работодателем, а не для описания характера фактически выполненной работы (в противоположность классу AssignedEntity).

**Атрибуты класса Employee**

A.3.3.2.1 Employee.jobCode:: CE (0..1)

Словарный домен: EmployeeJob

**Определение**

Код, указывающий вид работы, классифицируемый работодателем.

**Примечания к использованию**

Этот код используется в первую очередь для целей установления зарплаты/вознаграждения, а не для определения специфики выполняемой работы, ответственности и полномочий работника.

***Примеры — Бухгалтер, аналитик-программист, сиделка, штатная медсестра.***

A.3.3.2.2 Employee.jobTitleName:: SC (0..1)

Словарный домен: JobTitleName

**Определение**

Название занимаемой должности.

**Примечания к использованию**

Это местное название должности работника, которое не обязательно соответствует какой-либо номенклатуре должностей. Торговые партнеры, которым требуется кодированное представление, должны использовать атрибут occupationCode.

***Примеры — Вице-президент, старший технический аналитик.***

A.3.3.2.3 Employee.jobClassCode:: CE (0..1)

Словарный домен: EmployeeJobClass

**Определение**

Код, указывающий частоту или периодичность занятости.

***Примеры — Полная ставка, совместитель.***

A.3.3.2.4 Employee.occupationCode :: CE (0..1)

Словарный домен: EmployeeOccupationCode

**Определение**

Код, указывающий классификацию вида работы, основанную на территориальной или отраслевой номенклатуре.

A.3.3.2.5 Employee.salaryTypeCode:: CE (0..1)

Словарный домен: EmployeeSalaryType

**Определение**

Код, указывающий метод начисления заработной платы рабочему или служащему.

***Примеры — Почасовая ставка, годовая ставка, комиссионные.***

A.3.3.2.6 Employee.salaryQuantity:: MO (0..1)

**Определение**

Сумма заработной платы рабочего или служащего.

**Примечания к использованию**

Эта сумма должна определяться в соответствии с методом, указанным в значении атрибута salaryTypeCode. Например, если атрибут salaryTypeCode имеет значение «почасовая», то в атрибуте salaryQuantity передается сумма почасовой ставки.

A.3.3.2.7 Employee.hazardExposureText:: ED (0..1)

**Определение**

Тип вредности выполняемой работы.

***Примеры — Асбест, инфекционные агенты.***

A.3.3.2.8 Employee.protectiveEquipmentText:: ED (0..1)

**Определение**

Средства защиты, необходимые для выполнения работы.

***Примеры — Защитные очки, каска.***

A.3.3.3 Класс LicensedEntity (в предметной области Roles)

**Свойства класса LicensedEntity**

Атрибуты класса LicensedEntity:

- recertificationTime:: TS.

Класс LicensedEntity является специализацией класса Role.

**Определение класса LicensedEntity**

Специализация класса Role, описывающая роль сущности, обладающей лицензией или квалификацией (дипломом), подтверждающей право данной сущности выполнять специфичные функции.

**Примечания к использованию**

Исполнителем роли является квалифицированная сущность, контролером — организация, наделившая полномочиями. Лицензирование является частным случаем присвоения квалификации.

***Примеры***

***1 Дипломированный фельдшер скорой помощи.***

***2 Оборудование, имеющее сертификат безопасности.***

***3 Медицинский работник, имеющий сертификат специалиста, или организация, имеющая лицензию на право ведения медицинской деятельности.***

**Атрибуты класса LicensedEntity**

A.3.3.3.1 LicensedEntity.recertificationTime:: TS (0..1)

**Определение**

Требуемая дата повторной сертификации.

A.3.3.4 Класс Patient (в предметной области Roles)

**Свойства класса Patient**

Атрибуты класса Patient:

- veryImportantPersonCode:: CE.

Класс Patient является специализацией класса Role.

**Определение класса Patient**

Специализация класса Role, описывающая роль живого существа, получающего медицинскую помощь от ее поставщика.

**Атрибуты класса Patient**

A.3.3.4.1 Patient.veryImportantPersonCode:: CE (0..1)

Словарный домен: PatientImportance

**Определение**

Код, указывающий особый статус пациента, предоставленный ему контролирующей организацией.

**Обоснование**

Этот статус часто обеспечивает приоритетное лечение и особые условия оказания медицинской помощи.

***Примеры — Член правления, дипломат.***

A.3.3.5 Класс QualifiedEntity (в предметной области Roles)

**Свойства класса QualifiedEntity**

Атрибуты класса QualifiedEntity:

- equivalenceInd :: BL.

Класс QualifiedEntity является специализацией класса Role.

**Определение класса QualifiedEntity**

Сущность, признаваемая имеющей определенные навыки, опыт или иные характеристики, позволяющие ей быть подходящим исполнителем определенной деятельности.

**Примечания к использованию**

Квалифицированная сущность является исполнителем роли, образовательная или лицензирующая организация является контролером роли. Квалифицированная сущность — более общее понятие по сравнению с лицензированной сущностью.

**Атрибуты класса QualifiedEntity**

A.3.3.5.1 QualifiedEntity.equivalenceInd :: BL:: (0..1)

Словарный домен: PatientImportance

**Определение**

Признак, указывающий, что контролирующая сущность признает сочетание квалификаций эквивалентным нормальному определению условий назначения роли.

**Примечания к использованию**

Значение «True» атрибута equivalenceInd указывает, что контролирующая сущность объявляет, что она признает сочетание квалификаций эквивалентным нормальному определению условий назначения роли, характеризуемой значением атрибута code.

**Примечания к конструированию**

Как трактовать этот атрибут, если его значение не равно «True»?

***Примеры — Степень бакалавра (эквивалент), бакалавр сестринского дела (эквивалент), бакалавр стоматологической хирургии (эквивалент).***

A.3.3.6 Класс Role (в предметной области Roles)

**Свойства класса Role**

Атрибуты класса Role:

- classCode:: CS,
- id:: SET<II>,
- code:: CE,
- negationInd:: BL,
- name:: BAG<EN>,
- addr:: BAG<AD>,
- telecom:: BAG<TEL>,
- statusCode:: CS,
- effectiveTime:: IVL<TS>,
- certificateText:: ED,
- confidentialityCode :: SET<CE>,
- quantity:: RTO,

- positionNumber:: LIST<INT>.

Ассоциации класса Role:

- participation::(0..*) Participation::role::(1..1) (ассоциация с классом Participation, роль role — роль),
- player::(0..1) Entity::playedRole::(0..*) (ассоциация с классом Entity, роль playedRole — выполняемая роль),
- scoper::(0..1) Entity::scopedRole::(0..*) (ассоциация с классом Entity, роль scopedRole — контролируемая роль),
- outboundLink::(0..*) RoleLink::source::(1..1) (ассоциация с классом RoleLink, роль source — источник),
- inboundLink::(0..*) RoleLink::target::(1..1) (ассоциация с классом RoleLink, роль target — цель).

Класс Role является обобщением следующих классов:

- Access;
- Employee;
- LicensedEntity;
- Patient;
- QualifiedEntity;
- RoleHeir.

**Определение класса Role**

Компетенция сущности, выполняющей роль в соответствии с указаниями, определениями, гарантиями или признанием другой сущности, контролирующей эту роль.

**Примечания к использованию**

Сущность участвует в действии, выполняя определенную роль. Следует отметить, что конкретная сущность в конкретной роли может участвовать в действии многими способами. Например, лицо в роли врача может участвовать в ведении пациента как дежурант или как лечащий врач. Роль определяет компетенцию сущности, не зависящую от любого действия, в противоположность участию, которое ограничено рамками действия.

Каждая роль «выполняется» одной сущностью, называемой «исполнителем», и «контролируется» другой сущностью, называемой «контролером». Например, роль «пациент» может выполняться некоторым лицом и контролироваться организацией, оказывающей пациенту медицинскую помощь. Аналогичным образом работодатель контролирует роль «работника».

Идентификатор роли идентифицирует сущность, выполняющую эту роль. Этот идентификатор присваивается исполнителю контролирующей сущностью. Контролирующая сущность НЕ ОБЯЗАНА создавать новые идентификаторы. Она МОЖЕТ повторно использовать идентификатор, ранее присвоенный контролируемой сущности, чтобы идентифицировать эту сущность в контролируемой ею роли.

Большинство атрибутов класса Role представляют собой характеристики исполнителя роли, присущие ему в процессе выполнения конкретной роли.

**Атрибуты класса Role**

A.3.3.6.1 Role.classCode:: CS (1..1) Mandatory

Словарный домен: RoleClass

**Определение**

Код, указывающий основную категорию роли, к которой принадлежит экземпляр класса Role.

A.3.3.6.2 Role.id:: SET<II> (0..*)

**Определение**

Уникальный идентификатор сущности, выполняющей данную роль.

A.3.3.6.3 Role.code:: CE (0..1)

Словарный домен: RoleCode

**Определение**

Конкретный вид роли, к которому принадлежит экземпляр класса Role.

**Ограничение использования**

Значения атрибута Role.code должны концептуально представлять собой специализации понятия, указанного в атрибуте Role.classCode.

**Примечания к использованию**

Атрибут Role.code не должен использоваться как модификатор значения атрибута Role.classCode. Каждый из этих атрибутов представляет завершенное понятие или ролевые отношения между двумя сущностями, но значение атрибута Role.code может быть более специфичным, нежели значение атрибута Role.classCode.

Значение Role.code может быть не кодированным, если данный тип роли обычно обозначается только не кодированным именем.

**Примечания к конструированию**

Тип данных CE может стать запрещенным; в типе данных CD код обязателен.

A.3.3.6.4 Role.negationInd:: BL (0..1)

**Определение**

Признак, указывающий, что данный экземпляр класса Role описывает компетенцию, которая отсутствует у сущности, выполняющей эту роль.

**Примечания к конструированию**

Следует подтвердить аннотацию к значению по умолчанию.

***Примеры***

***1 Это лицо не является нашим работником.***

***2 Эта жидкость для полоскания рта не содержит алкоголь в качестве ингредиента.***

A.3.3.6.5 Role.name:: BAG<EN> (0..*)

**Определение**

Неуникальный текстовый идентификатор или псевдоним сущности, предназначенный прежде всего для использования во время выполнения роли.

**Примечания к использованию**

Вообще говоря, именования передаются в атрибуте Entity.name. Атрибут Role.name используется только в том случае, когда необходимо различать именования, которые уместно использовать при выполнении конкретной роли, но не в других ролях.

***Примеры — Именование, используемое в роли работника; именование, используемое в роли сертифицированного специалиста, и т. д.***

A.3.3.6.6 Role.addr:: BAG<AD> (0..*)

**Определение**

Почтовый адрес сущности во время выполнения роли.

A.3.3.6.7 Role.telecom:: BAG<TEL> (0..*)

**Определение**

Телекоммуникационный адрес сущности во время выполнения роли.

A.3.3.6.8 Role.statusCode:: CS (0..1)

Словарный домен: RoleStatus

**Определение**

Код, указывающий состояние данной роли, определенное в модели перехода состояний.

**Примечания к использованию**

В исходной модели RIM этот атрибут был определен как повторяющийся, чтобы можно было отразить наличие вложенных подсостояний, указанных на диаграмме перехода состояний. На практике, однако, необходимость передачи нескольких значений состояния никогда не возникала. Поэтому комитетам рекомендуется ограничить кратность этого атрибута до 1 во всех конструкциях сообщений.

A.3.3.6.9 Role.effectiveTime:: IVL<TS> (0..1)

**Определение**

Интервал времени выполнения роли, если такое время применимо и известно.

A.3.3.6.10 Role.certificateText:: ED (0..*)

**Определение**

Текстовое или мультимедийное представление сертификата, изданного сущностью, контролирующей данную роль, и удостоверяющего, что данная роль действительно принадлежит исполнителю.

**Примечания к использованию**

Субъектом сертификата является сущность, выполняющая данную роль. Издателем сертификата является сущность, контролирующая данную роль.

***Пример — Сертификат может быть представлен многими разными способами либо по значению, либо по ссылке в соответствии с типом данных ED. Типичные случаи таковы:***

***1) бумажный сертификат: значение, имеющее тип данных ED, может содержать ссылку на некоторый документ или файл, который может быть найден с помощью электронного интерфейса к архиву бумажных документов;***

***2) электронный сертификат: в этом атрибуте можно передать практически любую электронную схему сертификата, например, электронный текстовый документ с электронной подписью (в том числе квалифицированной);***

***3) цифровой сертификат (сертификат открытого ключа): в частности, в этом атрибуте можно передавать цифровые сертификаты по значению или ссылке. Блок данных сертификата может конструироваться в соответствии со стандартом цифровых сертификатов, например, X.509, SPKI, PGP и т. д.***

A.3.3.6.11 Role.confidentialityCode:: SET<CE> (0..*)

Словарный домен: Confidentiality

**Определение**

Код, контролирующий раскрытие информации о данной роли сущности.

**Примечания к использованию**

Сокрытие информации может вызвать серьезные последствия для предоставления медицинской помощи, поэтому к нему надо прибегать осторожно.

Необходимая конфиденциальность медицинской карты не может быть достигнута только с помощью кодов конфиденциальности, предназначенных для маскировки отдельных частей этой карты от определенных типов пользователей. Фильтрованная чувствительная информация нередко может быть выведена из другой, не отфильтрованной информации. Простейшей формой вывода может служить пример, когда из факта наличия направления на анализ иммуноблота или на анализ количества Т4- и Т8-клеток лимфоцитов с большой вероятностью можно сделать вывод, что пациент ВИЧ-инфицирован, даже если результат анализа неизвестен. Очень часто диагнозы могут быть выведены из лекарственных назначений, например, назначение зидовудина свидетельствует о лечении ВИЧ. Чтобы ослабить некоторые риски вывода, следует считать, что агрегированные данные имеют уровень конфиденциальности самого конфиденциального действия в агрегате.

A.3.3.6.12 Role.quantity:: RTO (0..1)

**Определение**

Отношение (числитель:знаменатель), указывающее относительное количество исполнителя роли в сущности, контролирующей роль. Оно используется для ролей, описывающих отношения композиции между контролирующими сущностями и исполнителями.

**Примечания к использованию**

В отношениях композиции (например, имеет части, имеет ингредиент, имеет содержание) значение атрибута Role.quantity указывает, что количество числителя целевой сущности отношения содержит в себе количество сущности-источника, указанное в знаменателе. Например, если коробка (источник отношения) «имеет содержание» 10 яиц (цель отношения), то атрибут quantity должен иметь значение 10:1; если 0,6 мл смеси содержат 75 мг $FeSO_4$, то атрибут quantity должен иметь значение 75мг:0,6 мл. Как числитель, так и знаменатель должны быть количественными величинами (экстенсивными величинами, например, число подсчетов, масса, объем, количество субстанции, количество энергии и т. д.).

***Примеры***

***1 Ингредиентом этого сиропа (контролирующая сущность) являются 160 мг (числитель) ацетаминофена (исполнитель) на столовую ложку (знаменатель).***

***2 Стадо (контролирующая сущность) состоит из 500 (числитель) голов крупного рогатого скота (исполнитель).***

***3 У компьютера VAX 6630 (контролирующая сущность) имеются 3 (числитель) центральных процессора (исполнитель).***

***4 Эта упаковка (контролирующая сущность) содержит 100 (числитель) таблеток (исполнитель).***

A.3.3.6.13 Role.positionNumber:: LIST<INT> (0..*)

**Определение**

Целое число, указывающее положение исполнителя роли относительно сущности, контролирующей эту роль.

**Примечания к использованию**

Этот атрибут в основном используется в экземплярах класса Role, описывающих содержание в чем-либо. Например, у некоторых контейнеров есть дискретные положения, в которых может быть размещено содержимое. В зависимости от геометрии контейнера для указания конкретного положения может использоваться скалярное порядковое число или вектор порядковых чисел (координаты). Начальным значением координаты всегда должно быть число 1.

Для некоторых контейнеров может использоваться собственный способ нумерации положений; способа нумерации может и не быть. В отсутствие какого-либо определенного способа позиционирования у конкретного типа контейнеров эмпирическое правило нумерации состоит в том, что координата, изменяющаяся раньше, должна позиционироваться первой. Для автоматизированного биохимического анализатора крови со штативом квадратной формы это означает, что первой координатой будет та, в каком направлении перемещается штатив на каждом шаге, а второй — та, в каком направлении штатив перемещается только время от времени.

A.3.3.6.14 Переходы состояний экземпляра класса Role (атрибут состояния — statusCode)

Диаграмма перехода состояний класса Role приведена на рисунке А.8.

Роль может иметь следующие состояния:

- active (активна) — подсостояние состояния normal: сущность в настоящий момент активна в данной роли;
- cancelled (отменена) — подсостояние состояния normal: роль была отменена до того, как стала активной;
- normal (нормальное) — «типичное» состояние. Исключает состояние nullified, которое указывает, что экземпляр класса Role был создан по ошибке;
- nullified (аннулирована) — это состояние является терминальным состоянием экземпляра класса Role, созданного по ошибке;
- pending (готовящаяся) — подсостояние состояния normal: это состояние отражает тот факт, что роль еще не стала активной;
- suspended (приостановлена) — подсостояние состояния normal: выполнение роли временно приостановлено. Переход в это состояние возможен из состояния active (активно);
- terminated (завершена) — подсостояние состояния normal: успешное завершение выполнения роли.

Между состояниями действия возможны следующие переходы:

- revise (пересмотреть) — из состояния active в состояние active;

- suspend (приостановить) — из состояния active в состояние suspended;
- terminate (завершить) — из состояния active в состояние terminated;
- nullify (аннулировать) — из состояния normal в состояние nullified;
- create (создать) — из начального (пустого) состояния в состояние active;
- create (создать) — из начального (пустого) состояния в состояние pending;
- revise (пересмотреть) — из состояния pending в состояние pending;
- resume (продолжить) — из состояния suspended в состояние active;
- revise (пересмотреть) — из состояния suspended в состояние suspended;
- terminate (завершить) — из состояния suspended в состояние terminated;
- reactivate (активизировать заново) — из состояния terminated в состояние active;
- revise (пересмотреть) — из состояния terminated в состояние terminated.

A.3.3.7 Класс RoleLink (в предметной области Roles)

**Свойства класса RoleLink**

Атрибуты класса RoleLink:

- typeCode:: CS,
- priorityNumber :: INT,
- effectiveTime:: IVL<TS>.

Ассоциации класса RoleLink:

- source::(1..1) Role::outboundLink::(0..*) (ассоциация с классом Role, роль outboundLink — исходящая связь),
- target::(1..1) Role::inboundLink::(0..*) (ассоциация с классом Role, роль inboundLink — входящая связь).

**Определение класса RoleLink**

Связь двух экземпляров класса Role, выражающая зависимость соответствующих ролей и позволяющая выполнять авторизацию или отмену подчиненной роли в зависимости от изменения состояния ведущей роли.

**Примечания к использованию**

Экземпляр класса RoleLink описывает отношения между ролями, а не между людьми (или другими сущностями). Люди (или другие сущности) в основном связаны между собой прямыми отношениями исполнитель/контролер, возникающими у роли исполнителя, а в более общем плане — отношениями взаимодействия (то есть их участием в действиях).

***Примеры***

***1 Роль должности или агента зависит от другой роли, а именно роли работника. Если роль работника завершается, то прекращается и роль должности. Тем самым определяется зависимость роли должности от роли работника, или другими словами — должность «является частью» роли работника.***

***2 Одна из ролей может иметь полномочия управления другой ролью (в иерархии подчинения). Например, работник категории «управляющий» может иметь полномочия управления служащими категории «аналитик», что может быть указано с помощью экземпляра класса RoleLink, у которого атрибут typeCode имеет значение «DIRAUTH» (has direct authority over — имеет прямые полномочия управления).***

**Атрибуты класса RoleLink**

A.3.3.7.1 RoleLink.typeCode:: CS (1..1) Mandatory

Словарный домен: RoleLinkType

**Определение**

Код, указывающий вид связи, представленной данным экземпляром класса RoleLink, например, «PART» (has part — имеет частью), «DIRAUTH» (has direct authority over — имеет прямые полномочия управления).

A.3.3.7.2 RoleLink.priorityNumber:: INT (0..1)

**Определение**

Целое значение, указывающее относительный приоритет данной связи среди других связей похожих типов, у которых источником является один и тот же экземпляр класса Role.

**Примечания к использованию**

Связи с меньшими значениями атрибута priorityNumber рассматриваются раньше и выше тех, что имеют более высокие значения. Приоритеты могут быть полностью упорядоченными (все номера приоритета уникальны) и частично упорядоченными, когда одинаковый приоритет может быть присвоен нескольким связям между ролями.

***Примеры***

***1 Если имеет несколько резервных исполнителей, можно указать их очередность.***

***2 Предпочтительное место оказания медицинской помощи для врача, работающего в конкретной медицинской организации.***

A.3.3.7.3 RoleLink.effectiveTime:: IVL<TS> (0..1)

**Определение**

Интервал времени, указывающий период действия связи между ролями.

**А.3.4 Классы предметной области CoreInfrastructure**

А.3.4.1 Класс ActHeir (в предметной области CoreInfrastructure)

Класс ActHeir является специализацией класса Act.

**Определение класса ActHeir**

Подтип класса Act, используемый исключительно для восполнения недостатка в текущих инструментальных средствах, не обеспечивающих возможность рефлексивного замыкания отношений генерализации (например, «действие является действием»).

**Примечания к использованию**

Хотя класс ActHeir используется для представления экземпляров класса Act, которые не имеют специализаций в модели RIM, он определен исключительно для восполнения (простительного) недостатка в текущих инструментальных средствах и структур данных, используемых в методологии HL7. Он не имеет концептуального значения или семантических последствий при моделировании. Следует обратить внимание, что классы EntityHeir и RoleHeir используются аналогичным образом для классов Act и Role соответственно.

**Обоснование**

Было обнаружено, что в иерархическом описании модели HMD нельзя создать структуру выбора choice для совокупности классов, если все они являются подтипами классов Act, Role или Entity, но при этом для них не определены отдельные физические классы. Другими словами, это классы, которые следовало бы определить в модели RIM в виде прямых потомков (наследников) классов Act, Role или Entity, не имеющих уникальных атрибутов или ассоциаций.

*Добавление такого единственного пустого класса в иерархию позволяет конструировать сообщения, имеющие соответствующую необходимую структуру выбора.* Последующее развитие методологии инструментальных средств может позволить исключить такие классы в пользу эквивалентной абстракции, введенной в методологию.

***Примеры — Рассмотрим уточненную информационную модель RMIM (refined message information model), в которую включен класс Act и его специализации Observation (исследование) и PatientEducationAct (санитарное просвещение). Последний является прямой специализацией («клоном») класса Act. В этом случае класс ActHeir используется как основа клона PatientEducationAct вместо самого класса Act из модели RIM. Класс Act используется здесь только как общая генерализация классов Observation и PatientEducationAct.***

А.3.4.2 Класс EntityHeir (в предметной области CoreInfrastructure)

Класс EntityHeir является специализацией класса Entity.

**Определение класса EntityHeir**

Подтип класса Entity, используемый исключительно для восполнения недостатка в текущих инструментальных средствах, не обеспечивающих возможность рефлексивного замыкания отношений генерализации (например, «сущность является сущностью»).

**Примечания к использованию**

Хотя класс EntityHeir используется для представления экземпляров класса Entity, которые не имеют специализаций в модели RIM, он определен исключительно для восполнения (простительного) недостатка в текущих инструментальных средствах и структур данных, используемых в методологии HL7. Он не имеет концептуального значения или семантических последствий при моделировании. Следует обратить внимание, что классы ActHeir и RoleHeir используются аналогичным образом для классов Act и Role соответственно.

**Обоснование**

Было обнаружено, что в иерархическом описании модели HMD нельзя создать структуру выбора choice для совокупности классов, если все они являются подтипами классов Act, Role или Entity, но при этом для них не определены отдельные физические классы. Другими словами, это классы, которые следовало бы определить в модели RIM в виде прямых потомков (наследников) классов Act, Role или Entity, не имеющих уникальных атрибутов или ассоциаций.

*Добавление такого единственного пустого класса в иерархию позволяет конструировать сообщения, имеющие соответствующую необходимую структуру выбора.* Последующее развитие методологии инструментальных средств может позволить исключить такие классы в пользу эквивалентной абстракции, введенной в методологию.

***Примеры — Пусть уточненная информационная модель RMIM содержит класс Entity и его специализации EnvironmentalEntity (сущность окружающей среды) и LivingSubject (живой субъект). Класс EnvironmentalEntity является прямой специализацией («клоном») класса Entity. В этом случае класс EntityHeir используется как основа клона EnvironmentalEntity вместо самого класса Entity из модели RIM. Класс Entity используется здесь только как общая генерализация классов EnvironmentalEntity и LivingSubject.***

А.3.4.3 Класс InfrastructureRoot (в предметной области CoreInfrastructure)

**Свойства класса InfrastructureRoot**

Атрибуты класса InfrastructureRoot:

- nullFlavor :: CS,

- realmCode :: SET<CS>,
- typeId :: II,
- templateId :: LIST<II>.

Специализации класса InfrastructureRoot:

- Act,
- ActRelationship,
- Participation,
- Entity,
- LanguageCommunication,
- Role,
- RoleLink,
- Acknowledgement,
- AcknowledgementDetail,
- Attachment,
- AttentionLine,
- CommunicationFunction,
- Transmission,
- TransmissionRelationship,
- Parameter,
- QueryEvent,
- SelectionExpression,
- SortControl.

**Определение класса InfrastructureRoot**

Абстрактная генерализация всех классов модели RIM, либо непосредственно, либо через наследование.

**Ограничение использования**

В общем случае объявления ограничений наподобие тех, что могут быть наложены с помощью атрибутов данного класса, могут возникать при создании экземпляров класса модели RIM или одного из его производных клонов в целях осуществления коммуникации по протоколу HL7. Поэтому эти атрибуты должны присутствовать во всех классах модели RIM и их клонах.

**Примечания к использованию**

В классе InfrastructureRoot предусмотрен ряд атрибутов коммуникационной инфраструктуры, которые могут использоваться в экземплярах коммуникаций, определенных комитетом HL7 на основе модели RIM. Будучи заданными в экземпляре коммуникации, эти атрибуты указывают, является ли структура информации ограниченной специально заданными шаблонами, областью применения или общими коммуникационными типами элементов.

**Атрибуты класса InfrastructureRoot**

A.3.4.3.1 InfrastructureRoot.nullFlavor :: CS (0..1)

Словарный домен: NullFlavor

**Определение**

Признак, указывающий, что данный экземпляр класса пуст, и идентифицирующий причину его пустоты.

**Открытый вопрос**

В исходном тексте было сказано, что «остальная часть информации об этом классе и его свойствах не должна передаваться». Если признак пустоты указывает невозможность представления значения (например, равен «PINF»), а не отсутствие значения, правильно ли это? Надо ли ограничивать значения?

A.3.4.3.2 InfrastructureRoot.realmCode:: SET<CS> (0..*)

Словарный домен: Realm

**Определение**

Квалификатор словарного домена, позволяющий специализацию словарного домена кодированных атрибутов в зависимости от географической, организационной или политической среды применения стандарта HL7.

A.3.4.3.3 InfrastructureRoot.typeId :: II (0..1)

**Определение**

Уникальный идентификатор статической структуры, определенной в стандарте HL7 для наложения ограничений на артефакт.

**Примечания к использованию**

Это могут быть ограничения общего типа, известные в коммуникационной среде обмена сообщениями как CMET (Common Message Elements Type — типы общих элементов сообщений), или содержание, включенное в определенную обертку.

A.3.4.3.4 InfrastructureRoot.templateId :: LIST<II> (0..*)

**Определение**

Уникальный идентификатор шаблона, накладывающего ограничения на артефакт.

A.3.4.4 Класс RoleHeir (в предметной области CoreInfrastructure)

Класс RoleHeir является специализацией класса Role.

**Определение класса RoleHeir**

Подтип класса Role, используемый исключительно для восполнения недостатка в текущих инструментальных средствах, не обеспечивающих возможности рефлексивного замыкания отношений генерализации (например, «роль является ролью»).

**Примечания к использованию**

Хотя класс RoleHeir используется для представления экземпляров класса Role, которые не имеют специализаций в модели RIM, он определен исключительно для восполнения (простительного) недостатка в текущих инструментальных средствах и структур данных, используемых в методологии HL7. Он не имеет концептуального значения или семантических последствий при моделировании. Следует обратить внимание, что классы ActHeir и EntityHei используются аналогичным образом для классов Act и Entity соответственно.

**Обоснование**

Было обнаружено, что в иерархическом описании модели HMD нельзя создать структуру выбора choice для совокупности классов, если все они являются подтипами классов Act, Role или Entity, но при этом для них не определены отдельные физические классы. Другими словами, это классы, которые следовало бы определить в модели RIM в виде прямых потомков (наследников) классов Act, Role или Entity, не имеющих уникальных атрибутов или ассоциаций.

*Добавление такого единственного пустого класса в иерархию позволяет конструировать сообщения, имеющие соответствующую необходимую структуру выбора.* Последующее развитие методологии инструментальных средств может позволить исключить такие классы в пользу эквивалентной абстракции, введенной в методологию.

***Примеры — Пусть уточненная информационная модель RMIM содержит класс Role и его специализации Employee (служащий) и Member (член). Класс Member является прямой специализацией («клоном») класса Role. В этом случае класс EntityHeir используется как основа клона Member вместо самого класса Entity из модели RIM. Класс Role используется здесь только как общая генерализация классов Employee и Member.***

**A.3.5 Классы предметной области MessageControl**

A.3.5.1 Класс Acknowledgement (в предметной области MessageControl)

**Свойства класса Acknowledgement**

Атрибуты класса Acknowledgement:

- typeCode :: CS,
- expectedSequenceNumber :: INT,
- messageWaitingNumber :: INT,
- messageWaitingPriorityCode :: CE.

Ассоциации класса Acknowledgement:

- acknowledgementDetail::(0..*) AcknowledgementDetail::acknowledgement::(1..1) (ассоциация с классом AcknowledgementDetail, роль acknowledgement — подтверждение),
- acknowledges::(1..1) Transmission::acknowledgedBy::(0..*) (ассоциация с классом Transmission, роль acknowledgedBy — подтверждено приложением),
- conveyingTransmission::(1..1) Transmission::conveyedAcknowledgement::(0..*) (ассоциация с классом Transmission, роль conveyedAcknowledgement — переданное подтверждение).

Класс Acknowledgement является специализацией класса InfrastructureRoot.

**Определение класса Acknowledgement**

Метаданные, необходимые при подтверждении другого сообщения.

**Атрибуты класса Acknowledgement**

A.3.5.1.1 Acknowledgement.typeCode :: CS (0..1)

Словарный домен: AcknowledgementType

**Определение**

Код типа подтверждения, определенный в перечислимом множестве значений.

***Примеры — Приложение-получатель успешно обработало сообщение; приложение-получатель нашло в сообщении одну или несколько ошибок.***

A.3.5.1.2 Acknowledgement.expectedSequenceNumber :: INT (0..1)

**Определение**

Порядковый номер сообщений в последовательности сообщений.

A.3.5.1.3 Acknowledgement.messageWaitingNumber :: INT (0..1)

**Определение**

Число сообщений, которое подтверждающее приложение ожидает в очереди сообщений, предназначенных приложению-получателю.

**Примечания к использованию**

Эти сообщения должны быть впоследствии получены с помощью запроса. Этот режим рассчитан на приложения-получатели, которые не могут получить прямое сообщение (то есть занимаются регулярным опросом).

***Пример — Если в очереди находятся 3 низкоприоритетных сообщения, 1 сообщение со средним приоритетом и 1 сообщение с высоким приоритетом, то число ожидаемых сообщений равно 5, то есть общему числу сообщений.***

A.3.5.1.4 Acknowledgement.messageWaitingPriorityCode :: CE (0..1)

Словарный домен: MessageWaitingPriority

**Определение**

Наивысший уровень важности в совокупности сообщений, которые подтверждающее приложение ожидает в очереди сообщений, предназначенных приложению-получателю.

**Примечания к использованию**

Эти сообщения должны быть впоследствии получены с помощью запроса. Этот режим рассчитан на приложения-получатели, которые не могут получить прямое сообщение (то есть занимаются регулярным опросом). Значение атрибута messageWaitingPriorityCode указывает, насколько важным является самое важное сообщение, и может влиять на частоту опроса очереди приложением-получателем. По местным соглашениям код важности может использоваться для определения интервала времени, в течение которого, как ожидается, приложение-получатель извлечет сообщения из очереди.

A.3.5.2 Класс AcknowledgementDetail (в предметной области MessageControl)

**Свойства класса AcknowledgementDetail**

Атрибуты класса AcknowledgementDetail:

- typeCode :: CS,
- code :: CE,
- text :: ED,
- location :: SET<ST>.

Ассоциации класса AcknowledgementDetail:

- acknowledgement::(1..1) Acknowledgement::acknowledgementDetail::(0..*) (ассоциация с классом Acknowledgement, роль acknowledgementDetail — детали подтверждения).

Класс AcknowledgementDetail является специализацией класса InfrastructureRoot.

**Определение класса AcknowledgementDetail**

Сообщение, предоставляющее информацию о коммуникации, разборе или форматно-логическом контроле (не учитывающем деловые правила) подтверждаемого сообщения.

**Атрибуты класса AcknowledgementDetail**

A.3.5.2.1 AcknowledgementDetail.typeCode :: CS (0..1)

Словарный домен: AcknowledgementDetailType

**Определение**

Вид информации, возвращаемой в подтверждающем сообщении.

***Примеры — Ошибка, предупреждение, информация.***

A.3.5.2.2 AcknowledgementDetail.code :: CE (0..1)

Словарный домен: AcknowledgementDetailCode

**Определение**

Тип подтверждения, который берется из перечислимого множества значений.

**Примечания к конструированию**

Поскольку в исходные примеры были включены специальные атрибуты и даты, то они описывают текст, а не код. Должны быть включены новые примеры со значениями, взятыми из словарного домена.

***Примеры — Обязательный атрибут отсутствует; неподдерживаемое взаимодействие; недопустимая система кодирования в значении типа CNE.***

A.3.5.2.3 AcknowledgementDetail.text :: ED (0..1)

**Определение**

Дополнительная диагностическая информация, релевантная возвращаемой информации.

**Примечания к использованию**

Диагностической информацией может быть свободный текст или структурированные данные (например, XML).

***Примеры — Исключение в среде Java, дамп памяти, внутренний код ошибки, информация стека вызовов.***

A.3.5.2.4 AcknowledgementDetail.location :: SET<ST> (0..*)

**Определение**

Позиция в подтверждаемом сообщении, к которой относится возвращаемая информация.

**Примечания к использованию**

Не с любой возвращаемой информацией можно связать соответствующую позицию. Открытый вопрос: необходимо идентифицировать специфичный формат строки, описывающей позицию в сообщении. Это может быть xPath или OCL.

***Пример — Позиция отсутствующего обязательного атрибута; позиция неправильного кода в значении типа CNE; позиция не имеет значения в неподдерживаемом взаимодействии.***

A.3.5.3 Класс Attachment (в предметной области MessageControl)

**Свойства класса Attachment**

Атрибуты класса Attachment:

- id :: II,
- text :: ED.

Ассоциации класса Attachment:

- transmission::(1..1) Transmission::attentionLine::(0..*) (ассоциация с классом Transmission, роль attachment — вложение).

Класс Attachment является специализацией класса InfrastructureRoot.

**Определение класса Attachment**

Адресуемый блок данных, на который может быть дана ссылка изнутри сообщения.

**Примечания к использованию**

Ссылки из тела сообщения на вложения осуществляются с помощью функциональности передачи по ссылке, предусмотренной в значениях с типом данных ED.

**Примечания к конструированию**

Открытый вопрос требует большей детализации.

**Открытый вопрос**

Для правильного использования этого класса (Attachment) требуется расширить механизм ссылок, предусмотренный в типе данных ED.

**Атрибуты класса Attachment**

A.3.5.3.1 Attachment.id :: II (0..1)

Словарный домен: AttentionKeyword

**Определение**

Идентификатор вложения, на которое есть ссылка из атрибута с типом данных ED, содержащегося в каком-либо месте сообщения.

A.3.5.3.2 Attachment.text :: ED (1..1)

**Определение**

Блок данных, образующих вложение.

A.3.5.4 Класс AttentionLine (в предметной области MessageControl)

**Свойства класса AttentionLine**

Атрибуты класса AttentionLine:

- keyWordText :: SC,
- value :: ANY.

Ассоциации класса AttentionLine:

- transmission::(1..1) Transmission::attentionLine::(0..*) (ассоциация с классом Transmission, роль attentionLine — идентификация адресата).

Класс AttachmentLine является специализацией класса InfrastructureRoot.

**Определение класса AttentionLine**

Совокупность параметров транспорта, которые могут быть доступны из транспортной обертки.

**Ограничение использования**

Содержание этого класса должно быть связано с транспортом в целом и должно использоваться исключительно для целей, связанных с транспортом сообщений, и никаким образом не должно влиять на семантическую интерпретацию содержания транспорта.

**Примечания к использованию**

Класс AttentionLine представляет собой пару «имя — значение», в которой атрибут keyWordText задает тему, взятую из перечислимого набора данных, а атрибут value содержит значение параметра.

**Примечания к конструированию**

Подтвердить редакцию текста. Сделать определение более понятным, привести примеры.

**Атрибуты класса AttentionLine**

A.3.5.4.1 AttentionLine.keyWordText :: SC (0..1)

Словарный домен: AttentionKeyword

**Определение**

Категория параметра attentionLine.

***Примеры — Идентификатор пациента, тип события общественного здоровья.***

A.3.5.4.2 AttentionLine.value :: ANY (0..1)

**Определение**

Значение, ассоциированное с ключом, переданным в атрибуте AttentionLine.keyWordText.

**Примечания к использованию**

Значение предназначено для машинной обработки.

**Формальное ограничение**

Тип данных значения этого атрибута должен быть одним из следующих: BL, CV, II, URL, INT, REAL, TS, PQ, MO, IVL<PQ>, IVL<TS>.

A.3.5.5 Класс Batch (в предметной области MessageControl)

**Свойства класса Batch**

Атрибуты класса Batch:

- referenceControlId :: II,
- name :: SC,
- batchComment :: SET<ST>,
- transmissionQuantity :: INT,
- batchTotalNumber :: SET<INT>,
- contentProcessingModeCode :: CE.

Ассоциации класса Batch:

- transmission::(0..*) Transmission::batch::(0..1) (ассоциация с классом Transmission, роль batch — пакет).

Класс Batch является специализацией класса Transmission.

**Определение класса Batch**

Сообщение, являющееся коллекцией сообщений стандарта HL7 V3.

**Примечания к конструированию**

Должен ли экземпляр класса Batch иметь какой-либо эффект на входящее в него сообщение или же класс Batch представляет собой композицию входящих в него сообщений?

**Атрибуты класса Batch**

A.3.5.5.1 Batch.referenceControlId :: II (0..1)

**Определение**

Управляющий идентификатор пакета, присвоенный ему при первоначальной передаче.

A.3.5.5.2 Batch.name :: SC (0..1)

Словарный домен: BatchName

**Определение**

Идентификатор пакета.

**Примечания к использованию**

Этот атрибут используется приложением, обрабатывающим пакет.

A.3.5.5.3 Batch.batchComment :: SET<ST> (0..*)

**Определение**

Комментарии к содержанию пакета.

A.3.5.5.4 Batch.transmissionQuantity :: INT (0..1)

**Определение**

Общее число отдельных сообщений, содержащихся в пакете, включая вложенные пакеты.

A.3.5.5.5 Batch.batchTotalNumber :: SET<INT> (0..*)

**Определение**

Общее число сообщений в пакете.

**Примечания к использованию**

В случае вложенных пакетов значение атрибута batchTotalNumber относится к текущему пакету, в то время как значение атрибута transmissionQuantity суммирует число всех вложенных сообщений.

**Примечания к конструированию**

Следует подтвердить отличие от атрибута transmissionQuantity.

A.3.5.5.6 Batch.contentProcessingModeCode :: CE (0..1)

Словарный домен: ContentProcessingMode

**Определение**

Тип обработки содержания, которую получатель пакета ожидает предпринять.

**Примечания к использованию**

По умолчанию предполагается последовательная обработка.

***Примеры — Последовательная, неупорядоченная.***

A.3.5.6 Класс CommunicationFunction (в предметной области MessageControl)

**Свойства класса CommunicationFunction**

Атрибуты класса CommunicationFunction:

- typeCode :: CS,

- telecom :: TEL.

Ассоциации класса CommunicationFunction:

- entity::(1..*) Entity::communicationFunction::(0..*) (ассоциация с классом Entity, роль communicationFunction — коммуникационная функция),

- transmission::(1..*) Transmission::communicationFunction::(0..*) (ассоциация с классом Transmission, роль communicationFunction — коммуникационная функция).

Класс CommunicationFunction является специализацией класса InfrastructureRoot.

**Определение класса CommunicationFunction**

Класс отношения, связывающий с передачей сообщений различные задействованные в ней сущности (отправитель, получатель, реагирующее приложение).

**Атрибуты класса CommunicationFunction**

A.3.5.6.1 CommunicationFunction.typeCode :: CS (0..1)

Словарный домен: CommunicationFunctionType

**Определение**

Тип коммуникационной функции, выполняемой сущностью по отношению к передаче данных.

***Примеры — Отправитель, получатель, реагирующее приложение.***

A.3.5.6.2 CommunicationFunction.telecom :: TEL (0..1)

**Определение**

Телекоммуникационный адрес, который может использоваться для контакта с сущностью, выполняющей данную функцию.

A.3.5.7 Класс Message (в предметной области MessageControl)

**Свойства класса Message**

Атрибуты класса Message:

- processingCode :: CS,
- processingModeCode :: CS,
- acceptAckCode :: CS,
- responseCode :: CS,
- sequenceNumber :: INT,
- attachmentText :: SET<ED>.

Ассоциации класса Message:

- controlAct::(0..1) ControlAct::payload::(0..*) (ассоциация с классом ControlAct, роль payload — полезная нагрузка).

Класс Message является специализацией класса Transmission.

**Определение класса Message**

Родительский класс для всех сообщений стандарта HL7 Версии 3.

**Примечания к конструированию**

Этот класс может быть родительским, но это не то, что делает его сообщением. Каков критерий того, что нечто должно моделироваться как сообщение?

**Атрибуты класса Message**

A.3.5.7.1 Message.processingCode :: CS (0..1)

Словарный домен: ProcessingID

**Определение**

Цель передачи сообщения по отношению к состоянию системы-отправителя.

***Примеры — Производственная обработка, обучение, отладка.***

A.3.5.7.2 Message.processingModeCode :: CS (0..1)

Словарный домен: ProcessingMode

**Определение**

Режим обработки сообщения.

***Примеры — Текущая обработка, режим архивирования, режим начальной загрузки, режим восстановления из архива.***

A.3.5.7.3 Message.acceptAckCode :: CS (0..1)

Словарный домен: AcknowledgementCondition

**Определение**

Условия, при которых в ответ на данное сообщение должно быть возвращено подтверждение «приема».

A.3.5.7.4 Message.responseCode :: CS (0..1)

Словарный домен: ResponseLevel

**Определение**

Указывает, ожидается ли от адресата данного взаимодействия информация о прикладной обработке и насколько детальной она должна быть.

**Примечания к использованию**

Этот атрибут ограничивает варианты ответа получателя.

***Пример — Если в некотором взаимодействии получатель отвечает за посылку либо подтверждения приемлемости, либо отказа, а атрибут responseCode имеет значение «E» (exception — исключение), то получатель должен ответить на данное сообщение только в том случае, если он откажет в его обработке.***

A.3.5.7.5 Message.sequenceNumber :: INT (0..1)

**Определение**

Порядковый номер данного сообщения в последовательности сообщений.

**Примечания к использованию**

Этот атрибут предназначен для реализации протокола последовательной нумерации. Его значение увеличивается на 1 при каждом следующем присваивании.

A.3.5.7.6 Message.attachmentText :: SET<ED> (0..*)

**Примечания к использованию**

В любой спецификации технологии реализации рекомендуется выносить вложения за пределы основного тела сообщения. Для ссылок на вложения из тела сообщения используется ссылочная функциональность, предусмотренная в типе данных ED.

A.3.5.8 Класс Transmission (в предметной области MessageControl)

**Свойства класса Transmission**

Атрибуты класса Transmission:

- id :: II,
- creationTime :: TS,
- securityText :: ST,
- responseModeCode :: CS,
- versionCode :: CS,
- interactionId :: II,
- profileId :: LIST<II>.

Ассоциации класса Transmission:

- acknowledgedBy::(0..*) Acknowledgement::acknowledges::(1..1) (ассоциация с классом Acknowledgement, роль acknowledges — подтверждение),
- conveyedAcknowledgement::(0..*) Acknowledgement::conveyingTransmission::(1..1) (ассоциация с классом Acknowledgement, роль conveyingTransmission — передача данных),
- attentionLine::(0..*) AttentionLine::transmission::(1..1) (ассоциация с классом AttentionLine, роль transmission — передача),
- batch::(0..1) Batch::transmission::(0..*) (ассоциация с классом Batch, роль transmission — передача),
- communicationFunction::(0..*) CommunicationFunction::transmission::(1..*) (ассоциация с классом CommunicationFunction, роль transmission — передача);
- outboundRelationship :: (0..*) TransmissionRelationShip :: source :: (1..1) (ассоциация с классом TransmissionRelationShip, роль source — источник),
- inboundRelationship :: (0..*) TransmissionRelationShip :: target :: (1..1) (ассоциация с классом TransmissionRelationShip, роль target — цель).

Класс Transmission является специализацией класса InfrastructureRoot.

Класс Transmission является обобщением следующих классов:

- Batch;
- Message.

**Определение класса Transmission**

Представляет информацию о конкретном акте передачи информации от одного приложения к другому.

**Атрибуты класса Transmission**

A.3.5.8.1 Transmission.id :: II (0..1)

**Определение**

Уникальный идентификатор акта передачи информации.

A.3.5.8.2 Transmission.creationTime :: TS (0..1)

**Определение**

Дата и время создания акта передачи информации ее отправителем.

**Примечания к использованию**

Если указан часовой пояс, то по умолчанию он будет использоваться для всех передаваемых данных.

A.3.5.8.3 Transmission.securityText :: ST (0..1)

**Определение**

Этот атрибут используется приложениями для реализации функций безопасности акта передачи информации. В настоящее время его более детальное описание не предусмотрено.

A.3.5.8.4 Transmission.responseModeCode :: CS (0..1)

Словарный домен: ResponseMode

**Определение**

Режим акта передачи, в котором получатель должен проявить свою ответственность.

***Примеры — Получатель может не дать ответ немедленно; получатель должен дать немедленный ответ; получатель может сохранять все прикладные ответы в очереди на все время ее опроса.***

A.3.5.8.5 Transmission.versionCode :: CS (0..1)

Словарный домен: HL7StandardVersionCode

**Определение**

Уникальный идентификатор версии акта передачи, который может быть модифицирован и послан заново.

**Примечания к конструированию**

Определение переписано. Поскольку класс Message теперь не содержит этот атрибут, то удалено первоначальное примечание «Этот атрибут присутствует также в дочернем классе Message. Это изменение было сделано вместо того, чтобы переместить атрибут в общий родительский класс Transmission. Такое решение было принято, поскольку не все вопросы методологии и обратной совместимости были проработаны. Как только сложится определенность с обратной совместимостью, этот атрибут должен быть перемещен в родительский класс. Проблема состоит в последовательной нумерации атрибутов в среде разработки HL7 (HL7 Development Framework, HDF) и в ее влиянии на спецификации реализуемых технологий (Implementable Technology Specification, ITS)».

A.3.5.8.6 Transmission.interactionId :: II (0..1)

**Определение**

Идентификатор взаимодействия, определенный в стандарте HL7 V3, ограничивающий данный акт передачи.

**Примечания к конструированию**

Определение переписано. Поскольку класс Batch теперь не содержит этот атрибут, то удалено первоначальное примечание «Этот атрибут присутствует также в дочернем классе Batch. Это изменение было сделано вместо того, чтобы переместить атрибут в общий родительский класс Transmission. Такое решение было принято, поскольку не все вопросы методологии и обратной совместимости были проработаны. Как только сложится определенность с обратной совместимостью, этот атрибут должен быть перемещен в родительский класс. Проблема состоит в последовательной нумерации атрибутов в среде разработки HL7 (HL7 Development Framework, HDF) и в ее влиянии на спецификации реализуемых технологий (Implementable Technology Specification, ITS)».

A.3.5.8.7 Transmission.profileId :: LIST<II> (0..*)

**Определение**

Идентификатор профиля(ей), ограничивающего(их) акт передачи.

**Ограничение использования**

Если задано несколько профилей, то экземпляр акта передачи ДОЛЖЕН всем им удовлетворять. Однако получатель МОЖЕТ проверить соответствие только первому профилю, который он распознает. Поэтому «предпочтительный» или более строгий профиль должен быть указан первым.

**Примечания к использованию**

Профиль акта передачи позволяет данной реализации явно указать ее отличие от стандартного определения взаимодействия.

**Примечания к конструированию**

Фраза «ее отличие от стандартного определения акта передачи» заменена на «ее отличие от стандартного определения взаимодействия».

A.3.5.9 Класс TransmissionRelationship (в предметной области MessageControl)

**Свойства класса TransmissionRelationship**

Атрибуты класса TransmissionRelationship:

- typeCode :: CS.

Ассоциации класса TransmissionRelationship:

- source :: (1..1) Transmission :: outboundRelationship :: (0..*) (ассоциация с классом Transmission, роль outboundRelationship — исходящее отношение),
- target :: (1..1) Transmission .: inboundRelationship :: (0..*) (ассоциация с классом Transmission, роль inboundRelationship — входящее отношение).

**Определение класса TransmissionRelationship**

Направленная ассоциация от экземпляра класса Transmission (источника) к целевому экземпляру класса Transmission.

**Примечания к использованию**

Экземпляры класса TransmissionRelationship, связанные с одним и тем же экземпляром класса Transmission (источником), называются «исходящими» отношениями. Экземпляры класса TransmissionRelationship, связанные с одним и тем же целевым экземпляром класса Transmission (источником), называются «входящими» отношениями.

Смысл и назначение отношения, описанного экземпляром класса TransmissionRelationship, задаются в атрибуте TransmissionRelationship.typeCode.

Начальная реализация выявила только одно значение для этого атрибута.

**Атрибуты класса TransmissionRelationship**

A.3.5.9.1 TransmissionRelationship.typeCode :: CS (1..1) Обязательный

Словарный домен: TransmissionRelationshipType.

**Определение**

Назначение экземпляра класса TransmissionRelationship.

**Ограничение использования**

Каждое значение предусматривает специфичные ограничения на связывание экземпляров класса Transmission.

***Пример — «Отношение, указывающее, что акт передачи (источник) является продолжением (SEQL) целевого акта передачи».***

**A.3.6 Классы предметной области QueryControl**

A.3.6.1 Класс LogicalExpression (в предметной области QueryControl)

**Свойства класса LogicalExpression**

Атрибуты класса LogicalExpression:

- relationalConjunctionCode :: CS.

Ассоциации класса LogicalExpression:

- userAsLeft::(0..*) SelectionExpression::leftSide::(0..1) (ассоциация с классом SelectionExpression, роль leftSide — левая часть),
- userAsRight::(0..*) SelectionExpression::rightSide::(0..1) (ассоциация с классом SelectionExpression, роль rightSide — правая часть).

Класс LogicalExpression является специализацией класса SelectionExpression.

**Атрибуты класса LogicalExpression**

A.3.6.1.1 LogicalExpression.relationalConjunctionCode :: CS (0..1)

Словарный домен: SQLConjunction

**Определение**

Два выражения поиска, соединенные кодом реляционного отношения.

**Примечания к использованию**

Выражение поиска принимает значение true (истина) или false (ложь) в зависимости от истинности участвующих в нем выражений и заданной реляционной операции их соединения.

**Примечания к конструированию**

Черновое определение.

A.3.6.2 Класс Parameter (абстрактный) (в предметной области QueryControl)

**Свойства класса Parameter**

Атрибуты класса Parameter:

- id :: II.

Ассоциации класса Parameter:

- queryByParameter::(0..1) QueryByParameter::parameter::(0..*) (ассоциация с классом QueryByParameter, роль parameter — параметр),
- parameterList::(0..1) ParameterList::parameter::(0..*) (ассоциация с классом ParameterList, роль parameter — параметр).

Класс Parameter является обобщением следующих классов:

- ParameterItem;
- ParameterList.

**Определение класса Parameter**

Уникально идентифицируемое значение или совокупность значений, используемая как критерий запроса.

**Комментарии к конструированию**

Подтвердить «значение» как синоним «параметра».

**Атрибуты класса Parameter**

A.3.6.2.1 Parameter.id :: II (0..1)

Уникальный идентификатор параметра.

A.3.6.3 Класс ParameterItem (в предметной области QueryControl)

**Свойства класса ParameterItem**

Атрибуты класса ParameterItem:

- value :: ANY,
- semanticsText :: ST.

Класс ParameterItem является специализацией класса Parameter.

**Определение класса ParameterItem**

Структура именованного значения (пара «имя — значение»), используемая для элемента, указываемого в запросе.

**Атрибуты класса ParameterItem**

A.3.6.3.1 ParameterItem.value :: ANY (0..1)

Словарный домен: QueryParameterValue.

**Определение**

Значение элемента, указанное в ответе на запрос.

**Примечания к использованию**

Это компонент «значение» пары «имя — значение».

A.3.6.3.2 ParameterItem.semanticsText :: ST (0..1)

**Определение**

Имя элемента в заданной структуре ответа на запрос.

**Примечания к использованию**

Это компонент «имя» пары «имя — значение».

A.3.6.4 Класс ParameterList (в предметной области QueryControl)

**Свойства класса ParameterList**

Ассоциации класса ParameterList:

- parameter::(0..*) Parameter::parameterList::(0..1) (ассоциация с классом Parameter, роль parameterList — список параметров).

Класс ParameterList является специализацией класса Parameter.

**Определение класса ParameterList**

Именованный список параметров.

A.3.6.5 Класс QueryAck (в предметной области QueryControl)

**Свойства класса QueryAck**

Атрибуты класса QueryAck:

- queryResponseCode :: CS,
- resultTotalQuantity :: INT,
- resultCurrentQuantity :: INT,
- resultRemainingQuantity :: INT,
- continuationToken :: ST.

Класс QueryAck является специализацией класса QueryEvent.

**Определение класса QueryAck**

Ответ на запрос.

**Атрибуты класса QueryAck**

A.3.6.5.1 QueryAck.queryResponseCode :: CS (0..1)

Словарный домен: QueryResponse

**Определение**

Статус результата запроса.

A.3.6.5.2 QueryAck.resultTotalQuantity :: INT (0..1)

**Определение**

Общее число совпадений с условием запроса.

A.3.6.5.3 QueryAck.resultCurrentQuantity:: INT (0..1)

**Определение**

Число совпадений с условием запроса, информация о которых передается в текущей порции ответа.

A.3.6.5.4 QueryAck.resultRemainingQuantity:: INT (0..1)

**Определение**

Число оставшихся совпадений с условием запроса, информация о которых все еще должна быть передана получателю.

A.3.6.5.5 QueryAck.continuationToken:: ST (0..1)

**Определение**

Статус продолжения сервера запросов.

**Примечания к использованию**

Структура значения continuationToken определяется сервером запросов.

**Примечания к конструированию**

Не предназначен для документа Query Infrasructure.

**Формальное ограничение**

Если значение этого атрибута присваивает сервер запросов, то запрашивающая система ДОЛЖНА заполнять атрибут QueryAck.continuationToken этим значением в любом следующем требовании продолжения или отмены запроса.

A.3.6.6 Класс QueryByParameter (в предметной области QueryControl)

Ассоциации класса QueryByParameter:

- parameter::(0..*) Parameter::queryByParameter::(0..1) (ассоциация с классом Parameter, роль queryByParameter — запрос с параметрами).

Класс QueryByParameter является специализацией класса QuerySpec.

**Определение класса QueryByParameter**

Определение запроса с параметрами в формате, предложенном комитетом HL7 для замены транзакции QRD/QRF.

**Примечания к использованию**

Этот формат считается замкнутым, поскольку спецификация сервера данных содержит фиксированный список параметров, опубликованный в объявлении соответствия запроса.

**Примечания к конструированию**

Определения запросов не проводят четкую грань между запросом с параметрами и селективным запросом, и документ Query Infrasructure не описывает класс селективного запроса QueryBySelection. Эти определения должны быть или разъяснены, или запрещены. Данный класс является абстрактным и не имеет собственных атрибутов (кроме унаследованных от генерализаций и специализаций).

A.3.6.7 Класс QueryBySelection (в предметной области QueryControl)

Ассоциации класса QueryBySelection:

- selectionExpression::(0..*) SelectionExpression::queryBySelection::(1..1) (ассоциация с классом SelectionExpression, роль queryBySelection — селективный запрос).

Класс QueryBySelection является специализацией класса QuerySpec.

**Определение класса QueryBySelection**

Один из вариантов запросов, определенных в стандарте HL7. В его условии можно указывать любое число переменных из числа тех, что предусмотрены сервером данных, а также любые допустимые операторы и значения для каждой переменной, отвечающие опубликованному объявлению о соответствии запроса.

**Примечания к использованию**

Этот формат запроса является открытым, поскольку в нем допускается спецификация селекции, основанная на опубликованной схеме базы данных.

**Примечания к конструированию**

Этот селективный запрос не описан в документе Query Infrasructure. Нет основы для оценки.

A.3.6.8 Класс QueryContinuation (в предметной области QueryControl)

**Свойства класса QueryContinuation**

Атрибуты класса QueryContinuation:

- startResultNumber :: INT,
- continuationQuantity :: INT.

Класс QueryContinuation является специализацией класса QueryEvent.

**Определение класса QueryContinuation**

Запрос дальнейших данных.

**Примечания к использованию**

В экземпляре этого класса передается информация о состоянии обмена, требуемая на прикладном уровне для обеспечения логического продолжения ответа на запрос.

**Атрибуты класса QueryContinuation**

A.3.6.8.1 QueryContinuation.startResultNumber :: INT (0..1)

**Определение**

Номер экземпляра в серии результатов исходного запроса, с которого начнется следующее сообщение ответа на запрос.

A.3.6.8.2 QueryContinuation.continuationQuantity :: INT (0..1)

**Определение**

Число совпадений с экземплярами, которые будут возвращены в следующем сообщении ответа на запрос.

A.3.6.8.3 QueryContinuation.continuationToken :: ST (0..1)

**Определение**

Значение атрибута continuationToken задается реагирующей системой.

**Примечания к использованию**

Запрашивающая система передает атрибут continuationToken реагирующей системе, чтобы та могла продолжить выполнение требуемого запроса.

**Формальное ограничение**

Если значение этого атрибута присваивает сервер запросов, то запрашивающая система ДОЛЖНА заполнять атрибут QueryAck.continuationToken этим значением в любом следующем требовании продолжения или отмены запроса.

A.3.6.9 Класс QueryEvent (в предметной области QueryControl)

**Свойства класса QueryEvent**

Атрибуты класса QueryEvent:

- queryId :: II,
- statusCode :: CS.

Ассоциации класса QueryEvent:

- controlAct::(1..1) ControlAct::queryEvent::(0..1) (ассоциация с классом ControlAct, роль queryEvent — событие запроса).

Класс QueryEvent является специализацией класса InfrastructureRoot.

Класс QueryEvent имеет следующие специализации:

- QueryAck,
- QueryContinuation,
- QuerySpec.

**Определение класса QueryEvent**

Абстрактный класс, обобщающий все взаимодействия сообщений запроса.

**Примечания к использованию**

Описание того, как сконструировано применение запросов в модели RIM, см. в группе предметных областей Specification Infrastructure, документ Query Infrastructure, глава Messaging.

**Примечания к конструированию**

Назначение подраздела обоснования не было понятным; удален.

Использованы определения, взятые из документа Query Infrastructure; следует синхронизировать изменения.

**Атрибуты класса QueryEvent**

A.3.6.9.1 QueryEvent.queryId :: II (0..1)

**Определение**

Уникальный идентификатор запроса.

**Примечания к использованию**

Значение этого атрибута предназначено для привязки ответных сообщений к исходному запросу. Идентификатор QueryEvent.queryId может оставаться одним и тем же в серии нескольких обменов сообщениями, осуществляемых при продолжении предшествующего им запроса.

A.3.6.9.2 QueryEvent.statusCode :: CS (0..1)

Словарный домен: QueryStatusCode

A.3.6.9.3 Переходы состояний класса QueryEvent (атрибутом состояния является statusCode)

Диаграмма перехода состояний класса QueryEvent приведена на рисунке А.14.

Состояния класса QueryEvent:

aborted (прекращен),

deliveredResponse (ответ доставлен),

executing (на выполнении),

new (новый),

waitContinuedQueryResponse (ожидает продолжения ответа на запрос).

Переходы состояний класса QueryEvent:

abort (прекратить) — (из состояния deliveredResponse в состояние aborted);

activateQueryContinue (активировать продолжение запроса) — (из состояния deliveredResponse в состояние executing);

abort (прекратить) — (из состояния executing в состояние aborted);

completeInitialQueryResponse (завершить ответ на исходный запрос) — (из состояния executing в состояние deliveredResponse);

completeQueryContinuation (завершить продолжение запроса) — (из состояния executing в состояние deliveredResponse);

executeQuerySpec (выполнить спецификацию запроса) — (из состояния new в состояние executing);

create (создать) — (из состояния null в состояние new).

A.3.6.10 Класс QuerySpec (в предметной области QueryControl)

**Свойства класса QuerySpec**

Атрибуты класса QuerySpec:

- modifyCode :: CS,
- responseElementGroupId :: SET<II>,
- responseModalityCode :: CS,
- responsePriorityCode :: CS,
- initialQuantity :: INT,
- initialQuantityCode :: CE,
- executionAndDeliveryTime :: TS.

Ассоциации класса QuerySpec:

- sortControl::(0..*) SortControl::querySpec::(1..1) (ассоциация с классом SortControl, роль querySpec — спецификация запроса).

Класс QuerySpec является детализацией класса QueryEvent.

Класс QuerySpec имеет следующие специализации:

- QueryByParameter;
- QuerySelection.

**Определение класса QuerySpec**

Критерии и ожидания ответа, применяемые к запросу.

**Атрибуты класса QuerySpec**

A.3.6.10.1 QuerySpec.modifyCode :: CS (0..1)

Словарный домен: ModifyIndicator

**Определение**

Указывает, является ли подписка на результаты запроса новой или модифицированной.

A.3.6.10.2 QuerySpec.responseElementGroupId :: SET<II> (0..*)

**Документация**

**Формальное ограничение**

Этот тип сообщения должен выбираться из числа типов, которые получатель обязан воспринимать в качестве ответов на запрос.

A.3.6.10.3 QuerySpec.responseModalityCode :: CS (0..1)

Словарный домен: ResponseModality

**Определение**

Оперативность и группировка экземпляров ответных сообщений.

***Примеры — Пакетный, в реальном времени, дискретный.***

A.3.6.10.4 QuerySpec.responsePriorityCode :: CS (0..1)

Словарный домен: QueryPriority

**Определение**

Временные рамки, в течение которых запрашивающее приложение ожидает ответа на запрос.

***Примеры — Немедленный, отложенный.***

A.3.6.10.5 QuerySpec.initialQuantity :: INT (0..1)

**Определение**

Максимальный размер ответа, который может быть принят запрашивающим приложением.

A.3.6.10.6 QuerySpec.initialQuantityCode :: CE (0..1)

Словарный домен: QueryRequestLimit

**Определение**

Единицы измерения максимальной длины ответа на запрос, переданной в атрибуте initialQuantity.

***Пример — (100) записей.***

A.3.6.10.7 QuerySpec.executionAndDeliveryTime :: TS (0..1)

**Определение**

Время, в течение которого должен быть возвращен ответ.

A.3.6.11 Класс RelationalExpression (в предметной области QueryControl)

**Свойства класса RelationalExpression**

Атрибуты класса RelationalExpression:

- elementName :: SC,
- relationalOperatorCode :: CS,
- value :: ST.

Класс RelationalExpression является специализацией класса SelectionExpression.

**Атрибуты класса RelationalExpression**

A.3.6.11.1 RelationalExpression.elementName :: SC (0..1)

Словарный домен: RelationalName

**Определение**

Имя элемента, участвующего в выборке.

**Примечания к конструированию**

Операция выборки не описана в документе Query Infrastructure.

A.3.6.11.2 RelationalExpression.relationalOperatorCode :: CS (0..1)

Словарный домен: RelationalOperator

**Определение**

Код, представляющий тип оператора сравнения значения элемента со значением, передаваемым в атрибуте value.

**Примечания к конструированию**

Операция выборки не описана в документе Query Infrastructure.

***Примеры — Равно, больше, меньше, не равно.***

A.3.6.11.3 RelationalExpression.value :: ST (0..1)

**Определение**

Значение, с которым сравнивается значение элемента в реляционном выражении.

**Примечания к конструированию**

Операция выборки не описана в документе Query Infrastructure.

A.3.6.12 Класс SelectionExpression (абстрактный) (в предметной области QueryControl)

Ассоциации класса SelectionExpression:

- leftSide::(0..1) LogicalExpression::userAsLeft::(0..*) (ассоциация с классом LogicalExpression, роль userAsLeft — используется как левая часть),
- queryBySelection::(1..1) QueryBySelection::selectionExpression::(0..*) (ассоциация с классом QueryBySelection, роль selectionExpression — выражение выборки),
- rightSide::(0..1) LogicalExpression::userAsRight::(0..*) (ассоциация с классом LogicalExpression, роль userAsRight — используется как правая часть).

Класс SelectionExpression имеет следующие специализации:

- LogicalExpression;
- RelationalExpression.

A.3.6.13 Класс SortControl (в предметной области QueryControl)

**Свойства класса SortControl**

Атрибуты класса SortControl:

- sequenceNumber :: INT,
- elementName :: SC,
- directionCode :: CS.

Ассоциации класса SortControl:

- querySpec::(1..1) QuerySpec::sortControl::(0..*) (ассоциация с классом QuerySpec, роль sortControl — управление сортировкой).

**Определение класса SortControl**

**Свойства класса SortControl**

Атрибуты класса SortControl:

- sequenceNumber :: INT,
- elementName :: SC,
- directionCode :: CS.

Ассоциации класса SortControl:

- querySpec::(1..1) QuerySpec::sortControl::(0..*) (ассоциация с классом QuerySpec, роль sortControl — управление сортировкой).

**Определение класса SortControl**

Спецификация порядка сортировки экземпляров, удовлетворяющих условию запроса.

**Атрибуты класса SortControl**

A.3.6.13.1 SortControl.sequenceNumber :: INT (0..1)

**Определение**

Очередность экземпляра класса SortControl в данном запросе.

A.3.6.13.2 SortControl.elementName :: SC (0..1)

Словарный домен: ElementName

**Определение**

Элемент модели RIM, по которому должны сортироваться ответы на запрос.

A.3.6.13.3 SortControl.directionCode :: CS (0..1)

Словарный домен: Sequencing

**Определение**

Направление сортировки.

***Примеры — Возрастание, убывание, произвольный порядок.***

**A.3.7 Классы предметной области StructuredDocuments**

A.3.7.1 Класс ContextStructure (в предметной области StructuredDocuments)

**Свойства класса ContextStructure**

Атрибуты класса ContextStructure:

- setId :: II,
- versionNumber :: INT.

Класс ContextStructure является специализацией класса Act.

Класс ContextStructure имеет следующую специализацию:

- Document.

**Определение класса ContextStructure**

Контейнер внутри документа.

**Примечания к использованию**

Структуры имеют заголовки, которые могут кодироваться. Структуры могут быть вложенными и могут содержать подразделы.

Исходный отчет является первой версией. Его атрибуту setId присваивается новое уникальное значение, а атрибуту versionNumber присваивается значение «1».

Дополнение является приложением к существующему отчету, содержащим дополнительную информацию. Приложение само по себе является исходным отчетом. Родительским отчет, к которому оно прилагается, связывается с ним с помощью экземпляра класса ActRelationship, у которого атрибут ActRelationship.typeCode имеет значение «APND» (appends — дополняет). Дополняемый родительский отчет остается на своем месте, его содержание и состояние не изменяются.

Заменяющий отчет заменяет существующий отчет. Атрибут setId у заменяющего отчета имеет то же значение, что у родительского, а значение атрибута номера версии (versionNumber) увеличивается на 1. Состояние замененного родительского отчета изменяется на «superceded» (пересмотрен), но сам этот отчет должен остаться в системе для исторических ссылок.

**Открытый вопрос**

Имя этого класса и допустимые значения словарного домена ActClass должны быть пересмотрены с тем, чтобы они стали совместимы с иерархией словарного домена ActContainer, которая уже пересматривается (по состоянию на ноябрь 2004 года).

**Примечания к конструированию**

Каков статус пересмотра? Всегда ли класс ContextStructure описывает «отчет»?

A.3.7.1.1 ContextStructure.setId :: II (0..1)

**Определение**

Уникальный идентификатор отчета.

**Примечания к использованию**

Значение атрибута setId одинаково у всех версий, произведенных от одного и того же источника.

A.3.7.1.2 ContextStructure.versionNumber :: INT (0..1)

**Определение**

Уникальный идентификатор версии отчета.

**Свойство IsDocumentCharacteristic**

Это свойство должно иметь значение «true».

A.3.7.2 Класс Document (в предметной области StructuredDocuments)

**Свойства класса Document**

Атрибуты класса Document:

- completionCode :: CE,
- storageCode :: CE,
- copyTime :: TS,
- bibliographicDesignationText :: SET<ED>.

Класс Document является специализацией класса ContextStructure.

**Определение класса Document**

Специализация класса Act, содержащая характеристики, уникальные для служб управления документами.

**Атрибуты класса Document**

A.3.7.2.1 Document.completionCode :: CE (0..1)

Словарный домен: DocumentCompletion

**Определение**

Код, указывающий состояние завершенности отчета.

***Примеры — Незавершен, заверен, юридически заверен.***

A.3.7.2.2 Document.storageCode :: CE (0..1)

Словарный домен: DocumentStorage

**Определение**

Код, указывающий статус хранения отчета.

***Примеры — Активный, архивирован, удален.***

A.3.7.2.3 Document.copyTime :: TS (0..1)

**Определение**

Дата и время предоставления документа (например, копирования или отправки на устройство визуализации) системой управления документами, осуществляющей контроль версий документа.

**Примечания к использованию**

Этот атрибут был введен, чтобы читатель документа имел некоторое представление о том, как долго документ находился вне безопасного контекста его системы управления документами.

**Формальное ограничение**

Будучи присвоенным, это значение не может быть изменено.

A.3.7.2.4 Document.bibliographicDesignationText :: SET<ED> (0..*)

**Определение**

Библиографическая ссылка на документ, позволяющая идентифицировать его, найти и/или извлечь из общих собраний.

### A.4 Ассоциации

**A.4.1 (1..1) Acknowledgement:: acknowledgementDetail :: (0..*) AcknowledgementDetail:: acknowledgement**

Идентифицирует отношение между подтверждением и детальной информацией об ошибке, предупреждении или информации, передаваемой в этом подтверждении.

**A.4.2 (0..*) ActRelationship:: source :: (1..1) Act:: outboundRelationship**

**A.4.3 (0..*) ActRelationship:: target :: (1..1) Act:: inboundRelationship**

**A.4.4 (0..*) Participation:: act :: (1..1) Act:: participation**

**A.4.5 (1..*) Entity:: communicationFunction :: (0..*) CommunicationFunction:: entity**

Это отношение позволяет идентифицировать сущности, выполняющие различные коммуникационные функции.

**A.4.6 (1..*) Transmission:: communicationFunction :: (0..*) CommunicationFunction:: transmission**

Это отношение связывает акт передачи информации с ее отправителем, получателем, стороной обратного вызова и т. д.

**A.4.7 (0..1) QueryEvent:: controlAct :: (1..1) ControlAct:: queryEvent**

**A.4.8 (0..*) Role:: player :: (0..1) Entity:: playedRole**

**A.4.9 (0..*) Role:: scoper :: (0..1) Entity:: scopedRole**

**A.4.10 (1..1) Entity:: languageCommunication :: (0..*) LanguageCommunication:: entity**

**A.4.11 (0..*) SelectionExpression:: leftSide :: (0..1) LogicalExpression:: userAsLeft**

**A.4.12 (0..*) SelectionExpression:: rightSide :: (0..1) LogicalExpression:: userAsRight**

**A.4.13 (0..1) ControlAct:: payload :: (0..*) Message:: controlAct**

**A.4.14 (0..1) ParameterList:: parameter :: (0..*) Parameter:: parameterList**

Задает отношение между списком параметров и входящими в него параметрами.

**A.4.15 (0..*) Parameter:: queryByParameter :: (0..1) QueryByParameter:: parameter**

**A.4.16 (0..*) SelectionExpression:: queryBySelection :: (1..1) QueryBySelection:: selectionExpression**

**A.4.17 (0..*) SortControl:: querySpec :: (1..1) QuerySpec:: sortControl**

**A.4.18 (0..*) Participation:: role :: (1..1) Role:: participation**

**A.4.19 (0..*) RoleLink:: source :: (1..1) Role:: outboundLink**

**A.4.20 (0..*) RoleLink:: target :: (1..1) Role:: inboundLink**

**A.4.21 (0..*) Acknowledgement:: acknowledges :: (1..1) Transmission:: acknowledgedBy**

Идентифицирует отношение между актом передачи информации и подтверждениями этого акта.

**A.4.22 (0..*) Acknowledgement:: conveyingTransmission :: (1..1) Transmission:: conveyedAcknowledgement**

Идентифицирует отношение между подтверждением и актом его передачи.

**A.4.23 (0..*) AttentionLine:: transmission :: (1..1) Transmission:: attachment**

**A.4.24 (0..*) AttentionLine:: transmission :: (1..1) Transmission:: attentionLine**

Это отношение позволяет представить параметры транспорта, специфичного для технологии реализации, во внешней обертке сообщения стандарта V3.

**A.4.25 (0..1) Batch:: transmission :: (0..*) Transmission:: batch**

**A.4.26 (0..*) TransmissionRelationship :: source :: (1..1) Transmission:: outboundRelationship**

**A.4.27 (0..*) TransmissionRelationship :: target :: (1..1) Transmission:: inboundRelationship**

### A.5 Обзор модели RIM

#### A.5.1 Цель

Настоящий стандарт представляет собой краткий обзор базовых понятий и логических обоснований, положенных в основу разработки и развития модели HL7 RIM (Reference Information Model — эталонная информационная модель). Он не является «руководством по применению» модели RIM, хотя в нем и представлено несколько частных примеров ее использования, полезных для объяснения фундаментальных понятий разработки и развития модели RIM.

#### A.5.2 Общие сведения

Модель RIM строится в целях согласованного совместного использования данных во многих «местных» контекстах. Вообще говоря, чем шире сфера интересов, тем важнее сделать явными все предположения,

полагаемые в основу темы или области интереса. Модель RIM HL7 Версии 3 спроектирована как единая база и развитый источник всей информации, используемой в спецификации HL7. RIM конкретно и однозначно формулирует точные определения понятий здравоохранения («предметы интереса»), используемых в сфере информационных систем здравоохранения, а также описывает отношения (они же «ассоциации») между этими понятиями.

Спецификации стандарта HL7 Версии 3 (например, сообщения HL7 Версии 3, структурированные документы и т. д.) позволяют взаимодействовать (например, обмениваться данными) слабо связанным информационным системам, используемым при разных условиях оказания медицинской помощи, предоставляемой разными поставщиками и на разных территориях. Поэтому область применения модели RIM HL7 включает в себя всю информацию, которой должны обмениваться участвующие информационные системы здравоохранения в целях ее обработки. Кроме того, следует отметить, что она не моделирует (и не должна моделировать) информацию, хранящуюся в конкретной информационной системе здравоохранения, но никогда не передаваемую другим системам.

Модель RIM представлена с помощью синтаксиса визуального моделирования, основанного на Унифицированном языке моделирования UML (Unified Modeling Language). (Определенные различия между «стандартным UML» и «UML HL7», использованным в модели RIM, например размещение названий ассоциаций, в настоящее время рассматриваются техническим комитетом по моделированию и методологии HL7 (Modeling and Methodology Technical Committee) в целях движения к максимальному согласованию обоих синтаксисов.) Комитет HL7 также ведет базу данных («хранилище модели RIM»), содержащую детальную информацию о каждом понятии модели, каждом атрибуте и каждой ассоциации, включая логическое обоснование элемента, определение, ограничения и историю редактирования/изменений. В перспективе эта информация может быть формально опубликована (частично или целиком) в виде профиля UML, специфичного для стандарта HL7.

Важно отметить, что модель RIM создается не для того, чтобы стать логической или физической моделью базы данных или иной структуры информационной системы конкретного производителя либо обслуживать интересы конкретной организации здравоохранения или сети организаций. В действительности модель RIM даже не представляет конкретные группы сообщений HL7, она является обобщенной совокупностью данных и отношений между данными, из которых может быть сконструировано любое релевантное сообщение HL7. Ожидается, что по мере необходимости конкретные пользователи модели RIM используют ее релевантные части, принимая во внимание их содержание при разработке и описании своих информационных моделей.

**А.5.3 Обоснование конструкции модели RIM**

Разветвленная структура RIM основана на шести «базовых» классах: Act (действие), Entity (сущность), Role (роль), Participation (участие), ActRelationship (связь действий) и RoleLink (связь ролей). Определения этих классов приведены в разделе А.1.4. Далее приводятся обсуждение фундаментальных принципов, положенных в основу этих шести классов и их связей, а также базовые рекомендации по их использованию.

В модели HL7 RIM выделены два основных «высокоуровневых» понятия, являющихся фундаментальными для понимания мира информации здравоохранения: намеренные «действия» или «услуги» (экземпляры класса Act) и «люди, места и предметы», представляющие интерес в сфере здравоохранения (экземпляры класса Entity).

Класс Act (и его подклассы) представляет все намеренные действия, документируемые медицинским работником в клиническом или административном контексте. Появление класса Act в качестве одного из центральных классов модели RIM отражает следующую позицию комитета HL7: с точки зрения информационного взаимодействия и обмена сообщениями оказание медицинской помощи состоит из последовательности атрибутированных намеренных действий. Таким образом, экземпляры класса Act описывают и клинические исследования (например, температуру пациента), и вмешательства (например, применение лекарств), и административные действия (например госпитализацию пациента). Следует учесть, что с такой «действиецентричной» точки зрения на оказание медицинской помощи действие исследования имеет два очевидно противоположных значения: «действие установления и документирования конкретного факта» и «описание исследуемого предмета». Другими словами, экземпляр класса Observation, являющегося специализацией класса Act, описывает как атрибутированное действие исследования, так и результаты исследования. Оба аспекта этого расширенного определения класса Observation отражены в специфических атрибутах класса Act или его подкласса Observation.

Класс Entity (и его подклассы) представляет все живые субъекты (например, людей, животных), организации (как формальные, так и неформальные), материалы (например, прочные и непрочные предметы, продукты, биоматериал, контейнеры) и места, которые могут представлять интерес в передаваемом контексте оказания медицинской помощи. Необходимо отметить, что понятие «коллекции информации» (например, медицинская карта) представляется не в виде экземпляра класса Entity или его подклассов, а моделируется как коллекция атрибутированных действий, описываемых экземплярами класса Act.

Между классами Act и Entity модель RIM помещает два дополнительных класса — Role (роль) и Participation (участие). Класс Role моделирует несколько важных понятий, превалирующих в предметной области оказания медицинской помощи. Во-первых, класс Role отражает тот факт, что в конкретном контексте медицинской помощи различные «статические» объекты могут «временно» выполнять одну или несколько «ролей» (например, паци-

ент, врач общей практики, ответственная сторона, медицинская сестра и т. д.). Во-вторых, понятия «способности» (например, возможность оказания медицинской помощи в соответствии с рекомендациями Advanced Cardiac Life Support) и «сертификации» (например, диплом медицинской сестры) также могут моделироваться с помощью экземпляров класса Role. Наконец, тщательное изучение кратности (0..1) и имен (контролер, исполнитель) двух ассоциаций между классами Entity и Role показывает, что класс Role может использоваться для «группировки» экземпляров класса Entity.

Важно различать понятие сущности, имеющей определенную роль, от понятия «функциональной роли, выполняемой этой сущностью в контексте определенного действия», специфичной для конкретного действия. Функциональные роли моделируются с помощью экземпляров класса Participation. Например, анестезиолог-ординатор (сущность, имеющая роль) применяет анестезию (что моделируется с помощью экземпляра класса Participation, описывающего функциональную роль «поставщика» в действии «применить анестезию») к пациенту (что моделируется с помощью экземпляра класса Participation, описывающего функциональную роль «получателя» в действии «применить анестезию»). Следует учесть, что отсутствие прямой ассоциации между классами Participation и Entity вытекает из принципиального положения модели HL7 RIM, согласно которому все экземпляры класса Entity, вовлеченные в действие, указанное в экземпляре класса Act, описывают участие сущностей в этом действии в конкретной роли, указанной в экземплярах класса Role.

В целом классы Participation и Role необходимы для полного моделирования сложной семантики взаимоотношений экземпляров классов Entity и Act. Точное определение точки зрения на оказание медицинской помощи, положенной в основу модели RIM, гласит: на самом верхнем уровне абстракции оказание медицинской помощи представляет собой ряд намеренных атрибутивных действий, в которых разными способами участвуют (выполняют, действуют по поручению, пользуются результатами и т. д.) несколько сущностей, имеющих определенные роли (например, «Джон Смит в роли пациента») и выполняющих определенные функции, описанные экземплярами класса Participaton (например, «участковый врач» и т. д.).

Два остальных базовых класса модели RIM — ActRelationship и RoleLink — используются, чтобы «объединить» или «связать» экземпляры классов, с которыми они ассоциированы. Класс RoleLink используется, чтобы описать «связь на основе зависимости» (например, подотчетность, цепочку доверительных отношений и т. д.) между двумя экземплярами сущностей, выполняющих определенные роли. Семантика класса ActRelationship объясняется далее.

**А.5.4 Связывание действий. Семантика класса ActRelationship**

Понимание семантики и применения класса ActRelationship начинается с понимания «фрактальной» или «пошаговой» природы совокупности действий. В свою очередь, для этого лучше всего отправляться от разветвленной структуры классификации трех типов «связывания отношений», представленных экземплярами класса ActRelationship: «целое/часть» (например, панели лабораторных анализов; см. далее обсуждение «пошаговых действий»); «связь на основе правил» (например, план лечения, протоколы и т. д.); «когнитивные действия» (например, суждение, переименование, замена, категоризация, обоснование, причина и т. д.).

В части уже упомянутой «фрактальной» или «пошаговой» природы экземпляры классов ActRelationship могут использоваться для моделирования понятий «фрактальных» или «пошаговых» отношений внутри иерархии «целое/часть». Рассмотрим хирургическую процедуру, например, лапароскопическую холецистэктомию. Она может быть представлена как единственный экземпляр класса Act или же в качестве альтернативы как «совокупность» (частично упорядоченных) экземпляров класса Act, каждый из которых описывает более тонкие детали, например, получить согласие на проведение операции, применить перед операцией определенные лекарства, управлять анестезией (в течение всей хирургической процедуры), сделать разрез и т. д. В свою очередь, каждое из этих более тонких действий может быть расчленено на еще более тонкие действия. Уровень степени детализации четко зависит от контекста действия или действий и степени интереса или намерений стороны, выполняющей (или не выполняющей) декомпозицию действий. На рисунке 15 показана «точка зрения хирурга» на некоторые экземпляры классов Act и ActRelationship при моделировании действия «холецисэктомия».

Прямоугольники с острыми углами представляют экземпляры класса Act (у каждого из них атрибут moodCode имеет значение «DEF», что означает описание действия). Скругленные прямоугольники представляют экземпляры класса ActRelationship, у которых атрибут typeCode имеет значение «COMP» (has component — имеет компонент). Значения атрибута sequenceNumber определяют порядок связи. Каждое действие, в свою очередь, может быть расщеплено на компоненты плана.

В целом классы Act и ActRealshionship необходимы для полного моделирования семантики каждого из трех упомянутых ранее типов «связывания отношений» с той степенью грубости или тонкости, которая необходима для конкретного контекста передачи данных. Кроме того, различные типы связывания и разные уровни детализации, представленные экземплярами класса ActRealshionship, могут (и будут) использоваться для коллективного формирования сложной семантики клинического мышления. Например, экземпляр класса ActRealshionship (описывающий связь «имеет обоснование») может использоваться для формирования связи экземпляра класса Observation, описывающего конкретный результат лабораторного анализа (например, скорость оседания эритроцитов равна 48), с экземпляром класса Observation, описывающим конкретный диагноз (например, системная красная волчанка).

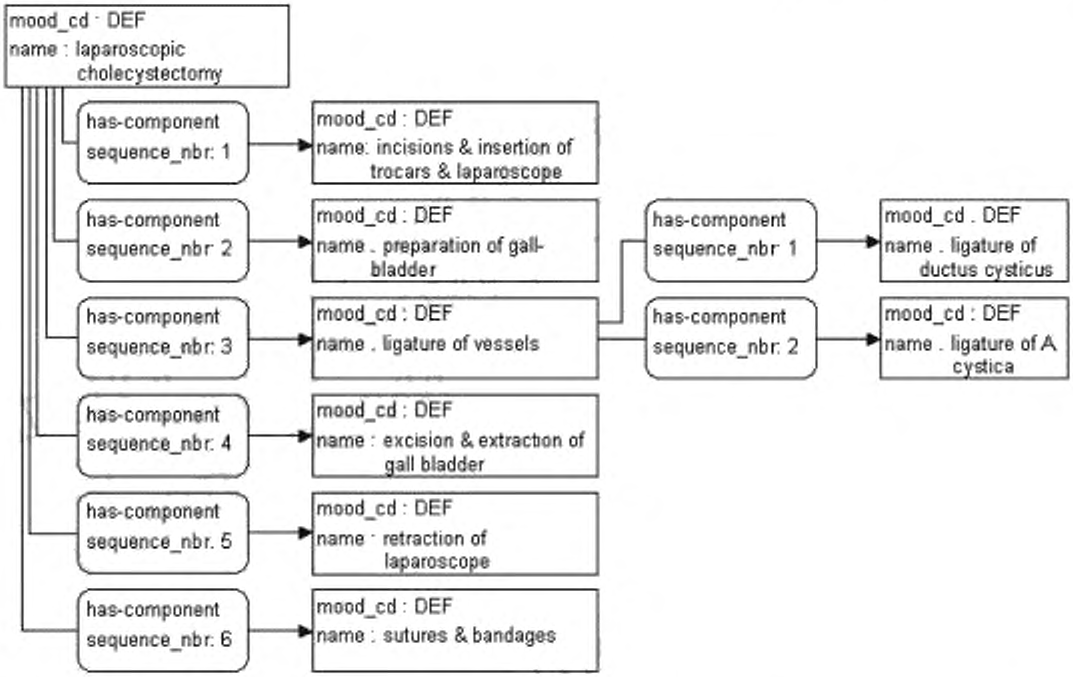


Рисунок А.15 — Пример конструкции последовательного плана лапароскопической холецистэктомии

**А.5.5 Определения шести базовых классов RIM**

А.5.5.1 Класс Act (действие)

Класс Act описывает интересующее действие, которое произошло, может произойти, происходит, запланировано или затребовано. Оно должно быть намеренным действием в предметной области стандарта HL7. Оказание медицинской помощи (и любая профессиональная или деловая деятельность) состоит из намеренных действий. Экземпляр класса Act представляет собой запись о таком намеренном действии.

А.5.5.2 Класс Entity (сущность)

Класс Entity описывает класс физических сущностей, или конкретный экземпляр такой сущности, или организацию либо группу физических сущностей, которые могут участвовать в действиях, артефакт. К ним относятся живые существа (включая людей), организации, материал и места. Иерархия специализаций класса Entity охватывает информацию о людях, организациях, живых организмах, устройствах, фармацевтических субстанциях и т. д. Она не включает в себя информацию о событиях, действиях, деятельности, определения предметов или о ролях, выполняемых сущностями (например, пациент, поставщик медицинской помощи).

А.5.5.3 Класс Role (роль)

Класс Role описывает компетентность сущности, выполняющей роль в соответствии с определениями, данными другой сущностью, контролирующей эту роль.

Сущность, имеющая конкретную роль, может участвовать в действии. Следует отметить, что конкретная сущность в конкретной роли может участвовать в действии многими способами. Например, лицо в роли врача может участвовать в ведении пациента как дежурант или как лечащий врач. Роль определяет компетентность сущности, не зависящую от любого действия, в противоположность участию, которое ограничено рамками действия.

Каждая роль выполняется одной сущностью (той, что имеет эту роль) и обычно «контролируется» другой сущностью. Например, роль «пациент» может выполняться (обычно) некоторым лицом и контролироваться организацией, оказывающей пациенту медицинскую помощь. Аналогичным образом работодатель контролирует роль «работника».

А.5.5.4 Класс Participation (участие)

Класс Participation моделирует ассоциацию между классом роли Role и классом действия Act. Экземпляр класса Participation описывает участие сущности, имеющей определенную роль, в ассоциированном действии. Один и тот же экземпляр класса Role может быть связан с помощью экземпляров класса Participation с несколькими экземплярами класса Act, а один и тот же экземпляр класса Act может быть связан с помощью экземпляров класса Participation с несколькими экземплярами класса Role. Один экземпляр класса Participation всегда является

ассоциацией между конкретным экземпляром класса Role и конкретным экземпляром класса Act. Участие ограничено рамками действия, роль же, напротив, описывает компетенцию сущности, не зависящую от какого-либо действия.

A.5.5.5 Класс ActRelationship (связь действий)

Класс ActRelationship моделирует ассоциацию между двумя экземплярами класса Act. Примерами могут служить такие отношения между действиями, как целое/часть, предшественник/последователь, причина/следствие.

Класс ActRelationship имеет две ассоциации с классом Act, одна из них имеет имя source (источник), вторая — target (цель). Связи, ассоциированные с экземпляром класса Act, должны рассматриваться как свойства экземпляра-источника. Это означает, что источник информации, передаваемой в экземпляре класса Act, несет ответственность не только за значения атрибутов этого экземпляра, но и за все связи с этим экземпляром, имеющие его в качестве источника.

Правило приписывания состоит в том, что все экземпляры класса ActRelationship приписаны ответственному действующему лицу того экземпляра класса Act, который является источником ассоциации с экземплярами класса ActRelationship (действием-источником).

A.5.5.6 Класс RoleLink (связь ролей)

Класс RoleLink описывает связь между двумя ролями, выражающую зависимость между этими ролями.

A.5.5.7 Подклассы классов Act, Entity и Role

Классы Act, Entity и Role представляют собой классы «высокого уровня», хотя и не являются «абстрактными» в формальном смысле (то есть значащие экземпляры этих классов вполне обычны). Поэтому потребовалось определить некоторое число более специализированных подклассов этих трех классов, чтобы указать дополнительные данные (атрибуты классов), требуемые в более конкретном контексте (например, класс Observation, являющийся подклассом класса Act, и классы LivingSubject и Material, являющиеся подклассами класса Entity). Атрибуты подкласса должны быть и полезны, и уникальны для этого подкласса. Подклассы наследуют все атрибуты родительского класса.

В каждой из этих иерархий есть значимые подклассы, для которых не нужны дополнительные атрибуты, поэтому они не представляются как отдельные классы в RIM. Атрибут «classCode» в каждой из этих иерархий указывает, какой именно подкласс представлен данным классом. Система кодирования значений атрибута «classCode» строго контролируется комитетом HL7. Второй атрибут в каждой иерархии — атрибут «code» — используется для дальнейшей классификации подтипов каждого подкласса.

A.5.5.8 Понятие наклонения

Класс Act описывает намеренные действия. Эти действия могут существовать в разных «наклонениях» («moods»). Наклонения описывают цикл деятельности от определения деятельности к ее планированию, от запланированной или потребованной деятельности к завершенной. Наклонение действия, описанного в экземпляре класса Act, определяется значением атрибута Act.moodCode.

Любой экземпляр класса Act имеет одно и только одно наклонение и не меняет его на протяжении своего жизненного цикла. Наклонения — определение, намерение, требование, событие — описывают разные моменты жизненного цикла деятельности. Однако участники каждого такого момента деятельности различны, как и данные, которыми они оперируют. Поэтому наклонение, описанное в экземпляре класса Act, является статическим. Актуализацию процесса деятельности (то есть движение от ее определения к планированию, требованию и выполнению) называют «циклом деятельности», чтобы отличить ее от «жизненного цикла» единственного экземпляра действия. Экземпляры класса Act, формирующие такой «цикл деятельности», связаны между собой с помощью экземпляров класса ActRelationship.

### A.5.6 Типы данных и спецификации словарных доменов

Определения классов модели RIM, атрибутов и ассоциаций дают информацию о логическом значении этих элементов, но для полного определения необходимы типы данных и спецификации словарных доменов. Типы данных определяют допустимые значения атрибутов и «смысл» этих значений. Поэтому типы данных являются фундаментальными строительными блоками, образующими (и ограничивающими) всю семантику, которую в конечном счете можно выразить в модели RIM. В стандарте HL7 Версии 3 спецификации типов данных описаны в документах Data Types Abstract Specification (абстрактная спецификация типов данных) и V3 Data Types Implementable Technology Specification for XML (технологическая спецификация реализации типов данных HL7 Версии 3 на языке XML). Сводный перечень типов данных приведен также в разделе A.6.

Спецификации словарей значений атрибутов указывают словарный домен по умолчанию и «квалификатор расширяемости» («coding strength»). Словарные домены явно определены в приложении C. Спецификации словарей документируют перекрестные ссылки и альтернативные представления систем кодирования, а также описывают логические понятия, выражаемые каждым кодом. Каждый кодированный атрибут может принимать значения только из соответствующего словарного домена.

Квалификатор расширяемости (coding strength) является кодированным свойством атрибута, указывающим, может ли пользователь стандарта HL7 Версии 3 передавать кодированное значение, не входящее в словарный домен. Такое значение называется «исключением» для словарного домена. Квалификатор расширяемости может иметь одно из двух значений. Значение квалификатора расширяемости «CWE» (Coded with Exceptions — кодируемый с исключениями) указывает, что конечный пользователь может передавать кодированный термин, взятый из

местной системы кодирования, а не из формально определенного словарного домена. Значение квалификатора расширяемости «CNE» (Coded, No Exceptions — кодируемый без исключений) указывает, что экземпляры сообщений могут содержать только коды из словарного домена, ограничивающего значения данного атрибута.

**А.5.7 Методология стандарта HL7 Версии 3 и модель RIM**

В целом основным мотивом разработки модели RIM было желание точного определения различных элементов данных и отношений, образующих информационное пространство здравоохранения. Прямым результатом такого способа применения знаний явилась способность повторного использования того же самого понятия во многих сообщениях передачи медицинских данных. Полный процесс определения сообщения определен и обсужден в документе V3 Guide и в дополняющем его руководстве HL7 Message Definition Framework (MDF). Основными шагами этого процесса являются определение и документирование потребностей в обмене данными (например, комплекса сообщений для поддержки конкретного клинического или административного процесса) в форме сценариев (storyboard), выбор из модели RIM тех классов и атрибутов, которые необходимы для составления данного сообщения, и последующее применение дополнительных ограничений на число повторений и допустимые значения каждого атрибута.

Дополнительную информацию см. в упомянутых ранее документах V3 Guide и MDF.

**А.6 Сводный перечень типов данных HL7 Версии 3**

**А.6.1 Обзор типов данных**

В таблице А.1 приведены определения типов данных, которые были приняты как часть стандарта HL7 Версии 3 по результатам голосования, проведенного 2 декабря 2002 г. Следующие версии модели RIM будут содержать полную спецификацию типов данных, а тело модели RIM будет содержать гиперссылки на определения типов данных.

Таблица А.1 — Типы данных в модели RIM

| Имя | Символ | Описание |
|---|---|---|
| DataValue | ANY | Определяет основные свойства каждого типа данных. Это абстрактный тип, означающий, что никакое значение не может быть только значением данных, не принадлежащим никакому конкретному типу. Каждый конкретный тип является специализацией этого общего абстрактного типа DataValue |
| Boolean | BL | Булевский тип представляет значения двузначной логики. Булевское значение может быть или TRUE (истина), или FALSE (ложь), или, как и любое другое значение, может быть пустым (NULL) |
| Encapsulated Data | ED | Данные, в основном рассчитанные на чтение человеком или на дальнейшую машинную обработку данных за пределами области применения стандарта HL7. К ним относятся неформатированный или форматированный текст, мультимедийные данные или структурированная информация, определенная другим стандартом (например, XML-подписи). Вместо самих данных значение типа ED может содержать только ссылку на данные (см. описание типа данных TEL). Заметьте, что тип данных ST — это специализация типа данных ED, у которой компонент типа среды ED имеет значение «text/plain» (только текст) |
| Character String | ST | Строковый тип данных, предназначенный для текстовых данных, используемых главным образом для машинной обработки (например, сортировка, запрос, индексирование и т. д.). Используется для имен, символов и формальных выражений |
| Concept Descriptor | CD | Описание понятия, предназначенное для представления любого вида понятий, обычно с помощью кода, определенного в некоторой системе кодирования. Описание понятия может содержать исходный текст или фразу, послужившую основой для кодирования, и один или несколько эквивалентов этого кода в других системах кодирования. Описание понятия может также содержать квалификатор, используемый, к примеру, для описания понятия «левая нога» как посткоординированного термина, образованного от исходного кода «нога» и квалификатора «левый». В исключительных случаях описание понятия может не содержать код, а только исходный текст, описывающий это понятие |
| Coded Simple Value | CS | Кодированные данные в их простейшей форме, в которой указан только код без его значения, предназначенного для вывода на экран. Система кодирования и ее версия определяются контекстом, в котором передается значение типа CS. Тип данных CS используется для значений кодированных атрибутов, имеющих единственный словарный домен, определенный в стандарте HL7 |

*Продолжение таблицы А.1*

| Имя | Символ | Описание |
|---|---|---|
| Coded With Equivalents | CE | Кодированные данные, состоящие из кодированного значения (типа CV) и необязательно одного или нескольких кодированных значений, принадлежащих другим системам кодирования и описывающих то же самое понятие. Используется при наличии альтернативных кодов |
| Character String with Code | SC | Строка символов, к которой при необходимости может быть добавлен код. Если код присутствует, то должен присутствовать и текст. Код часто берется из местной системы кодирования |
| Instance Identifier | II | Идентификатор, который уникально идентифицирует предмет или объект. Примерами могут служить объектные идентификаторы (ОИД) объектов RIM HL7, номер медицинской карты, идентификатор направления, идентификатор позиции прейскуранта, идентификационный номер транспортного средства и т. д. Идентификаторы экземпляров определяются на основе объектных идентификаторов ветви ИСО |
| Telecommunication Address | TEL | Номер телефона (голосового или факса), адрес электронной почты или другой указатель ресурса, управляемого телекоммуникационным оборудованием. Адрес указывается в форме Единого указателя ресурсов URL (Uniform Resource Locator), к которому добавлены спецификация времени и коды использования, помогающие установить, какой адрес используется в конкретное время и для каких целей |
| Postal Address | AD | Почтовый, домашний или юридический адрес. Последовательность компонентов адреса, например, улица или почтовый ящик, город, почтовый индекс, страна и т. д. |
| Entity Name | EN | Фамилия, имя, отчество лица, название организации, места или предмета. Последовательность компонентов, например имя или фамилия, приставка, суффикс и т. д. Примерами могут служить «Джим Боб Уолтон, мл.», «Health Level Seven, Inc.», «Озеро Тахо» и т. д. Именование объекта может быть простой строкой символов или состоять из нескольких компонентов, например, «Джим», «Боб», «Уолтон» и «мл.»; «Health Level Seven» и «Inc.»; «Озеро» и «Тахо» |
| Trivial Name | TN | Ограничение именования сущности в форме одной строки, используемой для простого названия предметов и мест |
| Person Name | PN | Фамилия, имя, отчество лица. Последовательность компонентов, например, имя, фамилия, приставка, суффикс и т. д. |
| Organization Name | ON | Наименование организации. Последовательность компонентов наименования |
| Integer Number | INT | Целые числа (–1,0,1,2,100,3398129 и т. д.) — точные числа, являющиеся результатами подсчета и нумерации. Целые числа дискретны, набор целых чисел бесконечен, но счетен. Диапазон значений целых чисел не ограничен никаким произвольным пределом. Для типа данных INT определены две причины пустоты (NULL flavor), описывающие положительную и отрицательную бесконечность |
| Real Number | REAL | Вещественные числа. Обычно используются в тех случаях, когда количества измеряются, оцениваются или вычисляются из других вещественных чисел. Типичным представлением вещественного числа служит десятичная запись, в которой число значащих десятичных цифр известно как точность |
| Ratio | RTO | Количество, представленное как отношение числителя к знаменателю. Общие множители в числителе и знаменателе автоматически не сокращаются. Тип данных RTO используется для титров (например, «1:128») и других величин в результатах лабораторных анализов, которые действительно представляют собой отношения. Отношения не являются просто «структурированными числовыми данными», поэтому измерения артериального давления (например, «120/60») не являются отношениями. Во многих случаях вместо типа данных RTO должен использоваться тип данных REAL |
| Physical Quantity | PQ | Размерная величина, выражающая результат измерения |

*Окончание таблицы А.1*

| Имя | Символ | Описание |
|---|---|---|
| Monetary Amount | MO | Денежная сумма, выражающая количество денег в некоторой валюте. Валюты — единицы, в которых указаны денежные суммы в разных экономических регионах. В то время как денежная сумма является единственным видом количества (денег), обменные курсы валют являются переменными. Это принципиальное отличие между физической величиной и денежными количествами, а также причина того, почему единицы валюты не являются физическими единицами |
| Point in Time | TS | Величина, указывающая момент времени на оси естественного времени. Момент времени чаще всего представляется как календарное выражение |
| Set | SET | Значение, содержащее другие различные значения в произвольном порядке |
| Sequence | LIST | Значение, содержащее другие дискретные значения в определенной последовательности |
| Bag | BAG | Неупорядоченная коллекция значений, в которой каждое значение может содержаться более одного раза |
| Interval | IVL | Множество последовательных значений упорядоченного базового типа данных |
| History | HIST | Множество значений данных, которые соответствуют типу исторических данных (HXIT), то есть имеют свойство действительного времени. Историческая информация не ограничена прошлым; в ней могут присутствовать ожидаемые будущие значения |
| Uncertain Value — Probabilistic | UVP | Общее расширение типа данных, используемое, чтобы указать вероятность, отражающую степень доверия поставщика данных к их значениям |
| Parametric Probability Distribution | PPD | Общее расширение типа данных, выражающее неопределенность количественных данных с помощью функции распределения и ее параметров. Помимо специфических параметров распределения всегда указывается среднее значение (математическое ожидание) и стандартное отклонение, чтобы обеспечить минимальный уровень интероперабельности на тот случай, если приложение-получатель не в состоянии работать с определенным распределением вероятности |
| General Timing Specification | GTS | Множество моментов времени, указывающее времена событий и действий, а также шаблоны циклов, которые могут применяться к некоторым видам информации, например, к телефонным номерам (утро, вечер), к адресам (для так называемых «перелетных птиц», которые живут зимой на юге, а летом на севере), а также к часам работы |

**Приложение В**
**(справочное)**

## Типы данных. Абстрактная спецификация

Основной/технический редактор: Gunther Schadow, gunther@aurora.rg.iupui.edu, Regenstrief Institute for Health Care.

Технический редактор: Paul Biron, paul.v.biron@kp.org, Kaiser Permanente, Southern California.

Технический редактор: Lloyd McKenzie, lmckenzi@ca.ibm.com, IBM Global Services.

Технический редактор: Grahame Grieve, grahame@kestral.com.au, Kestral Computing Pty. Ltd.

Технический редактор: Doug Pratt, Douglas.Pratt@siemens.com, Siemens.

Дата и время последней публикации: 20090130 08:47

**Предисловие**

Типы данных, используемые в стандарте HL7 Версии 3, определены в настоящем документе на абстрактном уровне, не зависящем от представления. Под «независимостью от представления» понимается независимость как абстрактного синтаксиса, так и его применения от конкретной технологии реализации.

Настоящее приложение сопровождается спецификациями технологии реализации ITS (Implementation Technology Specification). Документы ITS могут служить краткими справочниками по типам данных, которые лучше ориентированы на практическое представление этих типов с помощью данной технологии реализации.

Для ускоренного информирования в таблицах словарных значений, приведенных в настоящей спецификации, перечислено текущее содержание словарных доменов. Однако в любой заданный момент времени нормативным источником этих доменов являются таблицы словарных значений, хранящихся в базе данных модели RIM. Для некоторых больших доменов приведены только примеры возможных значений. Для получения полного содержания домена можно осуществить поиск по имени домена, ассоциированному с таблицей значений, хранящейся в словаре модели RIM.

**Выражения признательности**

Настоящий стандарт воплощает результат многолетней интенсивной работы с использованием электронной почты, телеконференций и обсуждений на различных встречах. Ее содержание согласовано с помощью голосования. Выражаем признательность многих лицам, которые в разное время участвовали в проектировании, дискуссиях и подаче предложений на голосование. Председатель рабочей группы по разработке этой спецификации Gunther Schadow (Regenstrief Institute for Health Care) возглавлял эту работу и стал основным автором настоящего стандарта. В разное время соредакторами этого стандарта были Paul V. Biron (Kaiser Permanente), Doug Pratt (Siemens), Lloyd McKenzie (IBM) и Grahame Grieve (Kestral Computing Pty. Ltd.). Основной вклад в форме размышлений и поддержки внесли Mark Tucker (Regenstrief Institute), George Beeler, Stan Huff (Intermountain Health Care), а также Mike Henderson (Kaiser Permanente), Anthony Julian (Mayo), Joann Larson (Kaiser Permanente), Mark Shafarman (Oasis Healthcare Systems), Wes Rishel (Gartner Group) и Robin Zimmerman (Kaiser Permanente). Выражаем признательность за критические замечания и предложение новых идей Bob Dolin (Kaiser Permanente), Clem McDonald (Regenstrief Institute), Kai Heitmann (HL7 Germany), Rob Seliger (Sentillion) и Harold Solbrig (Mayo Clinic). Неоценимую поддержку оказали члены рабочей группы Laticia Fitzpatrick (Kaiser Permanente), Matt Huges, Randy Marbach (Kaiser Permanente), Larry Reis (Wizdom Systems), Carlos Sanroman (Kaiser Permanente), Greg Thomas (Kaiser Permanente). Благодарим James Case (University of California, Davis), Norman Daoust (Partners HealthCare Systems), Irma Jongeneel (HL7 The Netherlands), Michio Kimura (HL7 Japan), John Molina (SMS), Richard Ohlmann (McKessonHBOC), David Rowed (HL7 Australia) и Klaus Veil (Macquarie Health Corp., HL7 Australia) за предоставление своих знаний при решении критичных вопросов. Разработка настоящей спецификации была обеспечена организацией Regenstrief Institute for Health Care.

**Открытые вопросы**

Спецификация соответствия с помощью ограничения типов данных.

### В.1 Введение

#### В.1.1 Что такое тип данных?

Каждый элемент данных имеет определенный тип данных, определяющий смысл (семантику) значений, которые могут быть присвоены элементу данных. Для содержательного обмена данными необходимо знать определения передаваемых значений. Это верно как для комплексных ««значений», например деловых сообщений, так и для более простых значений, например, строк символов или целых чисел.

В соответствии со стандартом ИСО 11404 тип данных представляет собой «совокупность различных значений, характеризуемую свойствами этих значений и операциями над этими значениями». С типом данных связаны

специфика и широта. Спецификой типа данных являются свойства, присущие каждому значению этого типа. Широта типа данных оценивается множеством значений, имеющих этот тип («набором значений» этого типа).

Семантическими свойствами типов данных является то, что стандарт 11404 называет «свойствами этих значений и операциями над этими значениями». Семантическому свойству типа данных присваивается некоторое имя. Семантическое свойство имеет определенное значение для каждого экземпляра этого типа данных. Значение свойства экземпляра данных само должно иметь определенный тип данных — не существует никаких экземпляров данных, для которых нельзя было бы определить тип данных.

Поэтому типы данных являются основными строительными блоками, используемыми для конструирования сущностей более высокого порядка — сообщений, компьютеризованных медицинских документов, объектов деловой сферы и транзакций с этими объектами. В чем же тогда состоит различие между типом данных и сообщением, документом или деловым объектом? Значения типа данных существуют сами по себе, рассматривается только значение, для него не определяются ни идентичность, ни состояние, ни изменение состояния. Напротив, для деловых объектов можно отслеживать идентичность и состояние, свойства идентифицированного объекта могут изменяться с течением времени. Этого нет у значений данных: значение и его свойства постоянны. Например, число 5 всегда является числом 5, между этим числом 5 и тем числом 5 нет никакой разницы (они не обладают идентичностью), число 5 никогда не станет числом 6 (состояние не изменяется). Значения данных можно рассматривать как неизменные объекты, идентичность которых не играет никакой роли (идентичность и равенство суть одно и то же)[1)].

**В.1.2 Представление значений данных**

Значения данных могут быть представлены с помощью разных символов, но смысл значения не связан ни с каким конкретным представлением.

Например, порядковые номера (неотрицательные целые числа) намеренно определены как тип данных, в котором для каждого значения есть следующее значение, а нуль не следует ни за каким порядковым значением. На основе этого определения можно определить операции сложения, умножения и другие математические операции. Какое бы представление ни отражало правила, описанные в намеренном определении порядкового типа данных, оно будет действительным представлением порядковых номеров. Примерами действительных представлений порядковых номеров могут служить строки десятичных цифр, пакеты стеклянных шариков или царапины на стене. Число «пять» может быть представлено словом «пять», арабской цифрой «5» или римской цифрой «V». Представление не играет роли, пока оно соответствует семантическому определению типа данных.

Другим примером может служить булевский тип данных, который определяется совокупностью двух разных значений, соответствующих истине (true) и лжи (false), и правилами отрицания значения и сочетания этих значений в операциях конъюнкции и дизъюнкции. Булевские значения могут быть представлены словами «true» и «false», «да» и «нет», числами 0 и 1 или любыми двумя знаками, отличающимися друг от друга. Представление значений типа данных не играет роли, пока оно соответствует семантическому определению типа данных.

Настоящий стандарт определяет семантику, смысл типов данных, используемых в стандартах HL7. **Она касается только семантики и не зависит от особенностей представления или обработки либо от специфической технологии реализации.** Для различных технологических подходов представления значений определенных в ней типов данных разработаны дополнительные стандарты. Они называются «Спецификацией реализуемой технологии» (Implementable Technology Specification — ITS). Эти стандарты указывают, как должны представляться значения, соответствующие семантическим определениям, приведенным в настоящей спецификации, они могут задавать синтаксис символьного или двоичного представления, а также описывать компьютерные процедуры выполнения действий над этими представлениями значений данных. Смысл этих представлений, используемых при передаче данных, при создании данных и их обработке компьютерными программами определяется на основе настоящего стандарта, являющегося семантической спецификацией типов данных.

**В.1.3 Свойства значений данных**

Значения данных имеют свойства, определяемые их типом данных. Наиболее общим примером таких свойств являются «поля комплексных типов данных». Однако в более общем смысле под свойством значения данных надо подразумевать логические предикаты или математические функции; в более простых, но тем не менее корректных терминах свойства являются вопросами, которые надо задать о значении данных, чтобы получить в ответ другое значение данных.

Свойство обозначается именем. Например, тип данных integer (целое число) может иметь свойство с именем «sign» (знак). Свойство имеет домен, образованный множеством возможных «ответов». Это множество определяется типом данных свойства, но домен свойства может быть подмножеством набора значений этого типа данных.

Свойство может иметь аргументы, то есть дополнительную информацию, которую надо представить в вопросе для получения ответа. Например, важным свойством целого числа является то, что результат сложения одного

---

[1)] В методологии HL7 Message Development Framework определены «режимы изменения» полей сообщения. Поскольку значения данных не обладают ни идентичностью, ни состоянием, ни изменением состояния, то эти режимы не применяются к свойствам значений данных. Значения данных и их свойства никогда не изменяются. Поле объекта (например, сообщения) может быть изменено, и в этом случае значение этого поля заменяется другим значением. Но само значение никогда не изменяется.

целого числа с другим также представляет собой целое число. Поэтому для свойства целого числа «plus» (сложение) требуется аргумент, а именно другое целое число.

Имеют ли семантические свойства аргументы или нет, не является сколько-нибудь принципиальным отличием. Семантическое свойство типа данных, не имеющее аргументов, не обязательно представляет собой «поле комплексного типа данных». Например, для целочисленных значений можно определить свойство is-zero, принимающее булевское значение «true», если число равно нулю, и «false», если оно отлично от нуля. Это отнюдь не означает, что свойство is-zero должно быть явным компонентом любого представления целого числа.

Семантическое свойство типа данных, имеющее аргументы, не имеет специфических операционных обозначений наподобие «вызова процедуры», «передачи аргументов», «возвращения значений», «инициирования исключений» и т. д. Все перечисленное относится к реализации типов данных в компьютерных системах, но не имеет отношения к семантике типов данных.

**Настоящий стандарт затрагивает только семантику типов данных. В нем не обсуждается ни синтаксис представления значений (даже абстрактный синтаксис), ни интерфейс операций над значениями данных.**

**B.1.4 Необходимость абстрагирования**

**Почему в настоящем стандарте делается такой акцент на абстрагирование от синтаксиса представления и реализации операций?**

Такой вид абстрактной семантической спецификации типа данных необходим в стандартах HL7 по весьма практическим соображениям. Одной из важных особенностей конструирования стандартов HL7 Версии 3 является открытость по отношению к технологиям представления и реализации. Предполагается, что все спецификации стандартов HL7 Версии 3 будут представляться в форме, независимой от технологий представления и реализации. Комитет HL7 отдает себе отчет о том, что какое-то время некоторые технологии представления и реализации являются более популярными, но технологии подвержены изменениям, и при изменении технологии представление данных также изменится. Основной областью применения стандартов HL7 является обработка информации в сфере здравоохранения, не зависящая от технологии, обеспечивающей эту обработку. Комитет HL7 рассчитывает, что спецификации, не зависящие от современной технологии, будут оставаться полезными даже после очередной «смены технологической парадигмы».

Спецификация типов данных ближе к технологии реализации, нежели большинство других информационных стандартов HL7. Поэтому существует определенная опасность, что описания типов данных окажутся слишком зависимыми от современных технологий реализации.

Большинство стандартов HL7 посвящено сложным деловым объектам. Такие объекты, обладающие большим числом информационных атрибутов, могут быть определены с помощью абстрактного синтаксиса, в котором компоненты объектов описываются в терминах типов данных. Напротив, определение типов данных в терминах абстрактного синтаксиса приносит мало пользы, поскольку компоненты соответствующих синтаксический конструкций сами должны иметь типы данных[1)].

**Почему настоящий стандарт столь цикличен? Почему тип данных «ANY» определен в терминах специализации самого себя?**

Настоящий стандарт должен быть независимым от конкретной реализации, следовательно, абстрактным, не предназначенным для реализации. В этом отношении цикличность не является проблемой, поскольку не вносит никакой неопределенности в содержание спецификации.

**Почему в настоящей спецификации не определен набор примитивных типов данных, с помощью которых комплексные типы данных могут быть определены в терминах абстрактного синтаксиса?**

Технология любой конкретной реализации стандартов HL7 должна опираться исключительно на встроенные типы данных. Поэтому необходимо иметь возможность очень гибкого отображения абстрактных типов данных HL7 на типы данных, встроенные в конкретную технологию реализации. Соответствие спецификации реализуемой технологии (Implementable Technology Specification, ITS) семантической спецификации может быть обеспечено просто с помощью отображения конструкций, используемых этой технологией, на семантику типов данных HL7 Версии 3. С точки зрения семантики не имеет никакого значения, является ли тип данных примитивным или комплексным, и ответ может различаться для разных технологий реализации.

К примеру, настоящий стандарт описывает строку символов как тип данных со многими свойствами (например, кодировка, язык и т. д.). Однако во многих технологиях реализации строки символов являются примитивными типами данных первого класса. Рекомендуется использовать такие нативные типы данных вместо структур, раболепно представляющих все семантические свойства как «компоненты». Настоящий стандарт требует лишь то, что свойства, определенные для значений данных, должны каким-то образом выводиться из выбранного представления, а само представление особого значения не имеет. Не столь важно, используются ли «примитивные» или «комплексные» типы данных, с малым или большим числом «компонентов», определяемых как «поля» или «методы».

Другим примером могут служить такие представления вещественного числа, как десятичное представление, представление с плавающей точкой и представление в виде масштабированного целого числа. Все они являются нативными для разных технологий реализации. Некоторые из них имеют свойства, которыми другие не обладают.

---

[1)] По этой причине Абстрактная синтаксическая нотация версии один (ACH.1), стандартизованная ISO, не может рассматриваться как формализм, пригодный для семантических спецификаций типов данных.

Например, представление в виде масштабированного целого числа обеспечивает фиксированную точность и относительно небольшой диапазон значений. Представление в виде числа с плавающей точкой обеспечивает переменную точность и большой диапазон значений, однако информация о точности при этом потеряна. Представление в виде десятичного числа обеспечивает переменную точность и сохраняет информацию о точности (но при этом медленнее обрабатывается). Семантика типов данных должна не зависеть от таких случайных свойств различных представлений и должна определять существенные свойства, которые могут быть представлены с помощью любой технологии.

**B.1.5 Необходимость в стандарте типов данных HL7**

**Зачем комитету HL7 понадобился собственный стандарт типов данных? Почему нельзя было просто воспользоваться стандартом другого разработчика?**

Как упоминалось в предыдущем разделе, во всех технологиях реализации, предложенных комитетом HL7, используется некоторая система типов данных, но в разных технологиях системы типов данных обладают определенными отличиями. Кроме того, во многих технологиях реализации используемая система типов данных является недостаточно мощной для представления понятий, необходимых в стандартах HL7 прикладного уровня.

Например, понятия физических величин, точности, диапазонов, отсутствующих значений и неопределенности, необходимые в научных вычислениях и при обработке информации в сфере здравоохранения, предусмотрены мало в каких технологиях реализации.

С другой стороны, в технологиях реализации проводятся различия, которые не релевантны абстрактной семантике, например, вещественные числа с фиксированной и плавающей точкой, 8, 16, 32 или 64-битовые целые числа, дата и время и штамп даты и времени.

Для данной спецификации в качестве входной информации использовался целый ряд систем типов данных. К ним относятся системы, используемые в основных языках программирования, включая BASIC, Pascal, MODULA-2, C, C++, JAVA, ADA, LISP и SCHEME. Использовались и системы типов данных, не зависящие от языков программирования, например, Абстрактная синтаксическая нотация версии один (ACH.1), язык описания интерфейсов IDL (Interface Definition Language) и язык объектных ограничений OCL (Object Constraint Language), разработанные организацией OMG (Object Management Group), языки манипулирования данными SQL 92 и SQL 99, не зависящие от языка типы данных, описанные в стандарте ИСО 11404, а также типы данных, описанные в спецификации XML Schema Part 2. Рассматривались также типы данных, используемые в стандартах информатизации здоровья, в том числе HL7 Версии 2.x, стандарты сообщений обмена медицинскими данными, принятые техническим комитетом CEN TC 251, архитектура электронной медицинской карты EHCRA (Electronic Health Record Architecture) и стандарт DICOM.

**B.1.6 Требования**

При разработке типов данных, описанных в настоящей спецификации, учитывался ряд требований, включая следующие:

- требования, связанные с процессом моделирования;
- требования реализации;
- совместимость с другими стандартами типов данных;
- функциональные требования, описанные в других стандартах HL7, где используются типы данных.

Среди них наибольшее внимание уделялось последней группе. Эти типы данных обеспечивают функциональность, требуемую стандартами HL7. Все эти требования не всегда совместимы и в настоящей спецификации можно найти ряд мест, где конкретное решение не слишком оптимально для какой-либо из четырех указанных выше групп требований. В некоторых из этих случаев в подразделе требований перечислены конкретные требования, в соответствии с которыми предложено данное решение. Эти подразделы требований являются не нормативными, а справочными.

**Требование**

В Эталонной информационной модели RIM определен ряд базовых классов, на которых основаны все информационные модели предметных областей. Каждый из этих базовых классов имеет ряд атрибутов, которым присвоены определенные типы данных. Когда эти классы используются (в качестве клонов) в моделях предметных областей, то для уточнения и ограничения использования атрибутов в клонированном классе типы данных атрибутов могут быть заменены другими типами.

В настоящей спецификации должны быть определены правила подобной замены типов данных. В качестве основы для правил замены в ней используется метафора специализации, поскольку она понятна и широко используется как в теории, так и на практике, и такие правила легче воспринимаются и управляются по сравнению с другими альтернативами. Однако использование специализации может привести к результатам, которые некоторым могут показаться непривычными.

**B.1.7 Формы определений типов данных**

В настоящей спецификации используется несколько форм определений типов данных, а именно текстовое описание, диаграммы классов на языке UML, таблицы и формальное определение.

B.1.7.1 Язык формального определения типов данных

Формальное определение типов данных используется, чтобы наиболее однозначным образом описать семантику предлагаемых типов данных. Этот язык определения типов данных детально описан в подразделе B.1.9 «Введение в формальный язык определения типов данных DTDL (Formal Data Type Definition Language)».

Формальные языки обеспечивают возможность составления весьма четких утверждений, которые могут использоваться для представления некоторого формального аргумента в пользу подтверждения или опровержения. Однако лаконичность таких формальных утверждений может также затруднить их понимание человеком. Поэтому все важные выводы из формальных утверждений представлены также в виде предложений на естественном языке.

B.1.7.2 Таблицы свойств

Для удобства ознакомления в начало описаний многих типов данных, приведенных в настоящей спецификации, включены таблицы «основных» свойств. К числу «основных» относится несколько расплывчатая группа тех свойств, которые более похожи на «поля», когда такие типы данных реализуются как записи, или которые наиболее часто используются. Эти таблицы предназначены для облегчения обзора содержания и назначения типов данных. Не требуется, чтобы свойства, перечисленные в этих таблицах, были представлены как поля, и эти таблицы не являются определениями абстрактного синтаксиса.

Каждая строка таблицы свойств описывает одно свойство и содержит следующие элементы:

**1. Имя** — имя свойства, указанное в формальном определении. Для некоторых типов данных поле имени первого свойства в таблице может быть пустым. Это может иметь место для тех типов данных, которые определены как расширения других типов данных и в перечне свойств дочернего типа нет смысла показывать какие-либо свойства родительского типа.

**2. Тип** — тип данных этого свойства.

**3. Определение** — краткий текст, описывающий смысл свойства.

B.1.7.3 Диаграммы на Унифицированном языке моделирования UML (Unified Modeling Language)

Для графического представления связей между типами данных используются диаграммы классов на Унифицированном языке моделирования UML (Unified Modeling Language). Типы данных показаны как классы UML, которым присвоены краткие имена. Порождающие типы показаны как параметризованные классы UML с отношениями реализации, связывающими их экземпляры.

Многие детали объявлений типов данных не могут быть представлены на диаграммах UML. Поэтому для детальной спецификации таких типов данных должно использоваться формальное определение на языке DTDL (Data Type Definition Language — язык описания типов данных).

Некоторые из ограничений в определениях на языке DTDL представлены как ограничения на операции. Если ограничения присутствуют, то они представляют собой утверждения, взятые из спецификации на языке DTDL, которые должны быть истинными.

В диаграммах UML используется стереотип «mixin». Он применяется к параметризованному классу и означает, что этот класс является специализацией типа данных, имя которого является значением параметра T, и в дополнение к собственным свойствам содержит все свойства этого типа данных.

**B.1.8 Обзор типов данных**

Представление типов данных в форме диаграммы классов на языке UML показано на рисунке B.1. Перечень этих типов данных приведен в таблице B.1.

Таблица B.1 — Перечень типов данных в модели RIM

| Имя | Символ | Описание |
|---|---|---|
| DataValue | ANY | Определяет основные свойства каждого типа данных. Это абстрактный тип, означающий, что никакое значение не может быть только значением данных, не принадлежащим никакому конкретному типу. Каждый конкретный тип является специализацией этого общего абстрактного типа DataValue |
| Boolean | BL | Булевский тип представляет значения двузначной логики. Булевское значение может быть или TRUE (истина), или FALSE (ложь), или, как и любое другое значение, может быть пустым (NULL) |
| BooleanNonNull | BN | Тип данных BN представляет собой ограничение типа данных BL, в котором запрещено пустое значение. Этот тип предназначен для использования в тех случаях, когда пустое значение недопустимо |

*Продолжение таблицы В.1*

| Имя | Символ | Описание |
|---|---|---|
| Encapsulated Data | ED | Данные, в основном рассчитанные на чтение человеком или на дальнейшую машинную обработку данных за пределами области применения стандарта HL7. К ним относятся неформатированный или форматированный текст, мультимедийные данные или структурированная информация, определенная другим стандартом (например, XML-подпись). Вместо самих данных значение типа ED может содержать только ссылку на данные (см. описание типа данных TEL). Заметьте, что тип данных ST — это специализация типа данных ED, у которой компонент типа среды ED имеет значение «text/plain» (только текст) |
| Character String | ST | Строковый тип данных, предназначенный для текстовых данных, используемых главным образом для машинной обработки (например, сортировка, запрос, индексирование и т. д.). Используется для имен, символов и формальных выражений |
| Concept Descriptor | CD | Описание понятия, предназначенное для представления любого вида понятий, обычно с помощью кода, определенного в некоторой системе кодирования. Описание понятия может содержать исходный текст или фразу, послужившую основой для кодирования, и один или несколько эквивалентов этого кода в других системах кодирования. Описание понятия может также содержать квалификатор, используемый, к примеру, для описания понятия «левая нога» как посткоординированного термина, образованного от исходного кода «нога» и квалификатора «левый». В исключительных случаях описание понятия может не содержать код, а только исходный текст, описывающий это понятие |
| Coded Simple Value | CS | Кодированные данные в их простейшей форме, в которой указан только код без его значения, предназначенного для вывода на экран. Система кодирования и ее версия определяются контекстом, в котором передается значение типа CS. Тип данных CS используется для значений кодированных атрибутов, имеющих единственный словарный домен, определенный в стандарте HL7 |
| Coded Ordinal | CO | Кодированные данные, образующие систему кодирования, в которой коды упорядочены. Этот тип данных добавляет семантику упорядоченности, так что в моделях, использующих такие словарные домены, могут использоваться элементы, для которых существен порядок терминов в домене |
| Coded With Equivalents | CE | Кодированные данные, состоящие из кодированного значения (типа CV) и необязательно одного или нескольких кодированных значений, принадлежащих другим системам кодирования и описывающих то же самое понятие. Используется при наличии альтернативных кодов |
| Character String with Code | SC | Строка символов, к которой при необходимости может быть добавлен код. Если код присутствует, то должен присутствовать и текст. Код часто берется из местной системы кодирования |
| Instance Identifier | II | Идентификатор, который уникально идентифицирует предмет или объект. Примерами могут служить объектные идентификаторы (ОИД) объектов RIM HL7, номер медицинской карты, идентификатор направления, идентификатор позиции прейскуранта, идентификационный номер транспортного средства и т. д. Идентификаторы экземпляров определяются на основе объектных идентификаторов ветви ИСО |

*Продолжение таблицы В.1*

| Имя | Символ | Описание |
|---|---|---|
| Telecommunication Address | TEL | Номер телефона (голосового или факса), адрес электронной почты или другой указатель ресурса, управляемого телекоммуникационным оборудованием. Адрес указывается в форме Единого указателя ресурсов URL (Uniform Resource Locator), к которому добавлены спецификация времени и коды использования, помогающие установить, какой адрес используется в конкретное время и для каких целей |
| Postal Address | AD | Почтовый, домашний или юридический адрес. Последовательность компонентов адреса, например, улица или почтовый ящик, город, почтовый индекс, страна и т. д. |
| Entity Name | EN | Фамилия, имя, отчество лица, название организации, места или предмета. Последовательность компонентов, например, имя или фамилия, приставка, суффикс и т. д. Примерами могут служить «Джим Боб Уолтон, мл.», «Health Level Seven, Inc.», «Озеро Tahoe» и т. д. Именование объекта может быть простой строкой символов или состоять из нескольких компонентов, например, «Джим», «Боб», «Уолтон» и «мл.»; «Health Level Seven» и «Inc.»; «Озеро» и «Tahoe» |
| Trivial Name | TN | Ограничение именования сущности в форме одной строки, используемой для простого названия предметов и мест |
| Person Name | PN | Тип данных EN, используемый, когда именованной сущностью является физическое лицо. Последовательность компонентов, являющихся фамилией, именем, отчеством, префиксом, суффиксом и т. д. Эти компоненты являются ограничениями компонентов имени сущности, позволяющими указывать только те квалификаторы части, которые применимы к фамилиям, именам, отчествам физических лиц. Поскольку структура типа данных EN навеяна в основном фамилиями, именами и отчествами, эти ограничения весьма минимальны |
| Organization Name | ON | Тип данных EN, используемый для названия организации. Последовательность компонентов названия |
| Integer Number | INT | Целые числа (–1,0,1,2,100,3398129 и т. д.) — точные числа, являющиеся результатами подсчета и нумерации. Целые числа дискретны, набор целых чисел бесконечен, но счетен. Диапазон значений целых чисел не ограничен никаким произвольным пределом. Для типа данных INT определены две причины пустоты (NULL flavor), описывающие положительную и отрицательную бесконечность |
| Real Number | REAL | Вещественные числа. Обычно используются в тех случаях, когда количества измеряются, оцениваются или вычисляются из других вещественных чисел. Типичным представлением вещественного числа служит десятичная запись, в которой число значащих десятичных цифр известно как точность |
| Ratio | RTO | Количество, представленное как отношение числителя к знаменателю. Общие множители в числителе и знаменателе автоматически не сокращаются. Тип данных RTO используется для титров (например, «1:128») и других величин в результатах лабораторных анализов, которые действительно представляют собой отношения. Отношения не являются просто «структурированными числовыми данными», поэтому измерения артериального давления (например, «120/60») не являются отношениями. Во многих случаях вместо типа данных RTO должен использоваться тип данных REAL |

*Окончание таблицы В.1*

| Имя | Символ | Описание |
|---|---|---|
| Physical Quantity | PQ | Размерная величина, выражающая результат измерения |
| Monetary Amount | MO | Денежная сумма, выражающая количество денег в некоторой валюте. Валюты — единицы, в которых указаны денежные суммы в разных экономических регионах. В то время как денежная сумма является единственным видом количества (денег), обменные курсы валют являются переменными. Это принципиальное отличие между физическим количеством и денежными количествами, а также причина того, почему единицы валюты не являются единицами физических величин |
| Point in Time | TS | Величина, указывающая момент времени на оси естественного времени. Момент времени чаще всего представляется как календарное выражение |
| Set | SET | Значение, содержащее другие различные значения ни в каком конкретном порядке |
| Sequence | LIST | Значение, содержащее другие дискретные значения в определенной последовательности |
| Bag | BAG | Неупорядоченная коллекция значений, в которой каждое значение может содержаться более одного раза |
| Interval | IVL | Множество последовательных значений упорядоченного базового типа данных |
| History | HIST | Множество значений данных, которые соответствуют типу исторических данных (HXIT), то есть имеют свойство действительного времени. Историческая информация не ограничена прошлым; в ней могут присутствовать ожидаемые будущие значения |
| Uncertain Value — Probabilistic | UVP | Общее расширение типа данных, используемое, чтобы указать вероятность, отражающую степень доверия поставщика данных к их значениям |
| Periodic Interval of Time | PIVL | Периодически повторяющийся интервал времени. Имеет два свойства: фазу и период. Фаза задает «прототип интервала», который повторяется каждый период |
| Event-Related Periodic Interval of Time | EIVL | Периодически повторяющийся интервал времени, в котором повторение зависит от активности человека в течение дня или от других важных событий, которые связаны со временем, но не имеют точно заданного времени |
| General Timing Specification | GTS | Множество моментов времени, указывающее времена событий и действий, а также шаблоны циклов, которые могут применяться к некоторым видам информации, например, к телефонным номерам (утро, вечер), к адресам (для так называемых «перелетных птиц», которые живут зимой на юге, а летом на севере), а также к часам работы |
| Parametric Probability Distribution | PPD | Общее расширение типа данных, выражающее неопределенность количественных данных с помощью функции распределения и ее параметров. Помимо специфических параметров распределения всегда указываются среднее значение (математическое ожидание) и стандартное отклонение, чтобы обеспечить минимальный уровень интероперабельности на тот случай, если приложение-получатель не в состоянии работать с определенным распределением вероятности |

ГОСТ Р ИСО/HL7 27951—2016

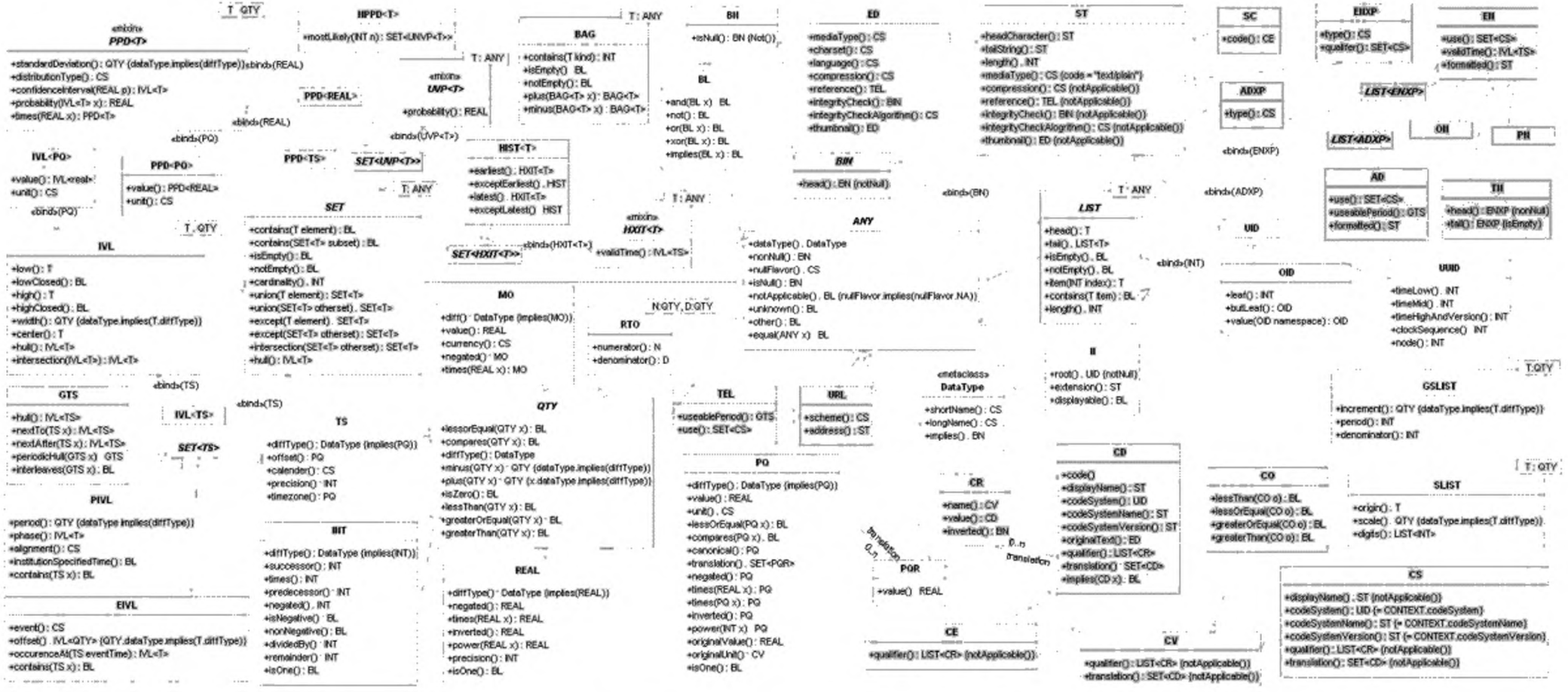


Рисунок В.1 — Представление типов данных в форме диаграммы классов на языке UML

### B.1.9 Введение в формальный язык определения типов данных DTDL (Formal Data Type Definition Language)

Примечание — Это не спецификация прикладного программного интерфейса (API). В то время как данный формальный язык может напоминать некоторые языки программирования или определения интерфейсов, он не предназначен для определения деталей программ и других средств реализации. Формальные определения являются нормативной частью настоящей спецификации, но данный конкретный язык не требует реализации или использования в системах, соответствующих стандарту. От них не требуется также реализации или использования всех семантических свойств. Внутренняя деятельность систем, способ реализации типов данных, их функциональность и предоставляемые сервисы полностью выходят за область применения настоящего стандарта. Формальное определение только указывает смысл значений данных с помощью утверждений, как с точки зрения теории эти значения связаны с собой и какими свойствами обладают.

Этот формальный язык определения типов данных[1)] задает:
- полное и краткое имя данных;
- именованные значения для полностью перечисляемого расширения;
- семантические свойства, унарные, бинарные свойства, а также свойства более высокого порядка;
- инварианты, то есть ограничения свойств;
- допустимые преобразования типа;
- синтаксис строкового представления значения (если таковое имеется).

Определение типа данных осуществляется в два этапа. Сначала делается объявление типа данных. В объявлении указываются имя нового типа данных и список имен, типов и сигнатур его семантических свойств. Это лишь объявление, а не определение типа. Определение содержит логические утверждения о том, что всегда истинно для значений этого типа и их свойств (инвариантные утверждения).

B.1.9.1 Объявление

Объявление каждого типа данных начинается с ключевого слова **type**. Например, ниже показан заголовок объявления типа данных Boolean, имеющего краткое имя BL и являющегося специализацией типа данных ANY[2)]:

```
type Boolean alias BL specializes ANY
   values{true, false}
{
   BL  not;
   BL  and{BL x};
};
```

Объявление типа данных Boolean содержит конструкцию **values**, специфицирующую полный набор значений типа Boolean (его расширение) в виде именованных сущностей. Эти именованные значения являются также допустимыми строковыми литералами. Поскольку никакой другой тип данных из числа определенных в настоящей спецификации не имеет конечное множество значений, то конструкция **values** уникальна для типа данных Boolean. В размеченном формальном языке имена значений выделяются курсивом.

---

1) Используемый здесь язык определения данных возник в результате экспериментов и опыта применения различных альтернатив, включая таблицы определения типов данных и язык определения интерфейса IDL (Interface Definition Language), предложенный организацией OMG (Object Management Group). Недостаток таблиц определения типов данных в том, что они дают неверное представление, что настоящий документ является спецификаций абстрактного синтаксиса, а не семантики. Аналогично недостатком IDL является то, что его применение дает неверное впечатление, что данный документ является определением прикладного программного интерфейса API (application programming interface).

Получившийся язык определения типов данных многое позаимствовал от свойств и стиля языка IDL, языка объектных ограничений OCL (Object Constraint Language), языков программирования JAVA, C++, а также от средств генерации разборщиков синтаксиса LEX и YACC. Заимствованные свойства были объединены и дополнены в целях получения ровно того, что требовалось для данной спецификации типов данных и создания минимального и самодостаточного языка. Кроме того, коль скоро основным назначением данного языка является определение типа данных, была сделана попытка обойтись без встроенных типов данных.

2) Нетрудно видеть, что ключевое слово type указано вместо ключевого слова keyword, используемого при описании интерфейсов на языках IDL и Java и классов на языках C++ и Java. Конструкция alias уникальна для настоящей спецификации и введена для того, чтобы можно было добавить весьма краткую мнемонику к более описательным именам. Конструкция specializes предпочтительнее двоеточия, используемого в языках C++ и IDL, поскольку ее значение более очевидно.

Блок текста в фигурных скобках, следующий за заголовком, содержит объявление семантических свойств, которыми обладает каждое значение типа данных. Объявление свойства завершается точкой с запятой, а другая точка с запятой, стоящая после закрывающей фигурной скобки, завершает объявление типа данных.

Объявление свойства включает в себя следующие компоненты (слева направо): (1) тип данных домена значений свойства, (2) имя свойства и (3) необязательный список аргументов. Этот список окружен круглыми скобками, внутри которых указана последовательность объявлений аргументов. Объявление аргумента содержит имя типа данных и имя аргумента. Для семантических свойств, не имеющих аргументов, пустой список не указывается[1)].

Конструкция **specializes** означает (а) наследование свойств от рода к виду, (б) возможность подстановки значений видового типа в переменные родового типа. Эта конструкция может включать в себя определение дополнительных свойств и спецификацию ограничений свойств, наследуемых специализированным типом.

Приведем следующий пример наследования: коль скоро тип данных CS является специализацией типа данных CD, а тип данных CD имеет свойство code, то тип данных CS также имеет это свойство code, даже если свойство isNull не указано явно в объявлении свойства BL. Справедлива также следующая возможность подстановки: если свойство имеет тип данных CD, а тип данных CS является специализацией типа данных CD, то значение этого свойства может иметь тип данных CS. Другими словами, возможность подстановки означает, что все значения типа данных CS являются также значениями типа данных CD[2)].

Объявление **type** может иметь в качестве квалификаторов ключевые слова **abstract** (абстрактный), **protected** (защищенный) или **private** (приватный). Абстрактный тип представляет собой тип данных, у которого ни одно значение не может иметь этот тип данных и при этом не принадлежать к его конкретной специализации. Защищенный тип представляет собой тип данных, который используется внутри данной спецификации, но при этом не может быть присвоен никакому свойству вне этой спецификации. Приватный тип представляет собой внутреннюю абстракцию «подсказки». Он определяется только для целей определения некоторого аспекта семантики типа данных, но не используется как тип другого защищенного или публичного свойства[3)]. (Квалификатор private используется здесь с единственной целью. Приватные типы определены только для формального определения других типов данных и ни в каком виде не используются вне этой спецификации.)

B.1.9.2 Инвариантное утверждение

Объявление семантических свойств, их имен, типов данных и аргументов представляет собой только ключ к пониманию назначения типа данных. Истинное определение заключается в инвариантных утверждениях, являющихся логическими утверждениями, истинными в любое время.

В настоящей спецификации инвариантные утверждения представлены с использованием формального синтаксиса, а также на естественном языке. Преимуществами формального синтаксиса являются однозначность его интерпретации и строгое типизирование. А утверждения на естественном языке более доступны для понимания, особенно тем лицам, кто не натренирован в чтении утверждений на формальных языках.

Формальный синтаксис помогает придать убедительность настоящей спецификации. Однако в некоторых случаях полная семантика типа не может быть полностью выражена с помощью таких инвариантных утверждений. Сочетание естественного и формального языка помогает сделать настоящую спецификацию более точной.

Инвариантные утверждения образуются с помощью ключевого слова **invariant**, которое объявляет одну или несколько переменных в том же формате, что у списка аргументов свойства. Инвариантное утверждение может содержать конструкцию **where**, ограничивающую аргументы для всего тела утверждения. Тело инвари-

---

1) Обратите внимание, что понятия входных и выходных аргументов, используемые в языке IDL, а также понятия возвращаемых значений и исключений, используемые в языках IDL, JAVA и C++, неуместны для настоящей спецификации. Семантика типов данных не имеет отношения к вызовам процедур, передаче параметров или нормального и аварийного возвращения управления из тела процедуры. Вместо этого каждое семантическое свойство можно концептуально рассматривать как функцию, отображающую значение и необязательно аргумента на другое значение. Это отображение не «вычисляется» и не «генерируется», оно существует на логическом уровне и для актуализации отображения не требуется «вызывать» такую функцию.

2) Ограничивающий аспект специализации заслуживает разъяснений. Обычно утверждается, что наследование на должно отказываться от каких-либо свойств, определенных для рода. Это действительно так для ограничения, поскольку при этом от свойства не отказываются, но ограничивают его значения меньшим множеством. Но если свойство родительского класса может быть пустым (NULL), то ограничение может состоять в том, что оно всегда пустое. В любом случае логически ограничение является специализацией с наследованием и возможностью подстановки.

3) Обратите внимание, что значение ключевого слова protected несколько отличается от квалификаторов доступности (public, package, protected, private), используемых в языках JAVA и C++. Оно используется здесь не как признак скрытия информации об этом типе или исключения свойств защищенного типа из доступа вне настоящего «пакета» спецификаций. Его надо рассматривать как настоятельную рекомендацию не объявлять атрибуты или другие свойства таких защищенных типов. Защищенные типы должны использоваться как «встроенные» в другие типы данных. Защищенный тип непосредственно доступен внутри «обертки», в которую он встроен. Понятие «делегированных свойств» отсутствует.

антного утверждения заключено в фигурные скобки. Оно содержит список высказываний, которые все должны быть истинными.

```
invariant(BL x)
      where x.nonNull {
   x.and(true).equal(x);
};
```

Инвариантное утверждение имеет семантику логического предиката с квантором общности («для всех»).

Приведенное выше инвариантное утверждение может быть прочитано на естественном языке следующим образом: «Для всех значений x типа Boolean, где x не является пустым, справедливо высказывание «x AND *true* равно x». Все свойства должны быть именованными, чтобы высказывания можно было читать как предложения на естественном языке[1)].

Если для инвариантного утверждения аргументы не требуются, то список аргументов указывать в нем не надо.

```
invariant {
   true.not.equal(false);
   false.not.equal(true);
};
```

B.1.9.2.1 Выражение высказывания

Высказывания, включенные в инвариантные утверждения, представляют собой выражения, аргументами которых являются семантические свойства определяемых типов данных. Выражение высказывания должно иметь значение типа Boolean («true» или «false»)[2)]. Никакие примитивные типы данных или операции не существуют до определения типов данных. Единственными предопределенными свойствами языка выражений высказываний являются[3)]:

- строки символов, представляющие словесные выражения языка определения типов данных;
- понятие справедливости высказывания (*true*) или его ложности (*false*);
- инвариантное утверждение: invariant(...) where ... {...};
- выражение квантора общности в форме forall (...) where ... {...}; является синонимом инвариантного утверждения;
- выражения квантора существования в форме exists (...) where ... {...};
- явная конъюнкция (логический оператор AND) между утверждениями: утверждение_1; утверждение_2; ... утверждение_n, разделенными точками с запятой;
- переменные и объявления в списке аргументов инвариантного утверждения;
- ссылка на свойство с помощью точки: x.property;
- явное и неявное преобразование типов: (T)x;

---

1) Синтаксис и семантика инвариантного учреждения похожи на конструкцию «inv» в языке OCL. Однако этот язык не используется в настоящей спецификации по следующим причинам: (1) стиль синтаксиса языка OCL напоминает Smalltalk, отличающихся от стиля определения типов данных в языках C++/Java; (2) в языке OCL предусмотрено много примитивных конструкцию и типов данных, чего настоящий стандарт старается по возможности избежать; (3) богатство примитивных конструкций в языке OCL является одной из причин его сложности, не требуемой в настоящей спецификации.

2) Эта конструкция в определенном отношении является циклической; она исходит из существования типа данных Boolean, хотя этот тип данных сам определен таким же образом, как и другие типы данных. Кроме того, поскольку данный язык определения типов данных представлен в виде строк символов, понятие строки символов должно существовать до определения строкового типа данных character string. Таким образом, эти два типа, character string и Boolean, являются особыми, но на первый взгляд они определены аналогично другим типам данных. Поскольку не предполагается, что данная спецификация типов данных будет реализована, такая цикличность не представляет собой особую проблему. Даже если бы этот язык предполагалось реализовать, можно было бы использовать технологию «самозагрузки», которая вполне обычна, к примеру, в компиляторах, которые компилируют сами себя.

3) Большинство этих синтаксических свойств навеяно языком JAVA: использование списков аргументов, фигурные скобки для окаймления блоков, точка с запятой для завершения утверждения и точка для ссылки на свойства значения. Удвоенное двоеточие «::», используемое в языках C++ и IDL для различения ссылки на члена и ссылки на значение, здесь не используется (как и в языке Java). В отличие от языка Java и ближе к языкам C++ и IDL каждое утверждение завершается точкой с запятой, в том числе декларации типа. Неявное преобразование типов также позаимствовано у языка C++.

- круглые скобки для переопределения приоритетов операторов преобразования и разрешения свойств: (T)x.property по сравнению с ((T)x).property.

B.1.9.2.2 Выражения с вложенными кванторами

По аналогии с инвариантными выражениями внутри выражения высказываний могут быть указаны вложенные кванторы. В действительности квантор общности, записанный с помощью ключевого слова forall, является не чем иным, как инвариантным выражением. Как показано в следующем примере, квантор общности может использоваться во вложенном выражении, если этого требует сложность проблемы:

```
invariant(SET<T> x, y)
      where x.nonNull {
   x.subset(y).equal(
      forall(T element) where x.contains(element) {
         y.contains(element);
         });
};
```

Квантор существования имеет то же значение, что и в общей логике высказываний. Например, следующее инвариантное выражение означает: «Значения x и y типа SET (множество) пересекаются в том и только том случае, когда существует элемент e, принадлежащий обоим множествам x и y».

```
invariant(SET x, y)
      where x.nonNull {
   x.intersects(y).equal(
      exists(T e) {
         x.contains(e);
         y.contains(e);
         });
};
```

В кванторе существования может использоваться конструкция where, однако нет разницы в том, сделано ли высказывание в конструкции where или в теле квантора существования. Напротив, в кванторах общности конструкция where ослабляет высказывание, поскольку при наличии этой конструкции тело применяется только к тем значениям, которые удовлетворяют критерию, указанному в конструкции where.

B.1.9.3 Преобразование типов

В настоящей спецификации определены некоторые допустимые преобразования типов данных. Например, для типов данных ST (строка символов) и ED (инкапсулированные данные) существует пара преобразований. Это означает, что если ожидается значение типа ED, но получено значение типа ST, то можно преобразовать значение типа ST в значение типа ED[1].

Определены три вида преобразования типов данных: повышающее приведение (promotion), понижающее приведение (demotion) и литералы строк символов. Преобразования типов могут быть явными и неявными. Неявное преобразование типов имеет место, если ожидается значение определенного типа (например, в аргументе выражения), но в действительности получено значение другого типа. Если тип полученного значения допускает преобразование в ожидаемый тип, то преобразование должно быть сделано неявным образом.

Примечание — В спецификации реализуемой технологии должно быть указано, каким образом поддерживается неявное преобразование типов данных. В некоторых технологиях неявное преобразование поддерживается, в других нет; в любом случае должны быть описаны правила обработки, указывающие, как такие преобразования реализуются.

Явное преобразование можно задать в выражении утверждения, указав имя целевого типа данных в скобках перед преобразуемым значением. Например, ниже показано явное преобразование типа данных в конструкции where инвариантного выражения:

```
invariant(ED x)
   where ((ST)x).nonNull { ... };
```

Преобразование типа имеет более низкий приоритет по сравнению с точкой, используемой для указания свойства. Так, «(T)a.b» задает преобразование значения свойства b переменной a в тип данных T, в то время как

[1] Это означает, что если некто ожидает значение типа ED, но вместо него получил значение типа ST, то он может преобразовать значение типа ST в значение типа ED.

«((T)a).b» задает сначала преобразование значения переменой a в тип данных T, а затем ссылку на свойство b результата преобразования.

Неявное преобразование типа данных в выражениях высказываний осуществляется при возможности. Пусть объявлено, что формальный аргумент свойства имеет тип данных T. Если в выражении он используется как фактический аргумент типа U и при этом тип данных U не является расширением типа T, то в случае, если для типа данных U определено преобразование в тип данных T, выполняется преобразование из типа данных T в тип данных U.

B.1.9.3.1 Понижающее приведение

Понижающее приведение (demotion) представляет собой преобразование с потерей информации. В общем случае это означает, что более сложный тип преобразуется в более простой.

Примером понижающего приведения может служить преобразование значения, имеющего тип данных интервала (IVL), в простое количество (тип данных QTY), например, в центр интервала. В языке определения типов данных понижающее приведение объявляется с помощью ключевого слова **demotion** и имени результирующего типа данных:

```
type Interval alias IVL {
   ...
   demotion  QTY;
   ...
};
```

В спецификации понижающего приведения должно быть указано, какая информация теряется и каковы главные последствия потери информации.

B.1.9.3.2 Повышающее приведение

Повышающее приведение (promotion) представляет собой преобразование, при котором генерируется новая информация. В общем случае это означает, что более простой тип преобразуется в более сложный.

Например, можно преобразовать значение количества, имеющее тип данных (QTY), в значение интервала (тип данных IVL). Однако у типа данных IVL имеются дополнительные семантические свойства по отношению к типу данных QTY, а именно нижняя и верхняя граница. Таким образом, преобразование из типа данных QTY в тип данных IVL является повышающим приведением. Дополнительным свойствам типа данных QTY, отсутствующим у типа данных IVL, должны быть присвоены новые значения (значения по умолчанию или вычисляемые значения). В спецификации повышающего приведения должно быть указано, каковы будут новые значения и как они будут генерироваться.

Повышающее приведение типа данных QTY к типу данных IVL объявляется с помощью ключевого слова promotion и имени результирующего типа данных:

```
type Quantity alias QTY {
   ...
   promotion  IVL;
   ...
};
```

Обычно повышающее приведение определяется для преобразования более простого типа данных в более сложный тип. Обычно также простой тип объявляется в настоящем стандарте раньше более сложного. Объявление всех повышающих приведений в простом типе потребовало бы ссылок вперед по тексту, что неудобно читателю. Поэтому альтернативный синтаксис разрешает определять повышающее приведение в объявлении более сложного типа. Оно обозначается с помощью указания имени типа, к которому применяется повышающее приведение, в списке аргументов после имени результирующего типа:

```
type Interval alias IVL {
   ...
   promotion  IVL (QTY x);
   ...
};
```

B.1.9.4 Литеральная форма

Литерал представляет собой представление значения данных в виде строки символов. Литералы определены для многих типов данных. Литерал представляет собой преобразование типа из строкового типа данных ST и обратно с помощью специального определенного синтаксиса.

Не каждое преобразование в тип данных ST и обратно является литеральным преобразованием. Литерал типа данных должен обеспечивать представление всего множества значений этого типа данных, в то время как другие преобразования в тип данных ST и обратно могут отображать только меньшее подмножество значений преобразуемого типа данных.

Целью использования литералов является представление значений в краткой человекочитаемой форме. Например, литеральные представления целых чисел (тип данных INT) и вещественных чисел (тип данных REAL) являются строками, состоящими из знака числа, цифр, необязательной десятичной точки и т. д. Более важные типы интервалов (IVL<REAL>, IVL<PQ>, IVL<TS>) имеют литеральные представления наподобие «<5», означающего «меньше 5», что гораздо удобнее для чтения, нежели полностью структурированная форма интервала. Для некоторых более сложных типов данных, например интервалов, общей спецификации периодичности и параметрического распределения вероятности литеральная форма представления значений может оставаться единственной до тех пор, пока пользователь не привыкнет к используемой концептуализации.

Каждое литеральное преобразование имеет свой собственный синтаксис (грамматику), который может быть не очень простым для компьютерной обработки[1)].

Примечание — Спецификация реализуемой технологии, использующая строковое представление значений этих абстрактных типов данных, может использовать, а может и не использовать литеральные представления, приведенные в настоящем стандарте. Ожидается, что спецификация реализуемой технологии на языке XML будет использовать не все, но некоторую часть определенных здесь представлений.

B.1.9.4.1 Объявление

На языке определения типов данных литерал объявляется как свойство типа данных, используя ключевое слово literal, за которым следует имя типа данных ST, поскольку литерал представляет собой преобразование в тип данных ST и обратно:

```
type IntegerNumber alias INT {
   ...
   literal  ST;
   ...
};
```

B.1.9.4.2 Определение

Фактическое определение литеральной формы осуществляется вне тела объявления типа данных, используя атрибутивную грамматику. Такая грамматика определяет как семантику, так и синтаксис структур языка. Определенный в ней синтаксис по существу является формой Бэкуса — Наура BNF (Backus-Naur-Form)[2)].

К примеру, рассмотрим следующее простое определение типа данных порядковых чисел (положительных целых чисел). Определение этого типа данных зависит только от типа данных Boolean (BL) и содержит объявление литерала строкового типа:

```
type CardinalNumber alias CARD {
   BL       isZero;
   BL       equal(ANY x);
   CARD     successor;
   CARD     plus(CARD x);
   CARD     timesTen;
   literal  ST;
};
```

Ниже полностью представлены синтаксис и семантика литерала, а затем его детальное описание.

```
CARD.literal ST {
   CARD : CARD digit  { $.equal($1.timesTen.plus($2); }
        | digit       { $.equal($1); };
```

---

[1)] Наличие различных грамматик литералов не означает, что они будут объединены в одну общую грамматику для представления значений в стандартах HL7. Хотя и можно предпринять попытки разрешения неоднозначности для литералов разных типов, они могут оказаться опасными, поскольку некоторые из неоднозначностей останутся. Например, значение «1.2» является допустимым литералом как для объектного идентификатора (тип данных OID), так и для вещественного числа.

[2)] Используемый здесь вариант формы BNF близок к тем, что используются в языках синтаксического разборщика YACC и генератора лексического анализатора LEX, но он упрощен и сделан совместимым с синтаксисом и стилем объявлений, принятыми в данном языке определения типов данных. Отличия состоят в том, что все символы имеют ровно один атрибут, их значение строго типизировано с использованием одного из определенных типов данных. Тип каждого символа объявлен перед определением символа (например, INT digit : «0» | «1» | ... | «9»;). Начальный символ представляет собой не имя, а тип (например, INT : digit | INT digit;). Имя типа данных может служить именем символа, означающего литерал этого типа данных.

```
    CARD digit : "0"    { $.isZero; }
               | "1"    { $.equal(0.successor); }
               | "2"    { $.equal(1.successor); }
    ...
               | "8"    { $.equal(7.successor); }
               | "9"    { $.equal(8.successor); }
};
```

Каждое синтаксическое правило состоит из имени символа, двоеточия и определения символа (называемого продукцией). Продукция представляет собой последовательность символов. Эти другие символы либо также определены в грамматике, либо являются терминальными символами, представляющими собой строки символов, заключенные в двойные кавычки, либо шаблоны строк (называемые регулярными выражениями). Таким образом, запись

```
CARD : CARD digit
     | digit;
```

означает, что любой символ порядкового номера является либо порядковым номером, за которым следует цифра, либо просто цифрой. Вертикальная черта означает логическое «ИЛИ» (OR). Синтаксическое правило завершается точкой с запятой.

Каждый символ имеет ровно одно значение определенного типа данных. Тип данных значения символа объявляется в его определении. Запись

```
CARD digit : "0"
           | "1"
           | "2"
           | ...
           | "8"
           | "9";
```

означает, что символ digit имеет значение типа CARD. Начальный символ является именем типа данных, отдельного имени не требуется.

Семантика литерального выражения описана семантическими правилами, заключенными в фигурные скобки. Эти правила задаются для каждой определенной продукции символа:

*символ* : *продукция1* { *правило1* } | *продукция2* { *правило 2* } | ... | *продукцияn* { *правило n* };

Семантическое правило представляет собой простой список булевских выражений высказывания, разделяемых точкой с запятой. Эти выражения имеют тот же вид, что и выражения высказывания, используемые в инвариантных выражениях. Однако в семантическом правиле могут быть определены специальные переменные, имена которых начинаются со знака доллара (например, $, $1, $2, $3, ...). Одинокий знак $ означает значение текущего определенного символа, а $1, $2, $3 и т. д. означают значения частей продукции, ассоциированной с семантическим правилом. Например, в записи

```
CARD : CARD digit   { $.equal($1.timesTen.plus($2); }
     | digit        { $.equal($1); };
```

с первой продукцией «CARD digit» связано семантическое правило, которое гласит: «значение определенного символа ($) равно значению $1 первого символа CARD, умноженному на 10 (timesTen), сложенному (plus) со вторым символом digit»[1].

В качестве терминального символа может быть указан шаблон строки, так называемое регулярное выражение. Здесь использован классический синтаксис регулярных выражений, придуманный Aho и используемый в AWK, LEX, GREP и PERL. Регулярные выражения указывают между двумя косыми чертами /.../. В регулярном выражении

[1] Следует учесть, что свойство equal (равенство), определенное для всех типов данных (см. equal) является отношением, а именно проверкой на равенство, а не оператором присваивания. Значению нельзя присвоить значение. В отличие от анализаторов YACC и LEX, данный язык определения типов данных является чисто декларативным и понятие присваивания в нем отсутствует. Поэтому грамматические правила определяют литеральные выражения как для разбора, так и для конструирования.

каждый символ, кроме [ ] ^ $ . / : ( ) \ | ? * + { }, совпадает сам с собой. Другие символы, фактически используемые в настоящей спецификации, приведены в таблице В.2.

Таблица В.2 — Специальные символы для регулярных выражений

| Шаблон | Определение |
|---|---|
| [ ... ] | Описывает класс символов. Например, /[A-Za-z]/ совпадает с прописными и строчными буквами английского алфавита |
| [^ ...] | Задает исключение из класса символов. Например, /[^BCD]/ совпадает с любым символом, кроме B, C и D |
| ...? | Предшествующий шаблон не обязателен. Например, /ab?c/ совпадает и с «ac», и с «abc» |
| ...* | Предшествующий шаблон может повторяться от нуля до любого числа раз. Например, /ab*c/ совпадает с «ac», «abc», «abbc», «abbbc» и т. д. |
| ...+ | Предшествующий шаблон может повторяться от одного до любого числа раз. Например, /ab+c/ совпадает с «abc», «abbc», «abbbc», но не с «ac» |
| ... {n,m} | Предшествующий шаблон может повторяться от *n* до *m* раз, где *n* и *m* порядковые числа, удовлетворяющие условию $0 < n < m$. Например, /ab{2,4}c/ совпадает с «abbc», «abbbc» и «abbbbc» |
| ... \| ... | Совпадение может быть с любым из шаблонов, разделенных вертикальной чертой. Например, /ab\|cd/ совпадает с «abd» и«acd», но не с «abcd» |
| ( ... ) | Шаблон в скобках используется в описанных выше операторах как единое целое. Например, /a(bc)*/ совпадает с «a», «abc», «abcbc», «abcbcbc» и т. д. |
| ... : ... | Совпадение с левым шаблоном будет в том случае, если за ним следует правый шаблон, но при этом правый шаблон не участвует в проверке совпадения. Например, /ab:c/ совпадает с «abc», но не с «ab», однако значением совпадающего символа будет «ab», а «c» остается для следующего символа. Двоеточие представляет собой небольшую модификацию традиционной косой черты, но косая черта также традиционно используется для завершения всего шаблона и может также встретиться как совпадающий символ — три разных значения одного символа были бы уже слишком |
| ... \ ... | Буквальное совпадение со следующим символом, то есть устраняет любое специальное значение этого символа. Например, /a\+b/ совпадает с «a+b» |
| ... \/ ... | Совпадение с косой чертой как с символом. Например, /a\/bc/ совпадает с «a/bc» |

В.1.9.5 Параметризованные типы данных

Параметризованные типы данных имеют неполные определения. Эта неполнота обозначается наличием одного или нескольких параметров в определении типа. Обычно в качестве параметров выступают другие типы данных. С помощью параметров параметризованный тип может объявить семантические свойства других не полностью определенных типов данных. Например, параметризованный тип данных Interval объявлен с параметром *T*, значение которого может быть любым типом данных физической величины QTY. Его компоненты *low* и *high* объявлены как имеющие тип данных *T*:

```
template<QTY T>
type Interval<T> alias IVL<T> {
   T  low;
   T  high;
};
```

Создание экземпляра параметризованного типа данных завершает его определение. Например, для создания экземпляра интервального типа данных (Interval) необходимо указать базовый тип данных интервала. Это делается с помощью связывания параметра *T*. Чтобы создать экземпляр интервала целых чисел, надо связать параметр *T* с типом данных Integer. После этого неполный тип данных Interval становится полным типом данных *Interval of Integer* (интервал целых чисел).

Например, в следующем определении типа данных MyType объявляется свойство с именем «multiplicity» (кратность), имеющее тип данных интервала порядкового типа данных, использованного в предыдущих примерах:

```
type MyType alias MT {
   IVL<CARD>  multiplicity;
};
```

B.1.9.5.1 Параметризованные коллекции

В настоящей спецификации активно используются параметризованные типы данных для коллекций. Среди них наиболее важными являются:

**Set** (SET<T>) — множество, содержащее не упорядоченные и не повторяющиеся элементы.

**Sequence** (LIST<T>) — последовательность, представляющая собой коллекцию значений, имеющих произвольный, но конкретный порядок. У последовательности есть голова и хвост, где головой является элемент, а хвостом — последовательность без своей головы.

**Interval** (IVL<T>) — интервал, представляющий собой непрерывное подмножество упорядоченного типа данных.

Эти и другие параметризованные типы данных полностью описаны в B.1.9.5 «Параметризованные типы данных». Эти параметризованные типы данных и их свойства используются в данной спецификации, начиная с ранних разделов. Для лучшего понимания настоящей спецификации необходимо иметь определенные знания о множестве, последовательности и интервале; когда речь пойдет об использовании параметризованного типа для определения другого типа данных, рекомендуется обратиться к пункту B.1.9.5 «Параметризованные типы данных».

B.1.9.5.2 Расширения параметризованного типа данных

Расширения параметризованного типа данных являются параметризованными типами данных с одним типом параметра, специализируемым параметризованным типом. На формальном языке определения типов данных для специализации параметризованного типа используется следующий шаблон:

```
template<ANY T> type GenericTypeExtensionName specializes T {
   ...
};
```

Такие расширения параметризованного типа наследуют свойства от своего базового типа и добавляют к ним некоторые специфические особенности. Поскольку расширение параметризованного типа является специализацией базового типа, то значение, имеющее расширенный тип данных, может быть использовано вместо значения, имеющего базовый тип данных[1)].

Примечание — Значения, имеющие расширенный тип, могут быть заменены значениями, имеющими его базовый тип. Однако спецификация реализуемой технологии может наложить некоторые ограничения на то, какие расширения она допускает. В частности, расширения не должны определяться для тех компонентов, в которых содержатся значения свойств значения данных. Таким образом, для любого типа данных может быть указана аннотация вне спецификации типа данных, но спецификация реализуемой технологии может не обеспечивать возможность указания аннотации значения свойства значения данных.

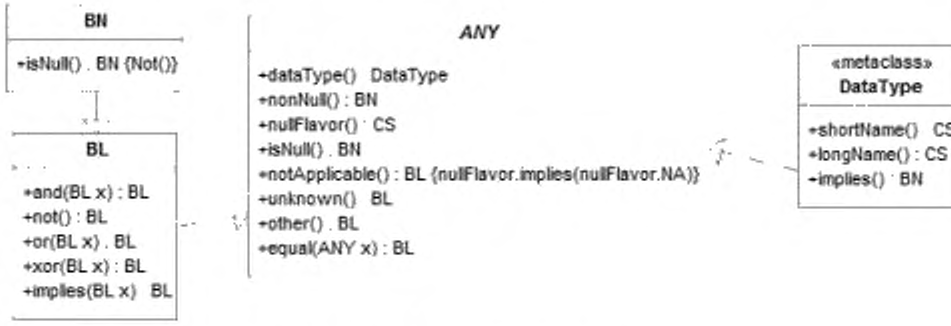


Рисунок B.2 — Фундаментальные типы данных

**B.1.10 Соответствие**

Если приложение получает или разбирает экземпляр, который не является допустимым по отношению к настоящей спецификации, то получателю разрешается отклонить этот экземпляр таким способом, который он сочтет наиболее приемлемым, однако это не является обязательным. Следует обратить внимание, что некоторые другие стандарты HL7 или такие артефакты, как объявление соответствия, могут накладывать дополнительные ограничения на поведение получателя в таких случаях.

[1)] Расширения параметризованного типа данных иногда называют «смесями» (mixins), поскольку их эффект состоит в примешивании некоторых свойств к уже определенному типу данных.

**B.1.11 Тип данных DataValue (ANY)**

Этот тип данных определяет базовые свойства любого типа данных. Он является абстрактным, то есть значение не может иметь этот тип данных и не принадлежать к какому-либо конкретному типу. Каждый конкретный тип данных является специализацией общего абстрактного типа данных DataValue.

```
abstract type DataValue alias ANY {
   TYPE  dataType;
   BN    nonNull;
   CS    nullFlavor;
   BN    isNull;
   BL    notApplicable;
   BL    unknown;
   BL    other;
   BL    equal(ANY x);
};
```

B.1.11.1 Свойство dataType TYPE

Определение: это свойство отражает тот факт, что каждое значение данных неявно несет информацию о своем типе данных. Таким образом, получив значение данных, можно запросить его тип данных.

```
invariant(ANY x) {
   x.dataType.nonNull;
};
```

B.1.11.2 Свойство nonNull: BN

Определение: указывает, что данное значение является непустым, то есть допустимым для данного типа данных.

```
invariant(ANY x) {
   x.isNull.equal(x.nonNull.not);
};
```

Если свойство, атрибут модели RIM или поле сообщения объявлены обязательными и непустыми (mandatory), то любое непустое значение типа данных, к которому принадлежит свойство, должно иметь непустое значение этого свойства. Другими словами, поле не может быть пустым, коль скоро его контейнер (объект, сегмент и т. д.) должно иметь непустое значение.

B.1.11.3 Свойство isNull: BN

Определение: указывает, что значение является исключительным, то есть пустым (NULL). Пустота означает, что информация не существует, или недоступна, или не может быть представлена в нормальном наборе значений этого типа данных.

Каждый элемент данных либо имеет допустимое значение, либо считается пустым. Если (и только если) оно пусто, то свойство nullFlavor предоставляет более детальную информацию о том, каким образом и почему у этого свойства нет правильного значения.

```
invariant(ANY x) {
   x.isNull.equal(x.nullFlavor.implies(NI));
};
```

B.1.11.4 Свойство nullFlavor: CS

Определение: если значение является исключительным (NULL), то это свойство указывает, каким образом и почему подходящая информация отсутствует.

```
invariant(ANY x) {
   x.nonNull.equal(x.nullFlavor.isNull);
};
```

Таблица B.3 — Словарный домен свойства nullFlavor (причина пустоты)

| Код | Имя | Определение |
|---|---|---|
| NI | NoInformation | Отсутствует какая бы то ни было информация, которую можно вывести из данного исключительного значения. Это наиболее общее исключительное значение, оно также используется по умолчанию |

*Окончание таблицы В.3*

| Код | Имя | Определение |
|---|---|---|
| OTH | other | Фактическое значение не является элементом домена значений переменной (например, в используемой системе кодирования данное понятие отсутствует) |
| NINF | negative infinity | Отрицательная бесконечность чисел |
| PINF | positive infinity | Положительная бесконечность чисел |
| UNK | unknown | Правильное значение имеется, но оно неизвестно |
| ASKU | asked but unknown | Информация запрошена, но ответ не получен (например, пациенту задали вопрос, а ответ он не знает) |
| NAV | temporarily unavailable | Информация в данное время недоступна, но может стать доступной позже |
| NASK | not asked | Эта информация не запрашивалась (например, пациенту вопрос не задавался) |
| TRC | trace | Содержание отлично от нуля, но слишком мало, чтобы можно было узнать его количество |
| MSK | masked | Информация об этом элементе имеется, но не предоставлена отправителем по причине безопасности, конфиденциальности и т. д. Для получения доступа к этой информации могут существовать альтернативные механизмы. Хотя детали информации и не раскрываются, предоставление такой причины пустоты может привести к нарушению конфиденциальности информации. Такая причина пустоты может быть указана в ситуации, когда получателя необходимо информировать о том, что информация существует, не представляя саму информацию |
| NA | not applicable | В данном контексте правильное значение не существует (например, последний менструальный период у мужчины) |
| NP | not present | Значение не представлено в сообщении. Такая причина пустоты определена только для сообщений, а не для прикладных данных! Все значения, не представленные в сообщении, должны быть заменены на правильное значение по умолчанию или иметь признак (NI) (нет информации), используемый как умолчание для всех умолчаний |

Причины пустоты представляют собой общее доменное расширение всех нормальных типов данных. Обратите внимание на отличие домена значений любого типа данных от словарного домена кодированных типов данных. Словарный домен является доменом кодированных значений, но не все домены значений являются словарными доменами.

Причина пустоты «other» (другое) используется, когда фактическое значение не принадлежит требуемому домену значений. Такое может быть, скажем, когда значение не удовлетворяет некоторым слишком строгим ограничениям (например, возраст должен быть меньше 100 лет).

Примечание — Причины пустоты применимы к любому свойству значения данных или атрибуту объекта более высокого уровня. Если различия в причинах пустоты не существенны, то причины пустоты можно не отражать в спецификации реализуемой технологии. Если иное не указано в настоящем стандарте, то спецификация реализуемой технологии может не представлять общую причину пустоты свойств значений данных.

Некоторые из причин пустоты ассоциируются с именованными свойствами, которые могут использоваться как простые предикаты для всех значений данных. Это сделано для упрощения формулирования инвариантных выражений в остальной части настоящего стандарта.

Запомните разницу между семантическими свойствами и «компонентами» представления значений данных. В спецификации реализуемой технологии должны быть представлены только те компоненты, которые необходимы для вывода семантических свойств. Все предикаты пустоты значения nonNull, isNull, notApplicable, unknown и other могут быть выведены из свойства nullFlavor.

B.1.11.5 Предикат notApplicable: BL

Определение: предикат, указывающий, что причина пустоты nullFlavor данного исключительного значения равна «NA» (неприменимо), то есть правильное значение не имеет смысла в данном контексте.

```
invariant(ANY x) {
   x.notApplicable.equal(x.nullFlavor.implies(NA));
};
```

B.1.11.6 Предикат unknown: BL

Определение: предикат, указывающий, что причина пустоты nullFlavor данного исключительного значения равна «UNK» (неизвестно).

```
invariant(ANY x) {
   x.unknown.equal(x.nullFlavor.implies(UNK));
};
```

B.1.11.7 Предикат other: BL

Определение: предикат, указывающий, что причина пустоты nullFlavor данного исключительного значения равна «OTH» (другое), то есть правильное значение не принадлежит требуемому домену значений.

```
invariant(ANY x) {
   x.other.equal(x.nullFlavor.implies(OTH));
};
```

B.1.11.8 Предикат equal: BL (равенство)

Определение: равенство является рефлексивным, симметричным и транзитивным отношением между двумя значениями данных. Равенство возможно только между допустимыми значениями, пустые значения никогда не равны (даже если имеют одинаковую причину пустоты).

```
invariant(ANY x, y, z)
      where x.nonNull.and(y.nonNull).and(z.nonNull) {
   x.equal(x);                                    /* рефлексивность */
   x.equal(y).equal(y.equal(x));                  /* симметричность */
   x.equal(y).and(y.equal(z)).implies(x.equal(z)) /* транзитивность */
   x.equal(y).implies(x.dataType.equal(y.dataType);
};
```

Способ установления равенства должен быть определен для каждого типа данных. Если иное не указано, то два значения данных равны, если они не различимы, то есть если у них нет различающихся семантических свойств. Это общее определение равенства может быть «переопределено» в типе данных с помощью указания собственного отношения равенства. Такое переопределение отношения равенства может быть использовано для исключения семантических свойств из проверки на равенство. Если в типе данных какие-то семантические свойства исключены из его определения равенства, это означает, что определенные свойства (или аспекты свойств), не ставшие частью проверки на равенство, не существенны для смысла значения.

Например, физическая величина имеет два семантических свойства: (1) вещественное число, (2) кодированная единица измерения. Однако при проверке на равенство необходимо учитывать тот факт, что, к примеру, 1 метр равен 100 сантиметрам. Таким образом, независимые равенства двух семантических свойств являются слишком строгим критерием равенства двух величин. Поэтому в определении типа данных физической величины необходимо переопределить отношение равенства.

**B.1.12 Тип данных DataType (TYPE) (специализация типа данных ANY)**

Определение: метатип, объявленный, чтобы можно было давать формальное определение типа данных значения. Любой тип данных, определенный в настоящей спецификации, является значением типа DataType.

```
private type DataType alias TYPE specializes DataValue {
   CS  shortName;
   CS  longName;
   BN  implies(TYPE that);
};
```

B.1.12.1 Свойство shortName: CS

Определение: значение типа CS, содержащее краткое имя типа данных.

```
invariant(DataType x)
      where x.nonNull {
   x.shortName.nonNull;
};
```

B.1.12.2 Свойство longName: CS

Определение: значение типа CS, содержащее полное имя типа данных.

B.1.12.3 Свойство implies: BN

Тип данных вытекает (implies) из другого типа данных, если он имеет тот же тип или является его специализацией.

### B.2 Базовые типы

#### B.2.1 Тип данных Boolean (BL) (специализация типа данных ANY)

Определение: тип данных *BL* описывает значения двузначной логики. Значением типа *BL* может быть или «true» (истина), или «false» (ложь), или, как и у других значений, NULL (пустое).

```
type Boolean alias BL specializes ANY
   values(true, false) {
            BL  and(BL x);
            BL  not;
   literal  ST;
            BL  or(BL x);
            BL  xor(BL x);
            BL  implies(BL x);
};
```

При наличии пустого значения NULL двузначная логика расширяется до трехзначной в соответствии с таблицей B.4.

Таблица B.4 — Таблицы истинности для булевской логики с пустыми значениями

| **NOT** | |
|---|---|
| **true** | false |
| **false** | true |
| **NULL** | NULL |

| **AND** | **true** | **false** | **NULL** |
|---|---|---|---|
| **true** | true | false | NULL |
| **false** | false | false | false |
| **NULL** | NULL | false | NULL |

| **OR** | **true** | **false** | **NULL** |
|---|---|---|---|
| **true** | true | true | true |
| **false** | true | false | NULL |
| **NULL** | true | NULL | NULL |

Когда булевская операция выполняется над двумя экземплярами типа данных, имеющими разные причины пустоты nullFlavor, то причиной пустоты результата является первый общий предшественник этих двух разных значений nullFlavor. Однако приложения, соответствующие настоящему стандарту, могут выбрать любого общего предшественника.

B.2.1.1 Свойство not: BL

Определение: отрицание (not) значения типа *BL* превращает «true» в «false», «false» в «true» и NULL в NULL.

```
invariant(BL x) {
   true.not.equal(false);
   false.not.equal(true);
   x.isNull.equal(x.not.isNull);
};
```

B.2.1.2 Свойство and: BL

Определение: конъюнкция (and) является ассоциативной и коммутативной, при этом в качестве нейтрального элемента выступает значение «true». Конъюнкция значения «false» с любым булевским значение дает результат «false». Эти правила справедливы, даже если один или оба операнда пусты (NULL). Конъюнкция двух операндов со значением NULL дает результат NULL.

```
invariant(BL x) {
   x.and(true).equal(x);
   x.and(false).equal(false);
   x.isNull.implies(x.and(y).isNull);
};
```

B.2.1.3 Свойство or: BL

Определение: дизъюнкция x or y дает результат «false» в том и только том случае, когда x имеет значение «false» и y имеет значение «false».

```
invariant(BL x, y) {
   x.or(y).equal(x.not.and(y.not).not);
};
```

B.2.1.4 Свойство xor: BL

Определение: исключающая дизъюнкция xor ограничивает дизъюнкцию or таким образом, что два операнда не могут одновременно иметь значение «true».

```
invariant(BL x, y) {
   x.xor(y).equal(x.or(y).and(x.and(y).not));
};
```

B.2.1.5 Свойство implies: BL

Определение: правило импликации в форме «Если условие, ТО следствие». Логическая импликация определяется как дизъюнкция следствия (conclusion) и отрицание условия. Это означает, что если условие (condition) имеет значение «true», то для того, чтобы результат всего выражения имел значение «true», необходимо, чтобы следствие (conclusion) имело значение «true». Логическая импликация важна для конструирования инвариантных выражений.

```
invariant(BL condition, conclusion) {
   condition.implies(conclusion).equal(
      condition.not.or(conclusion));
};
```

Импликация необратима и не указывает, что имеет значение «true», если условие имеет значение «false» (ex falso quodlibet — лат. «из лжи следует что угодно»).

B.2.1.6 Литеральная форма

Литеральная форма типа данных Boolean определяется именованными значениями, указанными в конструкции values, а именно «true» и «false».

**B.2.2 Тип данных BooleanNonNull (BN) (специализация типа данных BL)**

Определение: тип данных *BN* ограничивает тип данных BL таким образом, то значение этого типа данных не может быть пустым. Этот тип данных предназначен для использования в тех случаях, когда пустое значение недопустимо.

```
private type BooleanNonNull alias BN specializes BL;
};
```

B.2.2.1 Свойство isNull: BN

```
invariant (BN x) {
   x.isNull.not
};
```

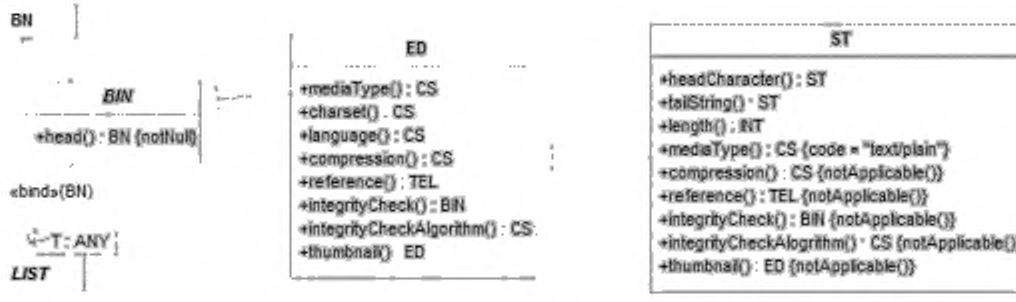


Рисунок B.3 — Обзор мультимедийных и текстовых типов данных

**B.2.3 Тип данных BinaryData (BIN) (специализация типа данных LIST<BN>)**

Определение: тип данных BIN представляет собой простой блок битов. Он является защищенным, так что не должен объявляться за пределами спецификации типов данных.

Бит семантически эквивалентен непустому значение типа BL. Таким образом, семантически все двоичные данные являются последовательностями непустых значений типа BL.

```
protected type BinaryData alias BIN specializes LIST<BN>;
```

Примечание — Конкретное представление произвольных двоичных данных является прерогативой спецификации реализуемой технологии. Как это в ней делается, зависит от базовой технологии реализации (символьная или двоичная) и от представляемых данных. Семантически символьные данные представляются как двоичные данные, однако символьная спецификация реализуемой технологии не должна преобразовывать символьные данные в произвольные двоичные данные, а затем эти двоичные данные преобразовывать в коды символов. В конечном счете при любой символьной технологии будут передаваться двоичные данные.

Пустая последовательность считается не двоичными данными, а пустым значением (NULL). Другими словами, непустые двоичные данные содержат хотя бы один бит. Ни один бит в непустых двоичных данных не может иметь значение NULL.

```
invariant(BIN x)
      where x.nonNull {
   x.notEmpty;
   x.length.greaterThan(0);
};
```

**B.2.4 Тип данных EncapsulatedData (ED) (специализация типа данных BIN)**

Определение: данные, в основном рассчитанные на чтение человеком или на дальнейшую машинную обработку данных за пределами области применения стандарта HL7. К ним относятся неформатированный или форматированный письменный текст, мультимедийные данные или структурированная информация, определенная другим стандартом (например, XML-подписи). Вместо самих данных значение типа *ED* может содержать только ссылку на данные (см. описание типа данных TEL). Заметьте, что тип данных ST — это специализация типа данных ED, у которой свойство mediaType имеет фиксированное значение text/plain (только текст).

Таблица B.5 — Сводка свойств типа данных EncapsulatedData

| Имя | Тип | Описание |
|---|---|---|
| mediaType | CS | Идентифицирует тип инкапсулированных данных и метод их интерпретации или визуализации |
| charset | CS | Для типов данных, основанных на кодировании символов, это свойство указывает используемый набор символов и его кодировку. Набор символов должен идентифицироваться в соответствии с правилами регистрации наборов символов в организации Internet Assigned Numbers Authority (IANA) [http://www.iana.org/assignments/character-sets] и положениями документа RFC 2978 [http://www.ietf.org/rfc/rfc2978.txt] |
| language | CS | Для информации, основанной на символах, свойство language (язык) указывает человеческий язык текста |
| compression | CS | Указывает, являются ли исходные байтовые данные сжатыми и какой алгоритм сжатия использован |
| reference | TEL | Телекоммуникационный адрес (TEL), например URL для HTTP или FTP, по которому можно получить в точности те же самые данные, которые могли бы быть представлены внутри значения типа данных ED |
| integrityCheck | BIN | Контролем целостности является короткое двоичное значение, представляющее криптографически стойкую контрольную сумму, вычисленную по двоичным данным. Это свойство предназначено для использования при передаче ссылок на данные и позволяет позже проверить, не изменились ли ссылочные данные после того, как было создано инкапсулированное значение, ссылающееся на эти данные |

*Окончание таблицы В.5*

| Имя | Тип | Описание |
|---|---|---|
| integrityCheckAlgorithm | CS | Указывает алгоритм вычисления свойства integrityCheck[1]. В настоящее время промышленным стандартом является алгоритм вычисления криптографически стойкой контрольной суммы Secure Hash Algorithm-1 (SHA-1). Он заменил алгоритм MD5 всего лишь пару лет назад, когда обнаружились некоторые недостатки в безопасности этого алгоритма. В настоящее время становится популярным также алгоритм SHA-256 |
| thumbnail | ED | Сокращенное представление полных данных (эскиз), требующее гораздо меньше ресурсов, нежели полные данные, но при этом обладающее отличительными способностями полных данных. Обычно эскизы используются для ссылочных инкапсулированных данных. Они позволяют пользователю выбирать нужные данные, не загружая по ссылке их полное представление |

```
type EncapsulatedData alias ED specializes BIN {
   CS   mediaType;
   CS   charset;
   CS   language;
   CS   compression;
   TEL  reference;
   BIN  integrityCheck;
   CS   integrityCheckAlgorithm;
   ED   thumbnail;
   BL   equal(ANY x);
};
```

Инкапсулированные данные могут быть представлены в двух формах — вложенные и ссылочные. Вложенные данные передаются или перемещаются как часть значения инкапсулированного типа данных. А ссылочные данные могут находиться в другом (дистанционно удаленном) месте. Независимо от формы данные являются теми же самыми.

В.2.4.1 Свойство mediaType: CS

Определение: идентифицирует тип инкапсулированных данных (тип среды) и метод их интерпретации или визуализации.

Свойство mediaType является обязательным непустым, то есть любой непустой экземпляр типа данных ED должен иметь непустое свойство mediaType.

```
invariant(ED x)
      where x.nonNull {
   x.mediaType.nonNull;
};
```

Организация IANA определяет домены типов среды в стандартах Интернет RFC 2045 [http://www.ietf.org/rfc/rfc2045.txt] и RFC 2046 [http://www.ietf.org/rfc/rfc2046.txt]. Согласно RFC 2046 тип среды состоит из двух частей:

- тип среды верхнего уровня;
- подтип среды.

Однако эта спецификация трактует полный тип среды как единую строку, формат которой определен организацией IANA, а именно, имя верхнего уровня, за которым следует символ косой черты «/», а за ним — имя подтипа среды. Актуальные зарегистрированные типы среды хранятся в базе данных организации IANA, к которой можно получить доступ по ссылке [http://www.iana.org/assignments/media-types/index.html]. В настоящее время определено свыше 160 разных типов среды MIME и этот список быстро растет. В принципе, могут использоваться все типы среды, определенные организацией IANA.

[1] В настоящее время промышленным стандартом является алгоритм вычисления криптографически стойкой контрольной суммы Secure Hash Algorithm-1 (SHA-1). Он заменил алгоритм MD5 всего лишь пару лет назад, когда обнаружились некоторые недостатки в безопасности этого алгоритма. В настоящее время по умолчанию для вычисления контрольной суммы используется алгоритм SHA-1. Становится популярным также алгоритм SHA-256.

Для обеспечения интероперабельности в настоящем стандарте некоторые типы среды объявляются более предпочтительными. Это позволяет задать наибольший общий знаменатель, при котором интероперабельность не только возможна, но и может быть достаточно эффективной для обеспечения достаточно развитых потребностей в передаче мультимедийных данных.

В таблице В.6 описан статус некоторых типов среды MIME, который формулируется в следующих терминах:

- **обязательный**: каждое приложение, соответствующее стандарту HL7 и рассчитанное на определенный вид среды, должно как минимум обрабатывать обязательные типы среды. Для каждого вида среды существует один обязательный тип среды. Некоторые типы среды предназначены для специальных целей, что обозначено формулировкой «требуется для...»;
- **рекомендованный**: другие типы среды рекомендуются для конкретных целей. Для каждой цели должно быть указано совсем немного дополнительных рекомендованных типов среды, при этом обоснование, условия применения и предположения об использовании этих типов должны быть как можно более ясными;
- **индифферентный**: такой статус означает, что комитет HL7 не запрещает, но и не одобряет использование этого типа среды. По умолчанию все типы среды, не указанные в таблице В.6, считаются индифферентными. Поскольку для большинства практических сценариев выделены один обязательный и несколько рекомендованных типов среды, то к применению индифферентных типов среды надо относиться очень консервативно;
- **запрещенный**: запрещенные типы среды не должны использоваться, поскольку либо им присущи недостатки, либо имеются более приемлемые альтернативы, либо их применение влечет за собой определенные риски. К последним могут относиться риски нарушения безопасности, скажем, потенциальный риск переноса компьютерных вирусов в данные этого типа среды. Однако не всякий тип среды, имеющий недостатки или подверженный риску, считается запрещенным. Типы среды, не упомянутые в таблице В.6 и по умолчанию считающиеся индифферентными, также могут обладать подобными недостатками.

Таблица В.6 — Домены типов среды

| Код | Имя | Статус | Определение |
|---|---|---|---|
| text/plain | Неформатированный текст | Обязательный | Любой неформатированный текст. Этот тип среды используется по умолчанию. Он эквивалентен строковому типу данных (ST) |
| text/x-hl7-ft | Текст HL7 | Рекомендованный | Этот тип среды предназначен для совместимости с типом данных FT, определенным в HL7 v2.x. Рекомендуется использовать его только для обратной совместимости с системами, где реализован стандарт HL7 v2.x |
| text/html | Текст HTML | Рекомендованный | Текст, размеченный в соответствии со спецификацией языка разметки гипертекста HTML (Hypertext Markup Language). Разметка HTML достаточна для представления большинства типографских документов. Язык HTML является платформенно независимым и широко используется |
| application/pdf | PDF | Рекомендованный | Формат переносимых документов PDF (Portable Document Format) рекомендован для сверстанного форматированного текста, предназначенного только для чтения. Он имеет открытую спецификацию, платформенно независим и широко распространен. Существуют свободно распространяемые средства для создания и отображения данных в этом формате |
| text/xml | Текст XML | Индифферентный | Предназначен для структурированных строковых данных. Существует определенный риск, что общий формат документов SGML/XML слишком сложен для совместного использования различными приложениями |
| text/rtf | Текст RTF | Индифферентный | Формат Rich Text (RTF) широко используется для совместного использования документов, подготовленных процессорами текстов. Однако у него есть проблемы совместимости, поскольку он существенно зависит от используемого процессора текстов. Может быть полезен для совместной работы с текстом, подготовленным разными процессорами текстов |

*Продолжение таблицы В.6*

| Код | Имя | Статус | Определение |
|---|---|---|---|
| application/msword | MSWORD | Запрещенный | Этот формат весьма подвержен проблемам совместимости. При необходимости совместного редактирования текста вместо него следует использовать обязательный формат «text/plain» или форматы «text/html» и «text/rtf» |
| audio/basic | Базовый формат аудиозаписи | Обязательный | Формат одноканального звука, закодированного в соответствии со спецификацией 8-бит правила мю ISDN [PCM] с частотой оцифровки 8000 Гц. Этот формат стандартизован организацией CCITT в документе Fascicle III.4 -Recommendation G.711. Pulse Code Modulation (PCM) of Voice Frequencies. Geneva, 1972 |
| audio/mpeg | MPEG audio layer 3 | Обязательный | MPEG-1 Audio layer-3 (MP3) представляет собой алгоритм сжатия и формат файла, определенные в стандартах ИСО 11172-3 и ИСО 13818-3. MP3 позволяет использовать настраиваемую частоту оцифровки от телефонии с сильным сжатием до звука с качеством компакт-диска CD |
| audio/k32adpcm | K32ADPCM Audio | Индиферентный | Формат ADPCM позволяет сжимать аудиоданные. Он определен в спецификации Интернет RFC 2421 [ftp://ftp.isi.edu/in-notes/rfc2421.txt]. Насколько широко он реализован, неизвестно |
| image/png | Изображение в формате PNG | Обязательный | PNG (Portable Network Graphics) [http://www.cdrom.com/pub/png] представляет собой широко используемый стандарт сжатия изображений без потерь. Доступны приложения с открытым кодом, обеспечивающие обработку изображений в формате PNG |
| image/gif | Изображение в формате GIF | Индиферентный | GIF представляет собой широко используемый популярный формат хранения изображений. Однако он отягощен патентами[1)], поэтому к его использованию надо подходить осторожно |
| image/jpeg | Изображение в формате JPEG | Обязательный | Этот формат обязателен для применения высокого сжатия к фотографии с широкой цветовой гаммой. Он обеспечивает сжатие с потерями, но различие от сжатия без потерь визуально почти неотличимо |
| application/dicom | DICOM | Рекомендованный | Тип среды MIME для формата DICOM (Digital Imaging and Communications in Medicine — цифровые изображения и связь в медицине) определен в документе RFC3240 [http://ietf.org/rfc/rfc3240.txt] |
| image/g3fax | Изображение в формате G3Fax | Рекомендованный | Рекомендуется только для факсимильных изображений |
| image/tiff | TIFF Image | Индиферентный | Хотя формат TIFF (Tag Image File Format — формат растровых графических изображений) и является международным стандартом, на практике с ним связано большое число проблем интероперабельности. У него слишком много версий, обрабатываемых не всеми программами |
| video/mpeg | MPEG видео | Обязательный | Международный стандарт MPEG широко используется. Он весьма эффективен для видеосигнала с широкой цветовой гаммой. Доступны приложения с открытым кодом. Степень интероперабельности очень высокая |

[1)] Согласно https://ru.wikipedia.org/wiki/GIF от 16.09.2014, срок действия последнего патента на GIF истек 11 августа 2006 года. — *Прим. перев.*

*Окончание таблицы В.6*

| Код | Имя | Статус | Определение |
|---|---|---|---|
| video/x-avi | X-AVI видео | Запрещенный | Формат файлов AVI представляет собой всего лишь обертку для многих различных кодеков. Он является источником большого числа проблем интероперабельности |
| model/vrml | VRML Model | Рекомендованный | Это открытый стандартизованный формат трехмерных моделей, который может использоваться для приложений виртуальной реальности, например, при проведении анатомических или биохимических исследований (визуализации пространственной структуры макромолекул) |

Перечень обязательных типов среды весьма мал, поэтому на приложения, соответствующие стандарту HL7, в особенности на унаследованные системы, невыполнимых требований не накладывается. В общем случае от таких приложений не требуется поддержки любого другого типа среды, отличающегося от неформатированного текста. Например, многим системам не нужны аудиоданные, поскольку они могут отобразить своим пользователям только неформатированный текст. В заявлении о соответствии таких приложений вполне можно указать, что они не обрабатывают аудиозапись. И только если заявлено об обработке аудиозаписи, такое приложение должно обеспечивать обработку обязательного типа среды для аудиозаписи.

В.2.4.2 Свойство charset: CS

Определение: для типов кодирования символов это свойство указывает используемый набор символов и используемую кодировку. Набор символов должен идентифицироваться в соответствии с правилами регистрации наборов символов в организации Internet Assigned Numbers Authority (IANA) [http://www.iana.org/assignments/character-sets] и положениями документа RFC 2978 [http://www.ietf.org/rfc/rfc2978.txt].

Домен наборов символов ведется организацией Internet Assigned Numbers Authority (IANA) [http://www.iana.org/assignments/character-sets]. В базе данных IANA указаны имена и различные псевдонимы для большинства наборов символов. Для целей стандартов HL7 использование нескольких различных имен одного и того же набора символов не допускается. В них должно использоваться то имя, которое помечено организацией IANA как «предпочтительное для MIME». Если ни один из псевдонимов не имеет такой пометки, то для стандартов HL7 должно использоваться основное имя набора символов.

В таблице В.7 перечислены некоторые из наборов символов, определенных организацией IANA и представляющих интерес для текущих членов комитета HL7.

Таблица В.7 — Домен наборов символов

| Код | Имя | Определение |
|---|---|---|
| EBCDIC | EBCDIC | Использование этого набора символов индифферентно для стандартов HL7 |
| ISO-10646-UCS-2 | ISO-10646-UCS-2 | Запрещен для использования в стандартах HL7 |
| ISO-10646-UCS-4 | ISO-10646-UCS-4 | Запрещен для использования в стандартах HL7 |
| ISO-8859-1 | ISO-8859-1 | Использование этого набора символов индифферентно для стандартов HL7 |
| ISO-8859-2 | ISO-8859-2 | Использование этого набора символов индифферентно для стандартов HL7 |
| ISO-8859-5 | ISO-8859-5 | Использование этого набора символов индифферентно для стандартов HL7 |
| JIS-2022-JP | JIS-2022-JP | Использование этого набора символов индифферентно для стандартов HL7 |
| US-ASCII | US-ASCII | Обязателен для использования в стандартах HL7 |
| UTF-7 | UTF-7 | Использование этого набора символов индифферентно для стандартов HL7 |
| UTF-8 | UTF-8 | Обязателен для поддержки кодировки Unicode |

Примечание — Перечень, приведенный в таблице В.7, не является полным и исключительным. Национальные филиалы комитета HL7 могут предложить собственные рекомендации по наборам символов, используемых на их территории. Эти рекомендации могут добавить дополнительные наборы символов и указать иные статусы рекомендаций для наборов, перечисленных в таблице В.7.

Свойство charset должно быть известно, если значение типа ED представляет собой символьные данные в любой форме. Если данные вложены в значение, то набор символов должен быть известен. Если эти данные предоставляются по ссылке, и метод доступа не обеспечивает идентификацию набора символов, обычно возвращаемую в заголовке MIME, то эта идентификация должна быть указана как компонент типа данных ED.

Более полное обсуждение набора символов и сопутствующих вопросов можно найти в публикации «Character Model for the World Wide Web» [http://www.w3.org/TR/charmod/].

В.2.4.3 Свойство language: CS

Применительно к символьной информации свойство language указывает человеческий язык текста.

Необходимость указания кода языка для текстовых данных документирована в публикации RFC 2277 «IETF Policy on Character Sets and Languages» [http://www.ietf.org/rfc/rfc2277.txt]. Дополнительную базовую информацию можно найти в документе «Using International Characters in Internet Mail» [http://www.imc.org/mail-i18n.html] (меморандум консорциума Internet Mail Consortium).

Принципы, положенные в основу домена значений этого свойства изложены в стандарте Интернет RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt]. Система кодирования, описанная в этом документе, конструируется из основного компонента, содержащего тег с кодом языка, определенным в стандарте ИСО 639, и двух кодов расширения языков, не предусмотренных в этом стандарте. Код может содержать дополнительный компонент, содержащий тег с двухбуквенным кодом страны, определенным в стандарте ИСО 3166, или с расширением кода языка, определенным организацией Internet Assigned Names Authority [http://www.iana.org/assignments/language-tags][1].

В то время как теги языка обычно изменяют смысл текста, код языка не меняет смысл символов текста[2].

Примечание — Представление тегов языка в тексте существенно зависит от спецификации реализуемой технологии. В ней может использоваться нативный способ указания тегов языка, предусмотренный в ее целевой технологии реализации. В некоторых технологиях информация о языке может быть указана в отдельном компоненте, например, в XML язык строки может быть указан в теге xml:lang. В других коды языка могут быть частью двоичного представления строк символов, как это специфицировано в стандарте ИСО 10646 (Unicode) и его тегах «plane-14».

Если в технологии реализации тег языка не обязателен, то не должно быть обязательным и свойство language. Семантически объявление языка в строках следует логике умолчаний. Если на данной территории реализации может поддерживаться несколько языков, то эта территория сама устанавливает правила задания языка в случае, когда никакой язык не указан. Если никаких правил не задано, то подразумевается местный язык читателя. Если язык указан для всего сообщения или документа, то этот язык используется по умолчанию. Если язык указан для любого элемента информации, стоящего выше в синтаксической иерархии, то этот язык используется по умолчанию во всех подчиненных текстовых значениях.

Если теги языка присутствуют в начале кодированного двоичного текста (например, в тегах plane-14 кодировки Unicode), то они являются источником значения свойства language для инкапсулированных данных.

В.2.4.4 Свойство compression: CS

Определение: указывает, являются ли байтовые данные результатом сжатия исходных данных и какой алгоритм сжатия был использован.

Таблица В.8 — Домен алгоритмов сжатия

| Код | Имя | Статус | Определение |
|---|---|---|---|
| DF | deflate | Обязательный | Формат дефляционного сжатия, предложенный в документе RFC 1951 [http://www.ietf.org/rfc/rfc1951.txt] |
| GZ | gzip | Индифферентный | Формат сжатия данных, совместимый с широко используемой утилитой GZIP и описанный в документе RFC 1952 [http://www.ietf.org/rfc/rfc1952.txt] (использует алгоритм дефляции) |

[1] Система кодирования, описанная в документе RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt], одобрена комитетом HL7 для всех ссылок на человеческие языки, используемых в типах данных и других местах.

[2] По этой причине система или место реализации, которые неспособны обрабатывать многоязычный текст или имена, могут спокойно игнорировать свойство language.

*Окончание таблицы В.8*

| Код | Имя | Статус | Определение |
|---|---|---|---|
| ZL | zlib | Индифферентный | Сжатый формат данных, также использующий алгоритм дефляции. Описан в документе RFC 1950 [http://www.ietf.org/rfc/rfc1952.txt] |
| Z | compress | Запрещенный | Оригинальный алгоритм сжатия и формат файла, предложенный в операционной системе UNIX и использующий алгоритм LZC (вариант алгоритма LZW). Отягощен патентами и менее эффективен по сравнению с алгоритмом дефляции |

Значения типа данных ST никогда не должны быть сжатыми.

В.2.4.5 Свойство reference: TEL

Определение: телекоммуникационный адрес (TEL), например URL для HTTP или FTP, по которому находятся ровно те же самые двоичные данные, которые могли бы быть вложенными в инкапсулированные данные.

Семантическое значение инкапсулированных данных одно и то же вне зависимости, являются ли они вложенными или ссылочными. Однако инкапсулированные данные, не являющиеся вложенными, имеют другое поведение, поскольку любая попытка использования этих данных требует загрузки из источника, на который показывает ссылка. Инкапсулированные данные могут одновременно содержать и вложенные данные, и ссылку.

Ссылка должна указывать на те же самые данные, что и вложены. Если данные, полученные по ссылке, не проходят проверку на целостность, не совпадают с вложенными данными или с теми данными, которые были извлечены по ссылке ранее и сохранены в кэше, то это должно признаваться ошибкой.

Свойство reference может содержать компонент usablePeriod (период доступности), указывающий, что данные могут быть доступны по ссылке только ограниченный период времени. Независимо от того, ограничена ли доступность ссылки значением usablePeriod, содержание ссылочных данных должно оставаться фиксированным. Любое приложение, использующее ссылку, должно получить те же самые данные. Ссылка не может быть повторно использована для получения другой версии этих данных или других данных.

В зависимости от атрибута или компонента, объявляющего инкапсулированные данные, использование ссылочных данных может быть запрещено. Значение типа ST должно всегда быть вложенным.

В.2.4.6 Свойство integrityCheck: BIN

Определение: свойство integrityCheck содержит короткое двоичное значение, представляющее криптографически стойкую контрольную сумму, вычисленную по двоичным данным. Это свойство предназначено для использования при передаче ссылок на данные и позволяет позже проверить, не изменились ли ссылочные данные после того, как было создано инкапсулированное значение, ссылающееся на эти данные.

Если данные, извлеченные по ссылке, не проходят проверку целостности, это должно признаваться ошибкой.

Контрольная сумма integrityCheck вычисляется с помощью алгоритма, указанного в свойстве integrityCheckAlgorithm. По умолчанию должен использоваться алгоритм *Secure Hash Algorithm-1* (SHA-1). Контрольная сумма представляется в двоичном виде в соответствии с правилами, принятыми в алгоритме контрольного суммирования.

Контрольная сумма вычисляется по двоичным данным, содержащимся в компоненте данных или доступным по ссылке. Перед вычислением контрольной суммы никаких преобразований не делается. Если данные сжаты, то контрольная сумма вычисляется по сжатым данным.

В.2.4.7 Свойство integrityCheckAlgorithm: CS

Определение: указывает алгоритм, использованный для вычисления значения свойства integrityCheck[1)].

Таблица В.9 — Домен значений свойства integrityCheckAlgorithm

| Код | Имя | Определение |
|---|---|---|
| SHA-1 | secure hash algorithm — 1 | Этот алгоритм определен в документе FIPS PUB 180-1: Secure Hash Standard (по состоянию на 17 апреля 1995 г.) |
| SHA-256 | secure hash algorithm — 256 | Этот алгоритм определен в документе FIPS PUB 180-2: Secure Hash Standard |

[1)] В настоящее время промышленным стандартом является алгоритм вычисления криптографически стойкой контрольной суммы Secure Hash Algorithm-1 (SHA-1). Он заменил алгоритм MD5 всего лишь пару лет назад, когда обнаружились некоторые недостатки в безопасности этого алгоритма. В настоящее время по умолчанию для вычисления контрольной суммы используется алгоритм SHA-1. Становится популярным также алгоритм SHA-256.

B.2.4.8 Свойство thumbnail: ED

Определение: сокращенное представление полных данных (эскиз), требующее гораздо меньше ресурсов, нежели полные данные, но при этом обладающее отличительными способностями полных данных. Обычно эскизы используются для ссылочных инкапсулированных данных. Они позволяют пользователю выбирать нужные данные, не загружая по ссылке их полное представление.

Первоначально термин «эскиз» (thumbnail) использовался для обозначения изображения в более низком разрешении (или меньшего размера) по сравнению с другим изображением. Однако это понятие может метафорически использоваться для других типов данных, кроме изображений. Например, видеозапись может быть представлена коротким клипом, аудиоклип может быть представлен другим аудиоклипом, более коротким, имеющим меньшую частоту оцифровки или использующим искажающее сжатие.

В зависимости от атрибута или компонента, объявляющего инкапсулированные данные, использование эскизов может быть запрещено. Значения типа ST никогда не имеют эскизов, и эскиз не может содержать вложенный эскиз.

```
invariant(ED x)
      where x.thumbnail.nonNull |
   x.thumbnail.thumbnail.isNull;
];
```

Примечание — В спецификациях реализуемой технологии следует рассматривать случай, когда и эскиз, и оригинал имеют одни и те же свойства типа, набора символов и сжатия. Тогда эти свойства можно не задавать для эскиза, а считать «унаследованными» от соответствующих свойств основных инкапсулированных данных.

B.2.4.9 Свойство equality: BL, унаследовано от типа данных ANY

Определение: два значения типа *ED* равны в том и только том случае, если у них одинаковы типы среды и данные. Если значения типа *ED* содержат ссылочные или сжатые (упакованные) данные, то в проверке на равенство должны участвовать данные, извлеченные по ссылке и распакованные. Свойства сжатия compression, эскиза thumbnail и ссылки reference в проверке на равенство не участвуют. Кроме того, из этой проверки исключается свойство языка language, поскольку сложно понять, какой язык подразумевается, если он не указан. Если типом среды (свойство mediaType) являются символьные данные и свойства набора символов (charset) не одинаковы, то перед проверкой надо отобразить данные на общий набор символов.

Свойства контрольной суммы integrityCheck и алгоритма контрольного суммирования integrityCheckAlgorithm исключаются из проверки на равенство. Однако в случае, если алгоритмы контрольного суммирования одинаковы, вполне практичной является проверка совпадения контрольных сумм, поскольку оно может служить признаком равенства.

**B.2.5 Тип данных CharacterString (ST) (специализация типа данных ED)**

Определение: строковый тип данных, предназначенный для представления текста, предназначенного в основном для машинной обработки (например, сортировка, запросы, индексирование и т. д.). Используется для представления имен, символов и формальных выражений.

Тип данных ST является ограничением типа данных ED, у которого свойство ED.mediaType имеет фиксированное значение «text/plain» (неформатированный текст), а данные являются вложенными и не сжатыми. Поэтому свойства compression, reference, integrityCheck, integrityCheckAlgorithm и thumbnail к типу данных ST неприменимы. Строковый тип данных используется, когда внешний вид текста не имеет значения, что справедливо для формальных выражений и всех типов имен.

Таблица B.10 — Сводка свойств строкового типа данных CharacterString

| Имя | Тип | Описание |
|---|---|---|
| mediaType | CS | Идентифицирует тип инкапсулированных данных и метод их интерпретации или визуализации |
| charset | CS | Для типов данных, основанных на кодировании символов, это свойство указывает используемый набор символов и его кодировку. Набор символов должен идентифицироваться в соответствии с правилами регистрации наборов символов в организации Internet Assigned Numbers Authority (IANA) [http://www.iana.org/assignments/character-sets] и положениями документа RFC 2978 [http://www.ietf.org/rfc/rfc2978.txt] |
| language | CS | Для информации, основанной на символах, свойство language (язык) указывает человеческий язык текста |

Тип данных ST интерпретирует инкапсулированные данные как символьные (а не битовые) в зависимости от свойства charset инкапсулированного типа данных.

```
type CharacterString alias ST specializes ED {
   INT  length;
   ST   headCharacter;
   ST   tailString;
};
```

Примечание — Поскольку в данном случае многие свойства инкапсулированных данных имеют значения по умолчанию, то в спецификации реализуемой технологии эти свойства вообще не надо предоставлять. В действительности если кодировка символов также фиксирована, то достаточно представлять только закодированные символьные данные.

Свойства headCharacter (голова) и tailString (хвост) определяют тип данных *ST* как последовательность элементов, каждый из которых уникально идентифицирует один символ из объединения наборов всех символов, известных во всех языках мира[1)].

Голова типа данных ST представляет собой строку, состоящую ровно из одного символа. Значение типа ST должно содержать хотя бы один символ, в противном случае оно считается пустым. Значение типа ST, имеющее нулевую длину, является исключительным значением (NULL), а не допустимым значением.

```
invariant(ST x)
      where x.nonNull {
   x.headCharacter.notEmpty;
   x.headCharacter.length.equal(1);
   x.headCharacter.tailString.isEmpty;
   x.tailString.isEmpty.implies(x.length.equal(1));
   x.tailString.notEmpty.implies(x.length.equal(x.tailString.length.successor));
};
```

Длиной значения типа данных ST является число символов, а не число кодированных байтов. Байтовое кодирование является предметом спецификации реализуемой технологии, а не прикладного уровня.

К пробельным символам, содержащимся в значениях типа данных ST, применяются следующие правила:

- пробельными символами считаются TAB, пробел и символ конца строки;
- как ведущие, так и концевые пробельные символы считаются значащими;
- различные пробельные символы не являются взаимозаменяемыми;
- различные представления конца строки нормализуются в соответствии с методом, описанным в спецификации языка XML [Section 2.11 End-of-Line Handling];
- последовательности пробельных символов не могут сжиматься для получения более коротких последовательностей.

**Требование**

Тип данных ST является специализацией типа данных ED, поэтому каждый атрибут модели RIM, имеющий тип ED, может быть ограничен до типа данных ST. Наиболее важным случаем является атрибут Act.text, который имеет тип данных ED, позволяющий указывать ссылки и использовать мультимедийные данные. Однако тип этого атрибута нередко ограничивается до неформатированного текста.

---

[1)] Стандарт ИСО/МЭК 10646-1:1993 определяет символ как «Член множества элементов, используемых для организации, управления или представления данных». В техническом отчете ИСО/МЭК ТО 15285 «Информационные технологии. Операционная модель применения графических символов и глифов» обсуждаются проблемы, связанные с определением понятия символов. Примечательно, что символы рассматриваются как абстрактные единицы информации, не зависящие от шрифта или языка. В стандарте ИСО 10646 (UNICODE [http://www.unicode.org]) (в Японии JIS X0221) описан глобально применимый набор символов, уникально идентифицирующий все символы всех языков мира.

В настоящей спецификации стандарт ИСО 10646 используется как семантическая модель строк символов. Важный момент состоит в том, что в ней нет понятия отдельных наборов символов и переключения от одного набора к другому. Наборы символов и их кодировки рассматриваются на уровне спецификации реализуемой технологии. Формальное определение указывает на этот эффект, поскольку каждый символ сам по себе является значением типа ST, имеющим свойство набора символов charset. Поэтому двоичное кодирование каждого символа всегда осуществляется в контексте определенного набора символов. Это вовсе не означает, что спецификация реализуемой технологии всегда должна представлять строку символов как последовательность полновесных значений типа данных ED. Это значит лишь то, что на прикладном уровне понятие кодировки символов не имеет значения для обсуждения строк символов.

B.2.5.1 Свойство mediaType: CS (унаследовано от типа данных ED)

```
invariant(ST x)
      where x.nonNull {
   x.mediaType.equal("text/plain");
};
```

Это свойство имеет фиксированное значение «text/plain».

B.2.5.2 Свойство charset: CS (унаследовано от типа данных ED)

```
invariant(ST x)
      where x.nonNull {
   x.charset.nonNull;
};
```

Для значений типа данных ST должен быть известен набор символов.

B.2.5.3 Свойство language: CS (унаследовано от типа данных ED)

Определение: применительно к символьной информации свойство language указывает человеческий язык текста.

Необходимость указания кода языка для текстовых данных документирована в публикации RFC 2277 «IETF Policy on Character Sets and Languages» [http://www.ietf.org/rfc/rfc2277.txt]. Дополнительную базовую информацию можно найти в документе «Using International Characters in Internet Mail» [http://www.imc.org/mail-i18n.html] (меморандум консорциуима Internet Mail Consortium).

Принципы, положенные в основу домена значений этого свойства, изложены в стандарте Интернет RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt]. Система кодирования, описанная в этом документе, конструируется из основного компонента, содержащего тег с кодом языка, определенным в стандарте ИСО 639, и двух кодов расширения языков, не предусмотренных в этом стандарте. Код может содержать дополнительный компонент, содержащий тег с двухбуквенным кодом страны, определенным в стандарте ИСО 3166, или с расширением кода языка, определенным организацией Internet Assigned Names Authority [http://www.iana.org/assignments/language-tags][1].

В то время как теги языка обычно изменяют смысл текста, код языка не меняет смысл символов текста[2].

Примечание — Представление тегов языка в тексте существенно зависит от спецификации реализуемой технологии. В ней может использоваться нативный способ указания тегов языка, предусмотренной в ее целевой технологии реализации. В некоторых технологиях информация о языке может быть указана в отдельном компоненте, например, в XML язык строки может быть указан в теге xml:lang. В других коды языка могут быть частью двоичного представления строк символов, как это специфицировано в стандарте ИСО 10646 (Unicode) и его тегах «plane-14».

Если в технологии реализации тег языка необязателен, то не должно быть обязательным и свойство language. Семантически объявление языка в строках следует логике умолчаний. Если на данной территории реализации может поддерживаться несколько языков, то эта территория сама устанавливает правила задания языка в случае, когда никакой язык не указан. Если никаких правил не задано, то подразумевается местный язык читателя. Если язык указан для всего сообщения или документа, то этот язык используется по умолчанию. Если язык указан для любого элемента информации, стоящего выше в синтаксической иерархии, то этот язык используется по умолчанию во всех подчиненных текстовых значениях.

Если теги языка присутствуют в начале кодированного двоичного текста (например, в тегах plane-14 кодировки Unicode), то они являются источником значения свойства language для инкапсулированных данных.

B.2.5.4 Свойство compression: CS (фиксированное)

```
invariant(ST x)
      where x.nonNull {
   x.compression.notApplicable;
};
```

Значения типа ST не могут быть сжатыми.

---

[1] Система кодирования, описанная в документе RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt], одобрена комитетом HL7 для всех ссылок на человеческие языки, используемых в типах данных и других местах.

[2] По этой причине система или место реализации, которые неспособны обрабатывать многоязычный текст или имена, могут спокойно игнорировать свойство language.

B.2.5.5 Свойство reference: TEL (фиксированное)

```
invariant(ST x)
      where x.nonNull {
   x.reference.notApplicable;
};
```

Значения типа ST не могут быть ссылаться на содержание, находящееся в каком-то другом месте.

B.2.5.6 Свойство integrityCheck: BIN (фиксированное)

```
invariant(ST x)
      where x.nonNull {
   x.integrityCheck.notApplicable;
};
```

Контрольная сумма значений типа ST не вычисляется.

B.2.5.7 Свойство integrityCheckAlgorithm: CS (фиксированное)

```
invariant(ST x)
      where x.nonNull {
   x.integrityCheckAlgorithm.notApplicable;
};
```

К значениям типа ST алгоритм контрольного суммирования не применяется.

B.2.5.8 Свойство thumbnail: ED (фиксированное)

```
invariant(ST x)
      where x.nonNull {
   x.thumbnail.notApplicable;
};
```

Эскизы для значений типа ST не могут быть определены.

B.2.5.9 Литеральная форма

Определены два варианта литералов типа данных ST: форма токена и строка в кавычках[1]. Форма токена состоит из строчных и прописных символов латинского алфавита, десяти десятичных цифр и подчеркивания. Строка в кавычках может содержать между кавычками любой символ. Кавычки предотвращают интерпретацию строки символов как литерал другого типа. Форма токена используется для представления имен и ключевых слов, разбираемых в языке спецификации типов данных.

```
ST.literal ST {
   ST : /"[^]+"/          { $.equal($1); }  /* quoted string */
      | /[a-zA-Z0-9_]+/   { $.equal($1); }; /* token form */
};
```

Примечание — Поскольку литералы типа данных ST играют фундаментальную роль в технологии реализации, в большинстве спецификаций реализуемой технологии будет использоваться некоторая модифицированная форма литерала строки символов. Однако разработчики этих спецификаций должны учитывать возможность пересечения литеральной формы типа данных ST с литеральными формами, определенными для других типов данных. Такое пересечение особенно критично, если литеральная форма другого типа данных структурирована в виде основных компонентов, разделенных специальными символами (например, вещественное число, физическая величина, литералы множества и списка и т. д.).

[1] Литерал типа данных ST представляет собой преобразование строки символов в другой тип данных. Очевидно, литералы типа данных ST для символьных строк представляют собой циклическое, если не избыточное образование. Поэтому литеральная форма в основном указывает, каким образом осуществляется разбор строк символов на языке определения типов данных.

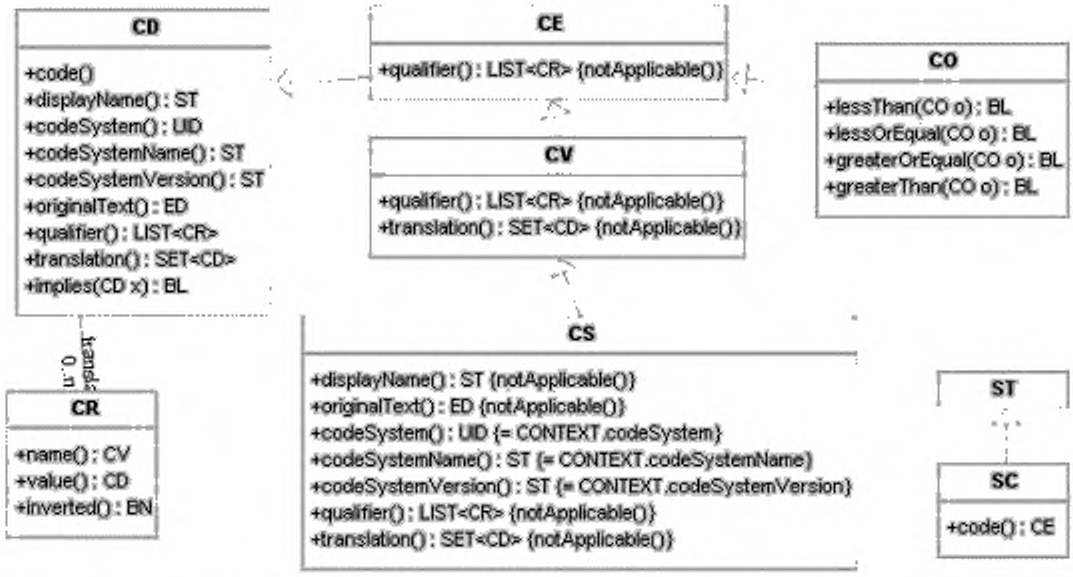


Рисунок В.4 — Информационная модель дескриптора понятия

**В.2.6 Тип данных ConceptDescriptor (CD) (специализация типа данных ANY)**

Определение: тип данных CD представляет любой вид понятия с помощью кода, определенного в системе кодирования. Значение типа данных CD может содержать исходный текст или фразу, используемые как основа для кодирования, а также одно или несколько преобразований в другие системы кодирования. Значение типа данных CD может содержать квалификаторы, позволяющие описать, к примеру, понятие «левая нога» как посткоординированный термин, составленный из основного кода «НОГА» и квалификатора «ЛЕВЫЙ». В исключительных значениях типа данных CD код отсутствует, а присутствует только исходный текст, описывающий данное понятие.

Таблица В.11 — Сводка свойств типа данных ConceptDescriptor

| Имя | Тип | Описание |
|---|---|---|
| code | ST | Символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9 |
| codeSystem | UID | Указывает систему кодирования, в которой определен код |
| codeSystemName | ST | Общее имя системы кодирования |
| codeSystemVersion | ST | Если применим, дескриптор версии, определенный специально для данной системы кодирования |
| displayName | ST | Имя или название кода, под которым система-отправитель показывает значение кода своим пользователям |
| originalText | ED | Текст или фраза, используемые в качестве основы для кодирования |
| translation | SET<CD> | Множество других дескрипторов понятия, которые преобразуют данный дескриптор понятия в другие системы кодирования |
| qualifier | LIST<CR> | Указывает дополнительные коды, которые повышают специфичность основного кода |

```
type ConceptDescriptor alias CD specializes ANY {
            ST          code;
            ST          displayName;
            UID         codeSystem;
            ST          codeSystemName;
            ST          codeSystemVersion;
            ED          originalText;
            LIST<CR>    qualifier;
            SET<CD>     translation;
            BL          equal(ANY x);
            BL          implies(CD x);
  demotion  ED;
};
```

Тип данных CD обычно используется в одной из своих ограниченных, или «профилированных» форм CS, CE, CV. Тип данных полного дескриптора понятия используется не часто. Для его применения необходимы осознанное решение и документированное обоснование. Во всех остальных случаях используется одно из ограничений типа данных CD[1)].

Все ограничения на тип данных CD накладываются на определенные свойства. Эти свойства могут быть ограничены до такой степени, что допустимым окажется единственное значение, после чего упоминание этого свойства станет избыточным. Ограничение свойства до единственного значения называется подавлением этого свойства. Хотя концептуально подавленное свойство продолжает быть семантически приемлемым, в любом интерфейсе HL7 можно вполне безопасно без проверки предполагать значение по умолчанию.

Примечание — Вообще говоря, сказанное верно для многих типов данных, определенных в настоящей спецификации, однако чаще всего возникают вопросы по поводу потомков типа данных CD.

B.2.6.1 Свойство code: ST

Определение: символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9.

Неисключительное значение типа данных CD имеет непустое свойство code, значением которого является строка символов, определенная в системе кодирования, идентифицированной в свойстве codeSystem. И наоборот, значение типа CD, у которого свойство code не имеет значения или имеет значение, не принадлежащее указанной системе кодирования, считается исключительным значением (пустым значением (NULL) с типом пустоты «other»).

```
invariant(CD x)
      where x.nonNull {
   x.code.nonNull;
};
```

B.2.6.2 Свойство codeSystem: UID

Определение: указывает систему кодирования, в которой определено значение свойства code.

Идентификатор системы кодирования должен иметь тип данных UID, позволяющий однозначно указать стандартные системы кодирования HL7, другие стандартные системы кодирования, а также местные системы кодирования. Комитет HL7 должен присваивать идентификатор типа UID каждой из своих таблиц кодов, а также внешним стандартным системам кодирования, которые используются в стандартах HL7. На местах должны использовать свои объектные идентификаторы ИСО (тип данных OID), с помощью которых можно сконструировать глобально уникальные идентификаторы местных систем кодирования.

Под ветвью комитета HL7, 2.16.840.1.113883, подветви 5 и 6 содержат соответственно идентификаторы стандартных систем кодирования HL7 и внешних систем кодирования. Эти ветви ведутся техническим комитетом HL7 Vocabulary Technical Committee.

Неисключительное значение типа CD (то есть значение типа CD с непустым свойством code) имеет непустое свойство codeSystem, указывающее систему понятий, в которой определено значение свойства code. Другими словами, если есть код, должна быть и система кодирования.

Примечание — Хотя для каждого непустого значения типа CD определена конкретная система кодирования, в некоторых обстоятельствах представление значения типа CD в соответствии со спецификацией

[1)] Преимуществом типа данных CD является его выразительность. Но постоянное использование всех его свойств, например исключений из кодирования, текста, преобразований и квалификаторов во всех реализациях становится очень затруднительным и небезопасным. Поэтому в большинстве случаев тип данных CD используется в ограниченной форме с меньшим числом свойств.

реализуемой технологии не нуждается в явном упоминании системы кодирования. Например, когда контекст подразумевает одну и только одну систему кодирования, то ее явное указание стало бы избыточным. Однако в таком случае свойство codeSystem принимает контекстно-зависимое значение по умолчанию и не является пустым.

```
invariant(CD x)
      where x.code.nonNull {
   x.codeSystem.nonNull;
};
```

Причина пустоты «other» у исключительного значения типа CD указывает, что понятие не может быть закодировано в указанной системе кодирования. Эта система кодирования, в которой нет такого исключительного понятия, должна быть указана в свойстве codeSystem.

Некоторые домены кодов квалифицированы таким образом, что они могут включать в себя некоторые части подходящей местной системы кодирования, не являющиеся парафразами стандартной системы кодирования (coded with extensibility, CWE — кодированные с расширением). Если поле с квалификатором «CWE» действительно содержит такой местный код, то в свойстве системы кодирования должен быть указан идентификатор местной системы кодирования, из которой взят этот код. Однако в доменах с квалификатором «CWE» местный код является допустимым членом домена, поэтому использование местного кода не является ни ошибкой, ни исключительным значением (с причиной пустоты «other») в смысле настоящей спецификации.

```
invariant(CD x)
      where x.other {
   x.code.other;
   x.codeSystem.nonNull;
};
```

B.2.6.3 Свойство codeSystemName: ST

Определение: общее имя системы кодирования.

Имя системы кодирования не используется для вычислений. Оно может быть указано для облегчения интерпретации человеком значений свойств code и codeSystem. Предполагается, хотя и не является обязательным, что в спецификациях реализуемой технологии будет предусматриваться указание значения свойства codeSystemName в качестве аннотации к идентификатору UID, рассчитанной на восприятие человеком.

Системы, соответствующие стандарту HL7, не должны функционально полагаться на значение свойства codeSystemName. Это значение не должно модифицировать значение свойства codeSystem и не может существовать без значения свойства codeSystem.

```
invariant(CD x) {
   x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.6.4 Свойство codeSystemVersion: ST

Определение: дескриптор версии, определенный специально для данной системы кодирования (если применим).

Для каждой внешней системы кодирования в стандарте HL7 должно быть указано, как формируется строка со значением версии. Если для конкретной системы кодирования такое указание в стандарте отсутствует, то для такой системы обозначение версии не имеет определенного значения.

Различные версии одной и той же системы кодирования должны быть совместимыми. Если система кодирования изменена несовместимым образом, то она представляет собой другую систему кодирования, а не другую версию, как бы издатель это ни называл.

Например, издатель классификаций МКБ-9 и МКБ-10 назвал эти системы кодирования «9-м пересмотром» и «10-м пересмотром». Однако МКБ-10 представляет собой полное изменение кодов МКБ и не является обратно совместимой. Поэтому в целях настоящей спецификации МКБ-9 и МКБ-10 рассматриваются как разные системы кодирования, а не как разные версии одной системы кодирования. Напротив, когда версия «1.0j» системы кодирования LOINC была обновлена до версии «1.0k», комитет HL7 рассматривал это как смену версий, поскольку новая версия была обратно совместимой с предыдущей.

```
invariant(CD x) {
   x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.6.5 Свойство displayName: ST

Определение: имя или название кода, под которым система-отправитель показывает значение кода своим пользователям.

Свойство displayName включено как для удобства интерпретации человеком значения кода, так и для документирования имени, используемого для изображения понятия пользователю. Оно не имеет функционального значения, не может существовать без кода и никогда не должно модифицировать смысл кода.

Примечания

1 В своих словарных доменах стандарты HL7 предусматривают атрибут «печатаемого имени» (print name). Значения этого атрибута могут использоваться в качестве значений свойства displayName.

2 Имена, предусмотренные для кодов, не могут менять смысл кодированного значения. Поэтому они не должны предоставляться пользователю прикладной системы-получателя, пока не будет уверенности в том, что такое имя адекватно отражает понятие, соответствующее кодированному значению. При коммуникациях нельзя просто полагаться на имя кодированного значения. Основная цель этого имени в обеспечении возможности отладки единиц данных в протоколах HL7 (например, сообщений).

```
invariant(CD x) {
   x.displayName.nonNull.implies(x.code.nonNull);
};
```

B.2.6.6 Свойство originalText: ED

Определение: текст или фраза, используемые в качестве основы для кодирования.

Исходный текст появляется в тех случаях, когда код был присвоен информации не ее источником, а специальным кодировщиком уже после ее появления (кодирование постфактум). Тогда при создании дескриптора понятия исходный текст может существовать без кода.

Примечание — Хотя кодирование постфактум часто осуществляется по информации, представленной как свободный текст, например в форме документов, сканированных изображений или диктовки, мультимедийные данные явным образом не разрешены в качестве исходного текста. Кроме того, свойство исходного текста originalText не может означать ссылку на весь исходный документ. Связи между различными артефактами медицинской информации (например, документом и кодированным результатом) не входят в область применения настоящей спецификации и рассматриваются в других стандартах HL7. Исходный текст должен представлять собой извлечение из оригинального источника, а не точную копию его содержания или указатель на этот источник. Поэтому исходный текст должен представляться в неформатированном виде.

Значения типа CD могут иметь непустое свойство исходного текста, даже если свойство code имеет пустое значение. В этом случае свойство originalText является именем или описанием понятия, которое не было закодировано. Такие исключительные значения типа CD могут также содержать преобразования translation. Такие преобразования обеспечивают непосредственное кодирование понятия, описанного в свойстве originalText.

Тип данных CD может быть понижен до типа данных ST. В этом случае значение типа ST будет представлено только свойством originalText значения типа CD.

```
invariant(CD x)
      where x.originalText.nonNull {
   ((ST)x).equal(x.originalText);
};
```

B.2.6.7 Свойство translation: SET<CD>

Определение: множество других дескрипторов понятия, которые преобразуют данный дескриптор понятия в другие системы кодирования.

Свойство translation представляет собой множество других значений типа CD, каждое из которых преобразует первое значение типа CD в другую систему кодирования. Каждый элемент множества преобразований был получен с помощью преобразования первого значения типа CD. Каждое преобразование, в свою очередь, может также содержать преобразования. Таким образом, если код многократно преобразуется, то информация о том, какой код послужил входным параметром каждого преобразования, сохранится.

Примечание — Преобразования являются квазисинонимами одного понятия реального мира. Предполагается, что каждое преобразование в этом множестве выражает то же самое понятие «другими словами». Однако между двумя структурно различными системами кодирования редко существует точная синонимия. Поэтому не все преобразования будут абсолютно точными.

B.2.6.8 Свойство qualifier: LIST<CR>

Определение: указывает дополнительные коды, которые повышают специфичность основного кода.

Основной код и все его квалификаторы (qualifier) в совокупности идентифицируют одно понятие. Значение типа CD с квалификаторами называется также кодовой фразой или посткоординируемым выражением.

Квалификаторы ограничивают значение основного кода, но их применение не должно приводить к отрицанию значения основного кода или к такому его изменению, при котором будет получено другое значение из основной системы кодирования.

Квалификаторы могут использоваться только в сочетании с хорошо определенными правилами посткоординации. Значение типа CD может иметь квалификаторы только в том случае, когда его система кодирования допускает использование или когда существует третья система кодирования, в которой указано, каким образом могут сочетаться другие системы кодирования.

Например, номенклатура медицинских терминов SNOMED CT позволяет конструировать понятие с помощью сочетания нескольких кодов. В этой номенклатуре определены понятие «целлюлит (нарушение)» (128045006), атрибут «локализация» (363698007) и другое понятие «структура ноги (структура тела)» (56459004). Номенклатура SNOMED CT позволяет создать кодовую фразу из этих кодов.

***Пример 1***

```
<observation>
   ...
   <value code="128045006" codeSystem="&amp;SNOMED-CT;" displayName="целлюлит (нару-
шение">
      <qualifier code="56459004" displayName="структура ноги (структура тела)">
         <name code="363698007" displayName="локализация"/>
      </qualifier>
   </value>
   ...
</observation>
```

В данном примере в одной системе кодирования, SNOMED-CT, определены и основной код, и все квалификаторы, а также правила их сочетания. Поэтому в нем не требуется отдельно упоминать систему кодирования codeSystem для имени и значений квалификаторов (значение свойства codeSystem наследуется от основного кода).

Важно отметить, что все допустимые квалификаторы указываются системой кодирования. Например, в номенклатуре SNOMED CT определено множество квалифицирующих атрибутов, при этом только исследования (Finding) и нарушения (Disorder) могут квалифицироваться атрибутом «локализация» (finding site). Использование квалификаторов за пределами границ, установленными системой кодирования, является несовместимым использованием типа данных CD. Следуя правилам, описанным в системе кодирования, можно сравнить посткоординированные выражения с прекоординированными понятиями (например, можно сравнить приведенную выше кодовую фразу с прекоординированным понятием «целлюлит ноги (нарушение)» (128276007), которое определено в номенклатуре SNOMED CT как относящееся к локализации «структура ноги»). В типе данных CD не предусмотрена нормализация композиции кодов, поэтому могут создаваться неоднозначные кодовые фразы. Следует иметь в виду, что для создания однозначных представлений данных с помощью композиции кодов с типом CD необходимо наложить на композицию дополнительные ограничения. Иначе может оказаться, что извлечение полного набора записей, удовлетворяющих определенному запросу, окажется невозможным.

Другим общим примером могут служить коды процедур, используемые организацией U.S. Centers for Medicare and Medicaid Services (CMS) (ранее известной под названием Health Care Financing Administration, HCFA). Коды процедур, используемые организацией CMS (HCPCS), основаны на классификации CPT-4 и добавляют к ней различные квалификаторы. Например, пациенту с указанным выше нарушением (страдающему также заболеванием периферических артерий, сахарным диабетом и хроническим поражением кожи на большом пальце левой стопы) могли ампутировать этот палец. Для кодирования этой процедуры используется понятие «Ампутация, плюснефаланговый сустав стопы» (код понятия 28820), определенное в классификации CPT-4 (28820), и квалификатор «большой палец левой стопы» с кодом TA из системы HCPCS. Ниже показан пример сочетания этого кода с квалификатором.

***Пример 2***

```
<procedure>
   ...
   <cd code="28820" codeSystem="&amp;CP4;" displayName=" Ампутация, плюснефаланговый
сустав стопы">
      <qualifier code="TA" codeSystem="&amp;HCP;" displayName="большой палец левой
стопы"/>
   </cd>
   ...
</procedure>
```

В этом примере система кодирования квалификатора (HCPCS) отличается от системы кодирования основного кода (CPT-4). Этот квалификатор может использоваться только в силу наличия хорошо определенных правил сочетания этих кодов. Учтите также, что имя роли квалификатора не является обязательным, и для кодов HCPCS имена ролей не назначаются.

Порядок применения квалификаторов сохраняется, особенно в случае, когда система кодирования позволяет посткоординацию, но не определяет имена ролей, как это имеет место для некоторых кодов клинической модификации МКБ-9 в США (ICD-9CM) или для «многоосевого» кодирования в старой версии номенклатуры SNOMED (multiaxial).

B.2.6.9 Свойство equality: BL (унаследовано от типа данных ANY)

Основными применениями дескрипторов понятий являются индексирование, запросы и принятие решения на основе кодированного значения. Поэтому для семантически однозначной спецификации кодированных значений требуется точное определение того, что означает равенство дескрипторов понятий и как можно сравнивать значения типа CD. (Дополнительные сведения о сравнении пре- и посткоординированных выражений см. в публикации Dolin RH, Spackman KA, Markwell D. Selective Retrieval of Pre- and Post-coordinated SNOMED Concepts. Fall AMIA 2002; 210-14, или в руководстве SNOMED CT Implementation Guide, July 2003.)

Равенство двух значений типа CD определяется исключительно по значениям свойств code и codeSystem. Свойство codeSystemVersion из проверки на равенство исключается[1]. Если присутствуют квалификаторы, то они учитываются при проверке на равенство. Преобразования не включаются в проверку[2]. Исключительные значения типа CD не считаются равными даже в том случае, если они имеют одну и ту же причину пустоты или один и тот же исходный текст[3].

```
invariant(CD x, y)
      where x.nonNull.and(y.nonNull) {
   x.equal(y).equal(x.code.equal(y.code)
             .and(x.codeSystem.equal(y.codingSystem))
             .and(x.qualifier.equal(y.qualifier))};
);
```

В некоторых системах кодирования определены разные стили представления кодовых значений. Например, в системе идентификации лекарственных средств U.S. National Drug Code (NDC) предусмотрены две формы: с дефисами и без дефисов. Примером формы с дефисами служит код 1234-5678-90, а формы без дефисов — 01234567890. Другим примером подобной проблемы служат некоторые таблицы кодов ИСО или ANSI, в которых определены необязательные алфавитно-цифровые и числовые формы кода из двух или трех символов для всех кодируемых значений.

В том случае, когда система кодирования предусматривает несколько представлений, комитетом HL7 должно быть принято решение о предпочтительной форме. Комитет должен документировать это решение, если им рекомендована соответствующая внешняя система кодирования. Решение о предпочтительной форме должно приниматься с учетом практичности и широты использования. Если четких критериев практичности и широты использования нет, то должно быть отдано предпочтение наиболее безопасной, расширяемой и наименее стилизованной (наименее декорированной) форме[4].

---

[1] Свойство codeSystemVersion не учитывается при проверке равенства, поскольку по определению символ кода должен иметь одно и то же значение во всех версиях системы кодирования. При переходе к другой версии код может быть объявлен устаревшим, но не может быть удален или повторно использован для другого понятия.

[2] Преобразования не включаются в проверку на равенство по соображениям безопасности. Альтернативой могло быть признание двух значений типа CD равными, если равны какие-либо из их преобразований. Однако некоторые преобразования могут оказаться одинаковыми, поскольку их система кодирования является очень грубой. Более интеллектуальные сравнения дескрипторов зависят от приложений и не обсуждаются в настоящей спецификации.

[3] Пустые значения (NULL) являются исключительными значениями, а не допустимыми понятиями. Было бы небезопасно приравнивать два значения только на том основании, что они оба исключительны (например, не закодированы или неизвестны). Аналогично, нет никакой гарантии, что исходный текст представляет собой уникальное описание понятия, поэтому совпадение исходных текстов не означает равенство понятий. Верно и обратное: так как одно и то же понятие может быть описано с помощью разных исходных текстов, то несовпадение текстов не означает, что понятия различаются.

[4] Это решение, принимаемое на этапе конструирования, необходимо, чтобы интерфейсы, предлагаемые в стандартах HL7, не были отягощены необходимостью реализации преобразований различных стилей литералов кодов во время исполнения, хотя это и может привести к ситуации, когда некоторым приложениям может потребоваться выполнять преобразования из одной формы представления кодов в другую, если они рассчитаны на вариант представления, не выбранный комитетом HL7.

B.2.6.10 Свойство implies: BL

Определение: указывает, является ли данное значение типа CD специализацией операнда типа CD.

Естественно, понятия могут делаться более узкими или более широкими, чтобы исключить или охватить другие понятия. Во многих системах кодирования обеспечиваются явные указания специализации и обобщения понятий. Принципы построения словаря HL7 также предусматривают специализацию понятий в наборах значений, определенных в стандартах HL7. Свойство implies (импликация) представляет собой предикат, устанавливающий, является ли одно понятие специализацией другого понятия и, следовательно, выводится из этого другого понятия.

При составлении предикатов (например, условных операторов), сравнивающих два кода, обычно проверяется не равенство кодов, а возможность вывода одного из другого.

Например, в таблице B.20, описывающей понятия «использования телекоммуникаций» определены следующие значения: «W» (рабочий), «H» (домашний), «HP» (основной домашний) и «HV» (домашний на время отпуска). Очевидно, что из обоих значений «HP» и «HV» вытекает значение «H». При поиске какого-либо номера домашнего телефона необходимо установить, вытекает ли из данного значения кода использования телекоммуникаций значение «H». Если бы выполнялась проверка равенства коду «H», то были бы найдены только домашние номера без специализации, а не номер основного домашнего телефона.

С вычислительной точки зрения импликация может быть установлена одним из двух способов. Литералы в системе кодирования могут быть устроены таким образом, что иерархия может задаваться в самом литерале кода (как это имеет место в классификации МКБ-9). Но за исключением таких особых случаев для вычисления выражений, содержащих импликацию, потребуется терминологическая база знаний и алгоритм проверки поглощения понятий. Для посткоординированных систем кодирования разработка такого алгоритма представляет собой нетривиальную задачу[1)].

**B.2.7 Тип данных ConceptRole (CR) (специализация типа данных ANY)**

Код квалификатора понятия с необязательной именованной ролью. Как роль квалификатора, так и коды значений должны быть определены в системе кодирования значений типа CD, содержащей квалификатор понятия. Например, если в номенклатуре SNOMED RT определены понятие «нога», отношение роли «имеет-сторону» и еще одно понятие «левая», то отношение роли понятия позволяет добавить квалификатор «имеет-сторону: левая» к основному коду «нога» и тем самым получить понятие «левая нога».

Таблица B.12 — Сводка свойств типа данных ConceptRole

| Имя | Тип | Описание |
|---|---|---|
| name | CV | Указывает способ, которым роль понятия вносит вклад в значение кодовой фразы. Например, если в номенклатуре SNOMED RT определены понятие «нога», отношение роли «имеет-сторону» и еще одно понятие «левая», то отношение роли понятия позволяет добавить квалификатор «имеет-сторону: левая» к основному коду «нога» и тем самым получить понятие «левая нога». В данном примере свойство name (имя) имеет значение «имеет-сторону» |
| value | CD | Понятие, которое модифицирует основной код кодовой фразы с помощью отношения роли. Например, если в номенклатуре SNOMED RT определены понятие «нога», отношение роли «имеет-сторону» и еще одно понятие «левая», то отношение роли понятия позволяет добавить квалификатор «имеет-сторону: левая» к основному коду «нога» и тем самым получить понятие «левая нога». В этом примере свойство value имеет значение «левая» |
| inverted | BN | Указывает, что смысл имени меняется на противоположный. Такое обращение смысла можно использовать в тех случаях, когда в соответствующей системе кодирования определено обращение, но не предусмотрена взаимно противоположная пара имен. По умолчанию свойство inverted имеет значение «false» |

Применение квалификаторов строго контролируется используемой системой кодирования. Определение типа данных CD не разрешает использовать квалификаторы вместе с системами кодирования, где они не предусмотрены (например, с прекоординированными системами, скажем, LOINC, ICD-10 PCS).

```
protected type ConceptRole alias CR specializes ANY {
   CV  name;
   BN  inverted;
   CD  value;
};
```

[1)] Это одна из причин, почему использование квалификаторов в посткоординированных системах кодирования должно быть умеренным и осторожным. Другая проблема посткоординации состоит в том, что общее правило равенства может вообще не существовать.

B.2.7.1 Свойство name: CV

Определение: указывает способ, которым роль понятия вносит вклад в значение кодовой фразы. Например, если в номенклатуре SNOMED RT определены понятие «нога», отношение роли «имеет-сторону» и еще одно понятие «левая», то отношение роли понятия позволяет добавить квалификатор «имеет-сторону: левая» к основному коду «нога» и тем самым получить понятие «левая нога». В данном примере свойство name (имя) имеет значение «имеет-сторону».

Если родительская система кодирования CD.codeSystem позволяет посткоординацию, но не предусматривает имена ролей (например, SNOMED), то свойство name может быть пустым (NULL).

B.2.7.2 Свойство value: CD

Определение: понятие, которое модифицирует основной код кодовой фразы с помощью отношения роли. Например, если в номенклатуре SNOMED RT определены понятие «нога», отношение роли «имеет-сторону» и еще одно понятие «левая», то отношение роли понятия позволяет добавить квалификатор «имеет-сторону: левая» к основному коду «нога» и тем самым получить понятие «левая нога». В данном примере свойство value имеет значение «левая».

Свойство value имеет тип данных CD и, следовательно, само может иметь квалификаторы. Таким образом, квалификаторы могут быть вложенными. Квалификаторы могут использоваться только в тех случаях, когда возможность их применения указана в соответствующей системе кодирования. Никакие виды квалификаторов нельзя применять для систем кодирования, в описании которых явно не указаны применение квалификаторов и правила их применения.

```
invariant(CR x)
      where x.nonNull {
   x.value.nonNull;
};
```

B.2.7.3 Свойство inverted: BN

Определение: указывает, что смысл имени меняется на противоположный. Такое обращение смысла можно использовать в тех случаях, когда в соответствующей системе кодирования определено обращение, но не предусмотрена взаимно противоположная пара имен. По умолчанию свойство inverted имеет значение «false».

Например, в системах кодирования могут быть определены отношение роли «вызывает заболевание» и понятия «Streptococcus pneumoniae» и «Пневмония». Если в этой системе кодирования разрешено обращение ролей, то можно сконструировать посткоординированное понятие «Пневмококковая пневмония» с помощью следующей кодовой фразы: «Пневмония», обращение понятия «вызывает заболевание» «Streptococcus pneumoniae».

Роли могут быть обращены, если в соответствующей системе кодирования определено такое обращение. Но если система кодирования определяет роли в обращаемых парах или намеренно не определяет некоторые обращения, то вместо обращения надо использовать подходящий код роли (например, «заболевание вызывается» вместо обращения роли «вызывает заболевание»). Обязательно должно быть известно, принимает ли свойство inverted значение «true» или «false», поскольку если оно пусто, то роль нельзя интерпретировать.

Примечание — Свойство inverted должно передаваться как атрибут признака, который по умолчанию имеет значение «false». В этом случае признак обращения роли не должен передаваться, если роль не обращена.

**B.2.8 Тип данных CodedSimpleValue (CS) (специализация типа данных CV)**

Определение: кодированные данные в их простейшей форма, в которой только свойство code не имеет предопределенных значений. Система кодирования и версия системы кодирования однозначно определяются контекстом значения типа CS. Тип данных CS используется для кодированных атрибутов, значения которых берутся из одного набора данных, определенного в стандарте HL7.

Таблица B.13 — Сводка свойств типа данных CodedSimpleValue

| Имя | Тип | Описание |
|---|---|---|
| code | ST | Символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9 |

```
type CodedSimpleValue alias CS specializes CV {
          ST  code;
   literal  ST;
};
```

Тип данных CS может использоваться только в следующих случаях:

- для кодированного атрибута, с которым связана единственная система кодирования, определенная комитетом HL7, и добавление кодов в которую требует от комитета формальных действий (например, гармонизации). Такие кодированные атрибуты должны иметь тип данных CS;

- для свойства, определенного в настоящей спецификации, которому присвоена единственная система кодирования, указанная в данной спецификации либо определенная вне комитета HL7 органом, ответственным за это понятие и ведение системы кодирования.

Например, организация IETF ведет систему кодирования типов среды MIME, используемую в типе данных ED. Таким образом, спецификация этого типа данных допускает также и предусмотренный этой организацией механизм расширения типов среды MIME (к примеру, «application/x-myapp»).

Для значений типа CS квалификатор расширяемости всегда имеет значение «CNE» (coded, non-extensible — кодируемый, не расширяемый) и по контексту определяется, какие можно использовать значения, определенные в стандарте HL7[1)].

B.2.8.1 Свойство code: ST (унаследовано от типа данных CD)

```
invariant(CS x)
      where x.nonNull {
   x.code.nonNull;
};
```

B.2.8.2 Свойство codeSystem: UID (фиксированное)

С каждым непустым значением типа данных CS связана определенная система кодирования. Она не обязана быть явно указана в представлении типа данных CS, используемом в спецификации реализуемой технологии, поскольку по контексту определяется одна и только одна используемая система кодирования. Ее явное указание избыточно. Но при этом для нее подразумевается определенное непустое контекстно-специфическое значение.

```
invariant(CS x)
      where x.code.nonNull {
   x.codeSystem.nonNull;
   x.codeSystem.equal(CONTEXT.codeSystem);
};
```

Исключительное значение типа CS с кодом причины пустоты «other» означает, что данное понятие не может быть закодировано в указанной системе кодирования. В этих случаях свойство code должно иметь пустое значение NULL.

```
invariant(CS x)
      where x.other {
   x.code.isNull;
   x.codeSystem.nonNull;
};
```

B.2.8.3 Свойство codeSystemName: ST (фиксированное)

```
invariant(CS x) {
   x.codeSystemName.equal(CONTEXT.codeSystemName);
};
```

B.2.8.4 Свойство codeSystemVersion: ST (фиксированное)

```
invariant(CS x) {
   x.codeSystemVersion.equal(CONTEXT.codeSystemVersion);
};
```

B.2.8.5 Свойство displayName: ST (фиксированное)

```
invariant(CS x) {
   x.displayName.notApplicable;
};
```

[1)] Это не противоречит тому, что используемый по ссылке внешний домен, например типы среды IETF MIME, может иметь механизм расширения. Эти расширенные коды типов среды MIME не должны рассматриваться как «расширения», нарушающие квалификатор расширяемости CNE. Нарушение будет иметь место в том случае, если будет сделана попытка использовать другую систему кодирования (посредством свойства CD.codeSystem), что невозможно сделать для типа данных CS.

B.2.8.6 Свойство originalText: ED (фиксированное)

```
invariant(CS x) {
   x.originalText.notApplicable;
};
```

B.2.8.7 Свойство translation: SET<CD> (фиксированное)

```
invariant(CS x) {
   x.translation.notApplicable;
};
```

B.2.8.8 Свойство qualifier: LIST<CR> (фиксированное)

```
invariant(CS x) {
   x.qualifier.notApplicable;
};
```

B.2.8.9 Литеральная форма

```
CS.literal ST {
   ST : /[a-zA-ZC-9_]+/  { $.equal($1); };
};
```

Строковая форма литеральной формы типа данных CS определена в основном для целей настоящей спецификации. Она является строковым представлением кода из системы кодирования, заданной контекстом, в котором используется тип данных CS. По самому литералу нельзя определить систему кодирования codeSystem или ее версию codeSystemVersion, поэтому литерал можно использовать, только если известен контекст.

**B.2.9 Тип данных CodedValue (CV) (специализация типа данных CE)**

Определение: кодированные данные, для которых указаны только код, система кодирования, а также необязательные имя для вывода на экран и исходный текст. Используется только как тип свойств других типов данных.

Таблица B.14 — Сводка свойств типа данных CodedValue

| Имя | Тип | Описание |
|---|---|---|
| code | ST | Символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9 |
| codeSystem | UID | Указывает систему кодирования, в которой определен код |
| codeSystemName | ST | Общее имя системы кодирования |
| codeSystemVersion | ST | Если применим, дескриптор версии, определенный специально для данной системы кодирования |
| displayName | ST | Имя или название кода, под которым система-отправитель показывает значение кода своим пользователям |
| originalText | ED | Текст или фраза, используемые в качестве основы для кодирования |

```
type CodedValue alias CV specializes CE {
   ST   code;
   UID  codeSystem;
   ST   codeSystemName;
   ST   codeSystemVersion;
   ST   displayName;
   ED   originalText;
};
```

В наиболее приемлемом сценарии использования типа данных CV требуется возвратить единственный код. Поэтому этот тип данных не должен использоваться, если для данного значения желательно передать несколько

альтернативных кодов. Этот тип данных может использоваться как с квалификатором расширяемости домена «CNE» (coded, non-extensible — кодируемый, не расширяемый), так с квалификатором «CWE» (coded — кодируемый, расширяемый).

B.2.9.1 Свойство code: ST (унаследовано от типа данных CD)

Определение: символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9.

Неисключительное значение типа данных CV имеет непустое свойство code, значением которого является строка символов, определенная в системе кодирования, идентифицированной в свойстве codeSystem. И наоборот, значение типа CV, у которого свойство code не имеет значения или имеет значение, не принадлежащее указанной системе кодирования, считается исключительным значением (пустым значением (NULL) с типом пустоты «other»).

```
invariant(CV x)
      where x.nonNull {
   x.code.nonNull;
};
```

B.2.9.2 Свойство codeSystem: UID (унаследовано от типа данных CD)

Определение: указывает систему кодирования, в которой определено значение свойства code.

Идентификатор системы кодирования должен иметь тип данных UID, позволяющий однозначно указать стандартные системы кодирования HL7, другие стандартные системы кодирования, а также местные системы кодирования. Комитет HL7 должен присваивать идентификатор типа UID каждой из своих таблиц кодов, а также внешним стандартным системам кодирования, которые используются в стандартах HL7. На местах должны использовать свои объектные идентификаторы ИСО (тип данных OID), с помощью которых можно сконструировать глобально уникальные идентификаторы местных систем кодирования.

Под ветвью комитета HL7, 2.16.840.1.113883, подветви 5 и 6 содержат соответственно идентификаторы стандартных систем кодирования HL7 и внешних систем кодирования. Эти ветви ведутся техническим комитетом HL7 Vocabulary Technical Committee.

Неисключительное значение типа CD (то есть значение типа CD с непустым свойством code) имеет непустое свойство codeSystem, указывающее систему понятий, в которой определено значение свойства code. Другими словами, если есть код, должна быть и система кодирования.

Примечание — Хотя для каждого непустого значения типа CD определена конкретная система кодирования, в некоторых обстоятельствах представление значения типа CD в соответствии со спецификацией реализуемой технологии не нуждается в явном упоминании системы кодирования. Например, когда контекст подразумевает одну и только одну систему кодирования, то ее явное указание стало бы избыточным. Однако в таком случае свойство codeSystem принимает контекстно-зависимое значение по умолчанию и не является пустым.

```
invariant(CV x)
      where x.code.nonNull {
   x.codeSystem.nonNull;
};
```

Причина пустоты «other» у исключительного значения типа CD указывает, что понятие не может быть закодировано в указанной системе кодирования. Эта система кодирования, в которой нет такого исключительного понятия, должна быть указана в свойстве codeSystem.

Некоторые домены кодов квалифицированы таким образом, что они могут включать в себя некоторые части подходящей местной системы кодирования, не являющиеся парафразами стандартной системы кодирования (coded with extensibility, CWE — кодированные с расширением). Если поле с квалификатором «CWE» действительно содержит такой местный код, то в свойстве системы кодирования должен быть указан идентификатор местной системы кодирования, из которой взят этот код. Однако в доменах с квалификатором «CWE» местный код является допустимым членом домена, поэтому использование местного кода не является ни ошибкой, ни исключительным значением (с причиной пустоты «other») в смысле настоящей спецификации.

```
invariant(CV x)
      where x.other {
   x.code.other;
   x.codeSystem.nonNull;
};
```

B.2.9.3 Свойство codeSystemName: ST (унаследовано от типа данных CD)

Определение: общераспространенное имя системы кодирования.

Имя системы кодирования не используется для вычислений. Оно может быть указано для облегчения интерпретации человеком значений свойств code и codeSystem. Предполагается, хотя и не является обязательным, что в спецификациях реализуемой технологии будет предусматриваться указание значения свойства codeSystemName в качестве аннотации к идентификатору UID, рассчитанной на восприятие человеком.

Системы, соответствующие стандарту HL7, не должны функционально полагаться на значение свойства codeSystemName. Это значение не должно модифицировать значение свойства codeSystem и не может существовать без значения свойства codeSystem.

```
invariant(CV x) {
   x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.9.4 Свойство codeSystemVersion: ST (унаследовано от типа данных CD)

Определение: дескриптор версии, определенный специально для данной системы кодирования (если применим).

Для каждой внешней системы кодирования в стандарте HL7 будет указано, как формируется строка со значением версии. Если для конкретной системы кодирования такое указание в стандарте отсутствует, то для такой системы обозначение версии не имеет определенного значения.

Различные версии одной и той же системы кодирования должны быть совместимыми. Если система кодирования изменена несовместимым образом, то она представляет собой другую систему кодирования, а не другую версию, как бы издатель это ни называл.

Например, издатель классификаций МКБ-9 и МКБ-10 назвал эти системы кодирования «9-м пересмотром» и «10-м пересмотром». Однако МКБ-10 представляет собой полное изменение кодов МКБ, и не является обратно совместимой. Поэтому в целях настоящей спецификации МКБ-9 и МКБ-10 рассматриваются как разные системы кодирования, а не как разные версии одной системы кодирования. Напротив, когда версия «1.0j» системы кодирования LOINC была обновлена до версии «1.0k», комитет HL7 рассматривал это как смену версий, поскольку новая версия была обратно совместимой с предыдущей.

```
invariant(CV x) {
   x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.9.5 Свойство displayName: ST (унаследовано от типа данных CD)

Имя или название кода, под которым система-отправитель показывает значение кода своим пользователям.

Свойство displayName включено как для удобства интерпретации человеком значения кода, так и для документирования имени, используемого для изображения понятия пользователю. Оно не имеет функционального значения, не может существовать без кода и никогда не должно модифицировать смысл кода.

Примечания

1 В своих словарных доменах стандарты HL7 предусматривают атрибут «печатаемого имени» (print name). Значения этого атрибута могут использоваться в качестве значений свойства displayName.

2 Имена, предусмотренные для кодов, не могут менять смысл кодированного значения. Поэтому они не должны предоставляться пользователю прикладной системы-получателя, пока не будет уверенности в том, что такое имя адекватно отражает понятие, соответствующее кодированному значению. При коммуникациях нельзя просто полагаться на имя кодированного значения. Основная цель этого имени в обеспечении возможности отладки единиц данных в протоколах HL7 (например, сообщений).

```
invariant(CV x) {
   x.displayName.nonNull.implies(x.code.nonNull);
};
```

B.2.9.6 Свойство originalText: ED

Определение: текст или фраза, используемые в качестве основы для кодирования.

Исходный текст появляется в тех случаях, когда код был присвоен информации не ее источником, а специальным кодировщиком уже после ее появления (кодирование постфактум). Тогда при создании дескриптора понятия исходный текст может существовать без кода.

Примечание — Хотя кодирование постфактум часто осуществляется по информации, представленной как свободный текст, например в форме документов, сканированных изображений или диктовки, мультимедийные данные явным образом не разрешены в качестве исходного текста. Кроме того, свойство исходного текста originalText не может означать ссылку на весь исходный документ. Связи между различными артефактами медицинской информации (например, документом и кодированным результатом) не входят в область применения

настоящей спецификации и рассматриваются в других стандартах HL7. Исходный текст должен представлять собой извлечение из оригинального источника, а не точную копию его содержания или указатель на этот источник. Поэтому исходный текст должен представляться в неформатированном виде.

Значения типа CV могут иметь непустое свойство исходного текста, даже если свойство code имеет пустое значение и, следовательно, является исключительным. В этом случае свойство originalText является именем или описанием понятия, которое не было закодировано. Такие исключительные значения могут также содержать преобразования translation. Такие преобразования обеспечивают непосредственное кодирование понятия, описанного в свойстве originalText.

Тип данных CV может быть понижен до типа данных ST. В этом случае значение типа ST будет представлено только свойством originalText значения типа CV.

```
invariant(CV x)
      where x.originalText.nonNull {
   ((ST)x).equal(x.originalText);
};
```

B.2.9.7 Свойство translation: SET<CD> (фиксированное)

```
invariant(CV x) {
   x.translation.notApplicable;
};
```

B.2.9.8 Свойство qualifier: LIST<CR> (фиксированное)

```
invariant(CV x) {
   x.qualifier.notApplicable;
};
```

**B.2.10 Тип данных CodedOrdinal (CO) (специализация типа данных CV)**

Определение: кодированные данные, принадлежащие системе кодирования с упорядоченными кодами. В типе данных CO добавлена семантика упорядоченности; таким образом, в моделях, использующих такие домены, могут быть определены элементы, содержащие утверждения о порядке следования терминов в домене.

```
type CodedOrdinal alias CO specializes CV {
   BL  lessOrEqual(CO o);
   BL  lessThan(CO o);
   BL  greaterThan(CO o);
   BL  greaterOrEqual(CO o);
};
```

Относительный порядок значений типа CO не обязан очевидно вытекать из их литерального представления. Предполагается, что приложение будет определять упорядоченность этих значений по некоторой таблице.

B.2.10.1 Свойство lessOrEqual: BL (отношение «меньше или равно»)

Определение: отношение порядка основано на свойстве lessOrEqual, которое считается примитивным в настоящей спецификации.

Из него могут быть выведены все другие отношения порядка. Так как свойство lessOrEqual является примитивным, то оно может задавать и частичную упорядоченность.

Отношения порядка обычно распространяются только внутри одной системы кодирования.

B.2.10.2 Свойство lessThan: BL (отношение «меньше»)

```
invariant(CO x, y)
      where x.nonNull.and(y.nonNull) {
   x.lessThan(y).equal(y.lessOrEqual(x).and(x.equal(y).not));
};
```

B.2.10.3 Свойство greaterThan: BL (отношение «больше»)

```
invariant(CO x, y)
      where x.nonNull.and(y.nonNull) {
   x.greaterThan(y).equal(y.lessThan(x));
};
```

B.2.10.4 Свойство greaterOrEqual: BL (отношение «больше или равно»)

```
invariant(CO x, y)
      where x.nonNull.and(y.nonNull) {
   x.greaterOrEqual(y).equal(y.lessOrEqual(x));
};
```

**B.2.11 Тип данных CodedWithEquivalents (CE) (специализация типа данных CD)**

Определение: кодированные данные, состоящие из кодированного значения и из необязательных кодированных значений, идентифицирующих то же самое понятие. Этот тип данных используется, когда могут существовать альтернативные коды.

Таблица B.15 — Сводка свойств типа данных CodedWithEquivalents

| Имя | Тип | Описание |
|---|---|---|
| code | ST | Символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9 |
| codeSystem | UID | Указывает систему кодирования, в которой определен код |
| codeSystemName | ST | Общее имя системы кодирования |
| codeSystemVersion | ST | Если применим, дескриптор версии, определенный специально для данной системы кодирования |
| displayName | ST | Имя или название кода, под которым система-отправитель показывает значение кода своим пользователям |
| originalText | ED | Текст или фраза, используемые в качестве основы для кодирования |
| translation | SET<CD> | Множество других дескрипторов понятия, которые преобразуют данный дескриптор понятия в другие системы кодирования |

```
type CodedWithEquivalents alias CE specializes CD {
   ST       code;
   UID      codeSystem;
   ST       codeSystemName;
   ST       codeSystemVersion;
   ST       displayName;
   ED       originalText;
   SET<CV>  translation;
};
```

Тип данных CE используется в сценариях, когда могут существовать альтернативные коды, полезные для получателя. Экземпляр типа данных CE содержит основное значение кода и может содержать множество альтернативных или эквивалентных представлений.

B.2.11.1 Свойство code: ST (унаследовано от типа данных CD)

Определение: символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9.

Неисключительное значение типа данных CD имеет непустое свойство code, значением которого является строка символов, определенная в системе кодирования, идентифицированной в свойстве codeSystem. И наоборот, значение типа CD, у которого свойство code не имеет значения или имеет значение, не принадлежащее указанной системе кодирования, считается исключительным значением (пустым значением (NULL) с типом пустоты «other»).

```
invariant(CE x)
      where x.nonNull {
   x.code.nonNull;
};
```

B.2.11.2 Свойство codeSystem: UID (унаследовано от типа данных CD)

Определение: указывает систему кодирования, в которой определено значение свойства code.

Идентификатор системы кодирования должен иметь тип данных UID, позволяющий однозначно указать стандартные системы кодирования HL7, другие стандартные системы кодирования, а также местные системы кодирования. Комитет HL7 должен присваивать идентификатор типа UID каждой из своих таблиц кодов, а также

внешним стандартным системам кодирования, которые используются в стандартах HL7. На местах должны использовать свои объектные идентификаторы ИСО (тип данных OID), с помощью которых можно сконструировать глобально уникальные идентификаторы местных систем кодирования.

Под ветвью комитета HL7, 2.16.840.1.113883, подветви 5 и 6 содержат соответственно идентификаторы стандартных систем кодирования HL7 и внешних систем кодирования. Эти ветви ведутся техническим комитетом HL7 Vocabulary Technical Committee.

Неисключительное значение типа CE (то есть значение типа CE с непустым свойством code) имеет непустое свойство codeSystem, указывающее систему понятий, в которой определено значение свойства code. Другими словами, если есть код, должна быть и система кодирования.

Примечание — Хотя для каждого непустого значения типа CE определена конкретная система кодирования, в некоторых обстоятельствах представление значения типа CE в соответствии со спецификацией реализуемой технологии не нуждается в явном упоминании системы кодирования. Например, когда контекст подразумевает одну и только одну систему кодирования, то ее явное указание стало бы избыточным. Однако в таком случае свойство codeSystem принимает контекстно-зависимое значение по умолчанию и не является пустым.

```
invariant(CE x)
      where x.code.nonNull {
   x.codeSystem.nonNull;
};
```

Причина пустоты «other» у исключительного значения типа CE указывает, что понятие не может быть закодировано в указанной системе кодирования. Эта система кодирования, в которой нет такого исключительного понятия, должна быть указана в свойстве codeSystem.

Некоторые домены кодов квалифицированы таким образом, что они могут включать в себя некоторые части подходящей местной системы кодирования, не являющиеся парафразами стандартной системы кодирования (coded with extensibility, CWE — кодированные с расширением). Если поле с квалификатором «CWE» действительно содержит такой местный код, то в свойстве системы кодирования должен быть указан идентификатор местной системы кодирования, из которой взят этот код. Однако в доменах с квалификатором «CWE» местный код является допустимым членом домена, поэтому использование местного кода не является ни ошибкой, ни исключительным значением (с причиной пустоты «other») в смысле настоящей спецификации.

```
invariant(CE x)
      where x.other {
   x.code.other;
   x.codeSystem.nonNull;
};
```

B.2.11.3 Свойство codeSystemName: ST (унаследовано от типа данных CD)

Определение: общее имя системы кодирования.

Имя системы кодирования не используется для вычислений. Оно может быть указано для облегчения интерпретации человеком значений свойств code и codeSystem. Предполагается, хотя и не является обязательным, что в спецификациях реализуемой технологии будет предусматриваться указание значения свойства codeSystemName в качестве аннотации к идентификатору UID, рассчитанной на восприятие человеком.

Системы, соответствующие стандарту HL7, не должны функционально полагаться на значение свойства codeSystemName. Это значение не должно модифицировать значение свойства codeSystem и не может существовать без значения свойства codeSystem.

```
invariant(CE x) {
   x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.11.4 Свойство codeSystemVersion: ST (унаследовано от типа данных CD)

Определение: дескриптор версии, определенный специально для данной системы кодирования (если применим).

Для каждой внешней системы кодирования в стандарте HL7 должно быть указано, как формируется строка со значением версии. Если для конкретной системы кодирования такое указание в стандарте отсутствует, то для такой системы обозначение версии не имеет определенного значения.

Различные версии одной и той же системы кодирования должны быть совместимыми. Если система кодирования изменена несовместимым образом, то она представляет собой другую систему кодирования, а не другую версию, как бы издатель это ни называл.

Например, издатель классификаций МКБ-9 и МКБ-10 назвал эти системы кодирования «9-м пересмотром» и «10-м пересмотром». Однако МКБ-10 представляет собой полное изменение кодов МКБ и не является обратно

совместимой. Поэтому в целях настоящей спецификации МКБ-9 и МКБ-10 рассматриваются как разные системы кодирования, а не как разные версии одной системы кодирования. Напротив, когда версия «1.0j» системы кодирования LOINC была обновлена до версии «1.0k», комитет HL7 рассматривал это как смену версий, поскольку новая версия была обратно совместимой с предыдущей.

```
invariant(CE x) {
   x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.11.5 Свойство displayName: ST (унаследовано от типа данных CD)

Определение: имя или название кода, под которым система-отправитель показывает значение кода своим пользователям.

Свойство displayName включено как для удобства интерпретации человеком значения кода, так и для документирования имени, используемого для изображения понятия пользователю. Оно не имеет функционального значения, не может существовать без кода и никогда не должно модифицировать смысл кода.

Примечания

1 В своих словарных доменах стандарты HL7 предусматривают атрибут «печатаемого имени» (print name). Значения этого атрибута могут использоваться в качестве значений свойства displayName.

2 Имена, предусмотренные для кодов, не могут менять смысл кодированного значения. Поэтому они не должны предоставляться пользователю прикладной системы-получателя, пока не будет уверенности в том, что такое имя адекватно отражает понятие, соответствующее кодированному значению. При коммуникациях нельзя просто полагаться на имя кодированного значения. Основная цель этого имени в обеспечении возможности отладки единиц данных в протоколах HL7 (например, сообщений).

```
invariant(CE x) {
   x.displayName.nonNull.implies(x.code.nonNull);
};
```

B.2.11.6 Свойство originalText: ED (унаследовано от типа данных CD)

Определение: текст или фраза, используемые в качестве основы для кодирования.

Исходный текст появляется в тех случаях, когда код был присвоен информации не ее источником, а специальным кодировщиком уже после ее появления (кодирование постфактум). Тогда при создании дескриптора понятия исходный текст может существовать без кода.

Примечание — Хотя кодирование постфактум часто осуществляется по информации, представленной как свободный текст, например в форме документов, сканированных изображений или диктовки, мультимедийные данные явным образом не разрешены в качестве исходного текста. Кроме того, свойство исходного текста originalText не может означать ссылку на весь исходный документ. Связи между различными артефактами медицинской информации (например, документом и кодированным результатом) не входят в область применения настоящей спецификации и рассматриваются в других стандартах HL7. Исходный текст должен представлять собой извлечение из оригинального источника, а не точную копию его содержания или указатель на этот источник. Поэтому исходный текст должен представляться в неформатированном виде.

Значения типа CD могут иметь непустое свойство исходного текста, даже если свойство code имеет пустое значение. В этом случае свойство originalText является именем или описанием понятия, которое не было закодировано. Такие исключительные значения типа CD могут также содержать преобразования translation. Такие преобразования обеспечивают непосредственное кодирование понятия, описанного в свойстве originalText.

Тип данных CE может быть понижен до типа данных ST. В этом случае значение типа ST будет представлено только свойством originalText значения типа CE.

```
invariant(CE x)
      where x.originalText.nonNull {
   ((ST)x).equal(x.originalText);
};
```

B.2.11.7 Свойство translation: SET<CD> (унаследовано от типа данных CD)

Определение: множество других дескрипторов понятия, которые преобразуют данный дескриптор понятия в другие системы кодирования.

Свойство translation представляет собой множество других значений типа CD, каждое из которых преобразует первое значение типа CD в другую систему кодирования. Каждый элемент множества преобразований был получен с помощью преобразования первого значения типа CD. Каждое преобразование, в свою очередь, может также содержать преобразования. Таким образом, если код многократно преобразуется, то информация о том, какой код послужил входным параметром каждого преобразования, сохранится.

Примечание — Преобразования являются квазисинонимами одного понятия реального мира. Предполагается, что каждое преобразование в этом множестве выражает то же самое понятие «другими словами». Однако между двумя структурно различными системами кодирования редко существует точная синонимия. Поэтому не все преобразования будут абсолютно точными.

B.2.11.8 Свойство qualifier: LIST<CR> (фиксированное)

```
invariant(CE x) {
   x.qualifier.notApplicable;
};
```

**B.2.12 Тип данных CharacterStringWithCode (SC) (специализация типа данных ST)**

Определение: строка символов, с которой может быть связан необязательный код. Если код присутствует, обязательно должен присутствовать и текст. В качестве кода нередко используется местный код.

Таблица B.16 — Сводка свойств типа данных CharacterStringWithCode

| Имя | Тип | Описание |
|---|---|---|
| code | CE | Код, представляющий строку символов. Например, такая строка может быть сообщением пользователю, выбираемым из каталога сообщений, а код представляет собой идентификатор сообщения в этом каталоге |

```
type CharacterStringWithCode alias SC specializes ST {
   CE code;
};
```

Тип данных SC используется в случаях, когда кодирование является скорее исключением, чем правилом (например, сообщения пользователю являются в основном текстовыми, и пользователю важно знать печатаемое сообщение). Однако иногда сообщения берутся из каталога заранее заготовленных текстов, ссылки на которые содержатся в значении типа SC.

Любое непустое значение типа SC МОЖЕТ иметь код, но код НЕ ДОЛЖЕН быть указан без текста.

```
invariant(SC x)
      where x.nonNull {
   x.code.nonNull.implies(x.notEmpty);
};
```

B.2.12.1 Свойство code: CE

Определение: код, представляющий строку символов. Например, такая строка может быть сообщением пользователю, выбираемым из каталога сообщений, а код представляет собой идентификатор сообщения в этом каталоге.

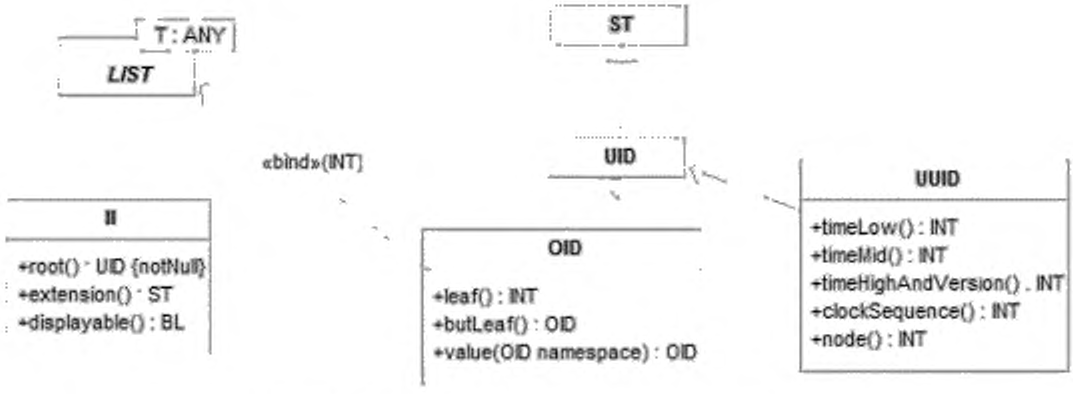


Рисунок B.5 — Типы данных экземпляров идентификаторов

**B.2.13 Тип данных UniqueIdentifierString (UID) (специализация типа данных ST)**

Тип данных UniqueIdentifierString описывает строку, являющуюся идентификатором объекта, обладающим свойством глобальной уникальности и не меняющимся с течением времени. Допустимые форматы, значения и процедуры этого типа данных строго контролируются комитетом HL7. В настоящее время пользовательскими идентификаторами могут быть определенные символьные представления объектных идентификаторов ИСО (тип данных OID) и универсально уникальные идентификаторы Распределенного компьютерного окружения DCE (тип данных UUID). Комитет HL7 также оставляет за собой право присваивать другие формы универсально уникальных идентификаторов (резервные схемы идентификаторов RUID), например, мнемонические идентификаторы систем кодирования.

Основным назначением уникальных идентификаторов (УИД) является идентификация объектов, обладающая свойством глобальной уникальности и не меняющаяся с течением времени. Формы УИД, будь то значения типов данных OID, UUID или RUID, совершенно не существенны. Пока они используются в стандартах HL7, единственное, что можно делать с УИД, это обозначать им объект, которому он присвоен. Сравнение УИД осуществляется по их литеральным формам, то есть если два УИД имеют идентичные литералы, то эти УИД считаются обозначениями одного и того же объекта. Если литералы двух УИД не идентичны, то они могут не обозначать один и тот же объект.

```
type UniqueIdentifierString alias UID specializes ST { };
```

Между различными допустимыми формами УИД нет никаких семантических различий. Разные формы не отличаются по компоненту, входящему в строку идентификатора или смежному с ней.

Хотя в настоящей спецификации и не проводится семантическое различие между разными формами уникальных идентификаторов, существуют определенные отличия в том, как эти идентификаторы образуются и как ими управляют, что и служит единственной причиной выделения подтипов УИД для каждого варианта.

**B.2.14 Тип данных объектного идентификатора ИСО ObjectIdentifier (OID) (специализация типа данных UID)**

Определение: глобально уникальная строка, представляющая объектный идентификатор ИСО (ОИД) в форме, образованной только из цифр и точек (например, «2.16.840.1.113883.3.1»). Согласно стандарту ИСО, идентификаторы ОИД являются путями в структуре дерева, где крайнее левое число означает корень дерева, а крайнее правое число — его лист.

Каждая ветвь под корнем соответствует организации, уполномоченной присваивать идентификаторы. В свою очередь, каждая из этих организаций может назначить свою совокупность уполномоченных организаций, действующих под их надзором, и так далее. В конечном счете одна из этих организаций присваивает уникальный (для себя) номер, соответствующий корневому узлу дерева. Этот лист может представлять уполномоченную организацию (в данном случае корневой ОИД, идентифицирующий организацию) или экземпляр объекта. Каждая уполномоченная организация владеет пространством имен, образованным ее поддеревом.

ОИД является предпочтительной схемой уникальных идентификаторов. Идентификаторы ОИД должны использоваться во всех случаях, за исключением ситуаций, когда выполняются критерии применимости других схем.

В разделе 28 ИСО/МЭК 8824:1990(E) объектный идентификатор определен следующим образом:

«28.9 Семантика значения объектного идентификатора определяется ссылкой на дерево объектных идентификаторов. Корень этого дерева соответствует стандарту [ИСО/МЭК 8824], а вершины (узлы) — уполномоченным по регистрации, отвечающим за выделение дуг [то есть ветвей], исходящих из этой вершины. Каждая дуга этого дерева помечается компонентом объектного идентификатора, являющегося [целым значением]. Каждому идентифицируемому объекту выделяется строго одна вершина (обычно являющаяся листом), которая не выделяется другим объектам (того же или другого типа). Таким образом, объект однозначно и недвусмысленно идентифицируется последовательностью целых значений (значений компонентов объектного идентификатора), помечающих дуги на пути от корня до выделенной объекту вершины.

28.10 Значение объектного идентификатора является семантически упорядоченным списком значений его компонентов. Начиная с корня дерева объектных идентификаторов, каждое значение компонента объектного идентификатора объекта отождествляет дугу в этом дереве. Последнее значение компонента объектного идентификатора отождествляет дугу, ведущую к вершине, которая была назначена объекту. Это и есть объект, который идентифицируется значением объектного идентификатора. [...]»

```
type ObjectIdentifier alias OID specializes UID, LIST<INT> {
          INT  leaf;
          OID  butLeaf;
          OID  value(namespace OID);
  literal ST;
};
```

В соответствии со стандартом ИСО/МЭК 8824 объектный идентификатор является упорядоченным списком значений его компонентов, представленных целыми числами. Эти значения компонентов упорядочены таким

образом, что корень дерева объектных идентификаторов является головой списка, а за ним следуют все дуги вплоть до листа, представляющего собой информационный объект, идентифицируемый полученным ОИД. Тот факт, что тип данных объектного идентификатора OID является специализацией типа данных LIST<INT>, отражает представление пути от корня до листа, образованного значениями компонентов объектного идентификатора.

Свойства «leaf» и «butLeaf» представляют собой взгляд на объектный идентификатор с другого конца. Свойство «leaf» является последним из значений компонентов объектного идентификатора в списке, а свойство «butLeaf» (кроме листа) представляет собой весь ОИД, за исключением листа, указанного в свойстве leaf. В известном смысле значение свойства «leaf» является значением идентификатора, а значение свойства «butLeaf» — идентификатором пространства имен, в котором значение этого идентификатора является уникальным и осмысленным.

Однако то, какая часть ОИД рассматривается как идентификатор значения, а какая — как идентификатор пространства имен, может трактоваться по-разному. В общем случае любая левая часть последовательности компонентов ОИД может рассматриваться как идентификатор пространства имен, а оставшаяся правая часть — как уникальное и осмысленное значение идентификатора. Эта точка зрения представлена свойством value в качестве аргумента в пространстве имен ОИД[1)].

```
invariant(OID x)
      where x.nonNull {
   x.notEmpty;
   x.tail.isEmpty.implies(x.leaf.equal(x.tail));
   x.tail.notEmpty.implies(x.leaf.equal(x.tail.leaf);
   x.tail.isEmpty.implies(x.butLeaf.isNull);
   x.tail.notEmpty.implies(x.butLeaf.head.equal(x.head)
         .and(x.butLeaf.tail.equal(x.butLeaf(x.tail))));
   forall(OID v; OID n) where v.equal(x.value(n)) {
      n.isEmpty.implies(v.equal(x));
      n.notEmpty.implies(v.equal(x.value(n.tail)));
      };
};
```

B.2.14.1 Объектные идентификаторы, присваиваемые комитетом HL7

Комитет HL7 должен создать регистр ОИД и по запросу присваивать ОИД в своей ветви пользователям и поставщикам, сотрудничающим с комитетом. Кроме того, комитет HL7 должен присвоить ОИД публичным организациям, присваивающим идентификаторы на национальном уровне в США (например, U.S. State driver license bureaus, U.S. Social Security Administration, HIPAA Provider ID registry и т. д.) и на международном уровне (например, органам социального страхования или регистрам граждан других стран). Для этих организаций должны использоваться ОИД, присвоенные комитетом HL7, даже если этим организациям присвоены ОИД из других источников.

Присваивая ОИД третьим сторонам или сущностям, комитет HL7 должен определить, не присвоены ли этим сущностям ОИД из других источников. В случае наличия таковых комитет HL7 должен занести соответствующий ОИД в свой каталог, но не присваивать собственный ОИД в ветви, контролируемой комитетом. По возможности комитет HL7 должен информировать третью сторону о том, что ей присвоен ОИД в ветви комитета HL7.

Хотя комитет HL7 должен провести тщательный поиск ОИД, уже присвоенных третьим сторонам, в отсутствие глобального механизма регистрации ОИД нельзя быть абсолютно уверенным, что третьей стороне еще не присвоен ОИД. Кроме того, повторное присвоение может произойти из других источников. В случае, когда комитету HL7 становится известно о наличии другого ОИД у третьей стороны, он должен предпринять определенные усилия для разрешения этой ситуации. Тем не менее для обеспечения интероперабельности тот ОИД, что присвоен комитетом HL7, должен быть предпочтительным.

Хотя большинство владельцев ОИД «конструируют» поддерево своего пространства имен, используя определенные принципы, общего способа придания смысла отдельным компонентам ОИД не существует. Комитет HL7 не стандартизует структуру поддеревьев пространства имен и не требует этого от других организаций. Тем не менее владелец ОИД или иное лицо, владеющее информацией о логической структуре компонентов ОИД, может использовать это знание для извлечения сведений об ассоциированном объекте. Однако такой подход не может быть обобщен.

Пример дерева ОИД показан на рисунке В.6. Комитету HL7 присвоен ОИД 2.16.840.1.113883.

Интерфейс, предложенный в стандарте HL7, не должен полагаться на какое-либо знание о структуре поддерева ОИД, присваивание которого комитет HL7 не может контролировать.

---

[1)] Представление объектного идентификатора ISO в виде пары значение/пространство имен отражает важную семантическую связь, а именно отношение между идентификатором и организацией, присваивающей идентификаторы (= пространство имен), являющееся очень важным в информационных системах здравоохранения вообще и в стандарте HL7 v2.x в частности.

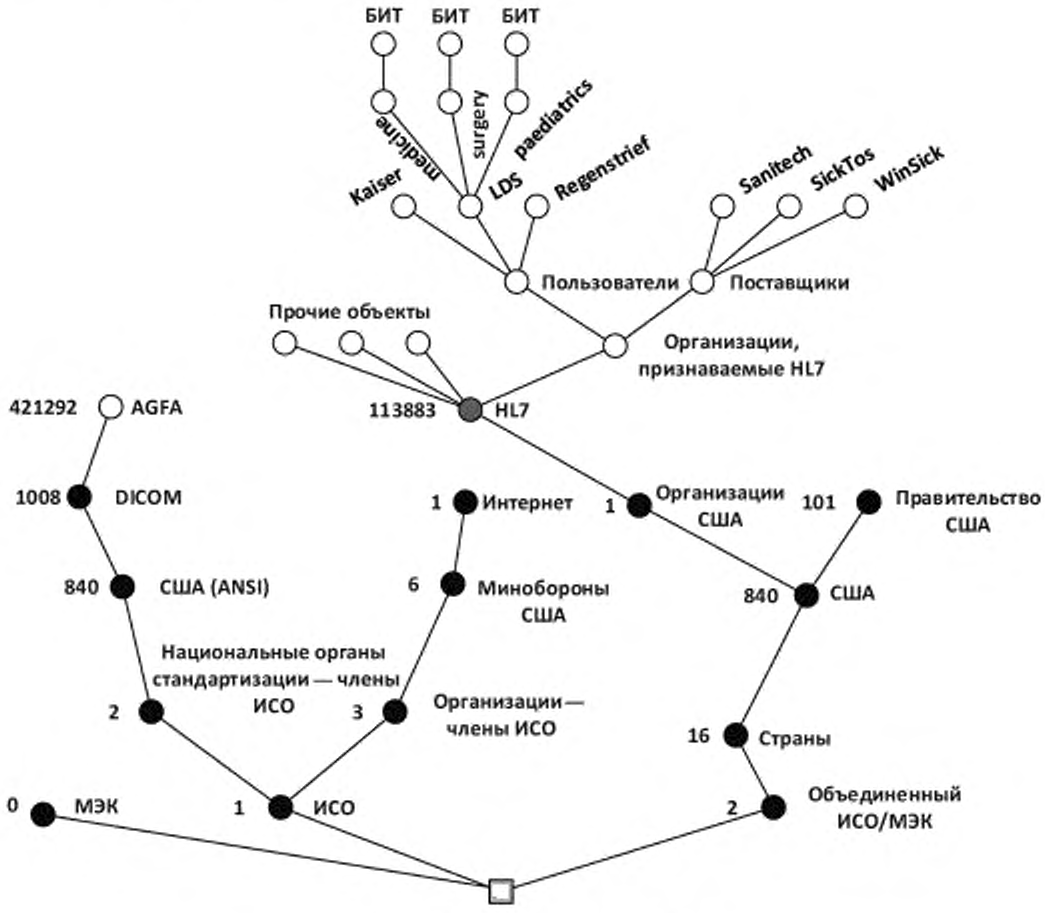


Рисунок В.6 — Пример дерева ОИД

В.2.14.2 Литеральная форма

Структурное определение типа данных ОИД представлено только в целях соответствия спецификации объектных идентификаторов. В стандартах HL7 идентификаторы ОИД используются только как строки УИД, то есть строка литерала является единственным передаваемым объектом, и получатель только ее и должен рассматривать при работе с идентификаторами ОИД в рамках спецификации HL7.

```
OID.literal ST {
   OID : INT "." OID { $.head.equal($1);
                       $.tail.equal($3); }
       | INT         { $.head.equal($1);
                       $.tail.isEmpty; }
};
```

Для совместимости со стандартом DICOM длина литеральной формы ОИД не должна превышать 64 символов (см. раздел 9 части 5 стандарта DICOM).

**В.2.15 Тип данных UniversalUniqueIdentifier (UUID) (специализация типа данных UID)**

Определение: глобально уникальная строка, представляющая Универсальный уникальный идентификатор (UUID) Распределенной вычислительной среды (DCE) в общем формате, состоящем из пяти групп шестнадцатеричных цифр, имеющих соответственно 8, 4, 4, 4 и 12 позиций и разделенных дефисами.

Как идентификатор UUID, так и его строковое представление определены в документе CDE 1.1 Remote Procedure Call specification, Appendix A организации Open Group.

Идентификаторы UUID присваиваются на основе MAC-адресов Ethernet, момента времени создания и некоторого случайного компонента. Предполагается, что это сочетание позволяет генерировать достаточно уникальные идентификаторы без какой-либо организационной политики присваивания идентификаторов (если не считать организацию присвоения MAC-адресов).

Идентификаторы UUID не являются предпочтительной схемой для использования в качестве УИД в стандартах HL7. Они могут использоваться, когда идентификаторы присваиваются отдельным объектам (например, идентификаторы экземпляров сущностей, идентификаторы событий действия и т. д.). Для объектов, описывающих классы сущностей или событий (например, каталога предметов), предпочтительной схемой идентификации являются ОИД.

```
type UniversalUniqueIdentifier alias UUID specializes UID {
   INT  timeLow;
   INT  timeMid;
   INT  timeHighAndVersion;
   INT  clockSequence;
   INT  node;
};
```

B.2.15.1 Литеральная форма

Структурное определение идентификаторов UUID представлено только в целях соответствия спецификации универсально уникальных идентификаторов. В стандартах HL7 идентификаторы UUID используются только как строки УИД, то есть строка литерала является единственным передаваемым объектом, и получатель только ее и должен рассматривать при работе с идентификаторами UUID в рамках спецификации HL7.

Литеральная форма идентификаторов UUID определена в соответствии с их исходной спецификацией. Следует иметь в виду, что в стандартах HL7 уникальные идентификаторы чувствительны к регистру, поэтому при использовании в соответствии со стандартами HL7 шестнадцатеричные цифры A-F в этих идентификаторах должны быть приведены к верхнему регистру.

```
UUID.literal ST {
   UUID        : hex8 "-" hex4 "-" hex4 "-" hex4 "-" hex12 {
                 $.timeLow.equal($1);
                 $.timeMid.equal($3);
                 $.timeHighAndVersion.equal($5);
                 $.clockSequence.equal($7);
                 $.node.equal($9);
                 }

   INT hex4    : hexDigit hexDigit hexDigit hexDigit {
                 $.equal($1.times(16).plus($2)
                  .times(16).plus($3)
                  .times(16).plus($4);
                 }

   INT hex8    : hexDigit hexDigit hexDigit hexDigit
                   hexDigit hexDigit hexDigit hexDigit {
                 $.equal($1.times(16).plus($2)
                  .times(16).plus($3)
                  .times(16).plus($4)
                  .times(16).plus($5)
                  .times(16).plus($6)
                  .times(16).plus($7)
                  .times(16).plus($8);
                 }

   INT hex12   : hexDigit hexDigit hexDigit hexDigit
                   hexDigit hexDigit hexDigit hexDigit
                   hexDigit hexDigit hexDigit hexDigit {
                 $.equal($1.times(16).plus($2)
                  .times(16).plus($3)
                  .times(16).plus($4)
                  .times(16).plus($5)
                  .times(16).plus($6)
                  .times(16).plus($7)
                  .times(16).plus($8)
```

```
                    .times(16).plus($9)
                    .times(16).plus($10)
                    .times(16).plus($11)
                    .times(16).plus($12);
                }

    INT hexDigit : "0"      { $.equal(0); }
                 | "1"      { $.equal(1); }
                 | "2"      { $.equal(2); }
                 | "3"      { $.equal(3); }
                 | "4"      { $.equal(4); }
                 | "5"      { $.equal(5); }
                 | "6"      { $.equal(6); }
                 | "7"      { $.equal(7); }
                 | "8"      { $.equal(8); }
                 | "9"      { $.equal(9); }
                 | "A"      { $.equal(10); }
                 | "B"      { $.equal(11); }
                 | "C"      { $.equal(12); }
                 | "D"      { $.equal(13); }
                 | "E"      { $.equal(14); }
                 | "F"      { $.equal(15); }
;
```

Примечание — Выходные данные программ и функций, использующих идентификаторы UUID, могут содержать всевозможные варианты форм их представления — в верхнем регистре, в нижнем регистре, с дефисами и без дефисов между группами цифр. Для соответствия спецификации HL7 эти выходные данные должны быть приведены к стандартному виду, то есть между группами цифр 8-4-4-4-12 должны быть указаны дефисы и все шестнадцатеричные цифры должны быть в верхнем регистре.

**B.2.16 Тип данных HL7 ReservedIdentifierScheme (RUID) (специализация типа данных UID)**

Определение: глобально уникальная строка, определенная исключительно в интересах комитета HL7. Идентификаторы в этой схеме утверждаются путем голосования спецификаций HL7. Местные сообщества или системы никогда не должны использовать такие зарезервированные идентификаторы на основе двусторонних соглашений.

Зарезервированные идентификаторы HL7 представляют собой строки, состоящие только из букв набора символов US-ASCII, цифр и дефисов, причем первый символ идентификатора должен быть буквой. Комитет HL7 может использовать эти зарезервированные идентификаторы как мнемонику основных понятий в сфере интересов комитета.

**B.2.17 Тип данных InstanceIdentifier (II) (специализация типа данных ANY)**

Идентификатор экземпляра, являющийся уникальным идентификатором предмета или объекта. Примерами служат идентификаторы объектов в модели HL7 RIM, номера медицинских карт, идентификаторы заказов, идентификаторы услуг в прейскуранте, идентификационные номера автомобилей VIN (Vehicle Identification Number) и т. д. Определения идентификаторов экземпляров основаны на объектных идентификаторах ИСО.

Таблица B.17 — Сводка свойств типа данных InstanceIdentifier

| Имя | Тип | Описание |
|---|---|---|
| root | UID | Уникальный идентификатор, гарантирующий глобальную уникальность идентификатора экземпляра. Значение свойства root может само по себе быть идентификатором экземпляра |
| extension | ST | Строка символов, являющаяся уникальным идентификатором в пространстве имен, заданном значением свойства root |
| assigningAuthorityName | ST | Человекочитаемое название или мнемоника уполномоченной организации по присвоению идентификаторов. Не имеет вычислительного значения. Это свойство предназначено для помощи человеку, интерпретирующему значение типа II.<br>Примечание: никакая автоматизированная обработка не должна зависеть от названия уполномоченной организации, в какой бы форме оно ни было представлено |
| displayable | BL | Указывает, предназначен ли идентификатор для вывода на экран и ввода данных (displayable = true) или же используется только для машинной обработки (displayable = false) |

```
type InstanceIdentifier alias II specializes ANY {
   ST   extension;
   UID  root;
   ST   assigningAuthorityName;
   BL   equal(ANY x);
};
```

B.2.17.1 Свойство root: UID

Определение: уникальный идентификатор, гарантирующий глобальную уникальность идентификатора экземпляра. Значение свойства root может само по себе быть идентификатором экземпляра.

При наличии непустого свойства extension значение свойства root обычно трактуется как «уполномоченная организации по присвоению идентификаторов», то есть предполагается, что это значение каким-то образом указывает организацию, присваивающие идентификаторы, передаваемые в свойстве extension. Однако значение свойства root не обязано представлять УИД организации, оно может также быть специально зарегистрированным уникальным идентификатором схемы идентификации[1)].

```
invariant(II x)
      where x.nonNull {
   root.nonNull;
};
```

B.2.17.2 Свойство extension: ST

Определение: Строка символов, являющаяся уникальным идентификатором в пространстве имен, заданном значением свойства root.

Значение свойства extension представляет собой строку символов, уникальную в пространстве имен, заданном значением свойства root. Если оно не является пустым, то свойство root задает пространство имен (иногда называемое «уполномоченной организацией по присвоению идентификаторов» или «типом идентификатора»). Свойство extension может быть пустым, и в таком случае значение свойства root само будет идентификатором экземпляра.

Схема идентификации, обеспечиваемая свойствами root и extension, фактически означает, что конкатенация значений свойств root и extension должна быть уникальным идентификатором объекта, идентифицируемым данным значением типа II.

Для внешних идентификаторов передаваемых объектов рекомендуется использовать схему OID. Свойство extension в основном предназначено для передачи унаследованных алфавитно-цифровых идентификаторов.

В некоторых схемах идентификации используется определенный стиль представления соответствующих им значений. Например, номер карточки социального страхования в США SSN (Social Security Number) обычно записывается с дефисами по шаблону «123-12-1234». Однако дефисы не являются значащими символами, и номер SSN может быть равным образом представлен без дефисов как «123121234».

В случае, если схемы идентификации предусматривают несколько разных представлений, комитетом HL7 должно быть принято решение о предпочтительной форме. Комитет должен документировать это решение, если им рекомендована соответствующая внешняя система идентификации. Решение о предпочтительной форме должно приниматься с учетом практичности и широты использования. Если четких критериев практичности и широты использования нет, то должно быть отдано предпочтение наиболее безопасной, расширяемой и наименее стилизованной (наименее декорированной) форме[2)].

---

[1)] Объекты DICOM идентифицируются только с помощью УИД. С точки зрения интеграции стандартов DICOM и HL7 было бы неразумно со стороны комитета HL7 требовать, чтобы свойство extension было обязательным и рассматривать УИД только как идентификатор организации, присваивающей идентификаторы. Поскольку идентификаторы УИД проще и не подвержены рискам незначащего декорирования, то комитет HL7 поддерживает практику использования в системах простых идентификаторов УИД в качестве ссылок на их объекты.

[2)] Это решение, принимаемое на этапе конструирования, необходимо, чтобы интерфейсы, предлагаемые в стандартах HL7, не были отягощены необходимостью реализации преобразований различных стилей литералов кодов во время исполнения, хотя это и может привести к ситуации, когда некоторым приложениям может потребоваться выполнять преобразования из одной формы представления кодов в другую, если они рассчитаны на вариант представления, не выбранный комитетом HL7.

На основании практического опыта рекомендуется, чтобы алфавитно-цифровые значения свойства extension типа данных II не содержали ведущих нулей (если в них присутствуют нули), поскольку эти нули нередко ошибочно вырезаются. Значения «000123» и «123» свойства extension должны рассматриваться как совершенно разные, но это правило может игнорироваться, что приводит к ошибочным совпадениям и дубликатам записей в базах данных. Но приложения должны учитывать все ведущие нули, обнаруженные в значениях свойства extension. Ведущие нули запрещены в идентификаторах ОИД, но в идентификаторах UUID они могут случаться, и это надо учитывать.

Отдельного свойства контрольной цифры нет. Контрольные цифры используются при ручном вводе данных и лучше всего работают, если они полностью прозрачны. Значения свойства extension типа данных II может со-

Комитет HL7 может также принять решение об отображении распространенных внешних идентификаторов в значения свойства II.root типа OID. Например, указанный выше номер SSN может быть представлен как 2.16.840.1.113883.4.1.123121234. В каждом конкретном случае решение комитета HL7 будет приниматься с учетом практичности и широты использования.

B.2.17.3 Свойство assigningAuthorityName: ST

Определение: человекочитаемое название или мнемоника уполномоченной организации по присвоению идентификаторов. Не имеет вычислительного значения. Это свойство предназначено для помощи человеку, интерпретирующему значение типа II.

Примечание — никакая автоматизированная обработка не должна зависеть от названия уполномоченной организации, в какой бы форме оно ни было представлено.

B.2.17.4 Свойство displayable: BL

Определение: указывает, предназначен ли идентификатор для вывода на экран и ввода данных (displayable = true) или же используется только для машинной обработки (displayable = false).

B.2.17.5 Свойство equality: BL (унаследовано от типа данных ANY)

Два идентификатора экземпляра равны в том и только том случае, если равны значения их свойств root и extension.

```
invariant(II x, y)
      where x.nonNull.and(y.nonNull) {
   x.equal(y).equal(x.root.equal(y.root)
            .and(x.extension.equal(y.extension))};
};
```

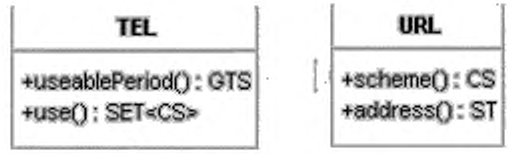


Рисунок B.7 — Типы данных URL и TEL

**B.2.18 Тип данных UniversalResourceLocator (URL) (специализация типа данных ANY)**

Определение: телекоммуникационный адрес, соответствующий стандарту Интернет RFC 2396 [http://www.ietf.org/rfc/rfc2396.txt]. Этот стандарт описывает идентификатор URI (Uniform Resource Identifier — унифицированный указатель ресурсов), задающий протокол и точку контакта, определяемую этим протоколом для ресурса. Важными примерами использования типа данных URL являются номера телефонов и факсов, адреса электронной почты, гипертекстовые ссылки, ссылки FTP и т. д.

Стандарт Интернет RFC 2396 [http://www.ietf.org/rfc/rfc2396.txt] определяет адрес URI следующим образом:

Поскольку существует много различных методов доступа к ресурсам, существует и несколько схем описания местонахождения таких ресурсов. Общий синтаксис адресов URL служит базой для определения новых схем, основанных на использовании протоколов, отличающихся от тех, что описаны в этом документе.

Адреса URL используется для указания «местонахождения» ресурсов, предоставляя абстрактную идентификацию местонахождения ресурса. Получив информацию о местонахождении ресурса, система может выполнить различные операции над этим ресурсом, которые могут быть охарактеризованы такими словами, как «доступ», «изменение», «замена», «поиск атрибутов». В общем случае для любой схемы URL достаточно указать только метод «доступ».

По местным соглашениям вместо адресов URL можно использовать идентификаторы URI. В этих случаях все же ожидается, что к идентифицированному ресурсу обеспечивается доступ некоторым согласованным методом. Идентификаторы URI широко используются для ссылки на вложения в конверты SOAP.

```
protected type UniversalResourceLocator alias URL specializes ANY {
            CS  scheme;
            ST  address;
   literal ST;
};
```

---

держать контрольные цифры в любом месте, а конкретная схема контрольного суммирования (если таковая есть) должна однозначно вытекать из значения свойства II.root. Но отдельное свойство контрольной цифры намеренно не включено в настоящую спецификацию.

B.2.18.1 Свойство scheme: CS

Определение: идентифицирует схему URL — протокол, используемый для интерпретации адресной строки и для доступа к ресурсу, адресованному таким способом.

Некоторые схемы URL регистрируются организацией Internet Assigned Numbers Authority (IANA) [http://www.iana.org], однако эта организация регистрирует только те схемы URL, которые описаны в документах Интернет RFC. В действительности есть целый ряд схем URL, определенных не в документах RFC. Часть из них зарегистрирована консорциумом World Wide Web Consortium (W3C)[1].

Аналогично типу среды ED.mediaType в стандарте HL7 даются некоторые рекомендации для значений свойства scheme, классифицируя их как обязательные, рекомендованные, прочие и запрещенные. Любая не упомянутая в стандарте схема имеет статус «прочая».

Таблица B.18 — Домен значений свойства scheme

| Код | Имя | Определение |
|---|---|---|
| fax | Fax (факс) | Телефонный номер факсимильного устройства [http://www.ietf.org/rfc/rfc3966.txt и http://www.ietf.org/rfc/rfc2806.txt] |
| file | File (файл) | Имена файлов, специфичные для места хранения [RCF 1738]. Следует обратить внимание, что схема file имеет смысл только для местных файлов. Передача таких имен другим системам мало востребована, поскольку система-получатель вряд ли будет способна получить непосредственный доступ к этому файлу |
| ftp | FTP | Протокол передачи файлов FTP (File Transfer Protocol) [http://www.ietf.org/rfc/rfc1738.txt] |
| http | HTTP | Протокол передачи гипертекста HTTP (Hypertext Transfer Protocol) [http://www.ietf.org/rfc/rfc2368.txt] |
| mailto | Mailto | Адрес электронной почты [http://www.ietf.org/rfc/rfc2368.txt] |
| mllp | MLLP | Традиционный протокол MLLP (Minimal Lower Layer Protocol — протокол минимально возможного уровня), используемый для передачи сообщений по стандартам HL7. Адрес URL для такого протокола имеет ту же форму, что и IP-адрес, например, «mllp://<host>:<port>/», где <host> является IP-адресом или именем узла DNS, а <port> является номером порта, обслуживающего протокол MLLP |
| modem | Modem (модем) | Телефонный номер модема [http://www.ietf.org/rfc/rfc3966.txt и http://www.ietf.org/rfc/rfc2806.txt] |
| nfs | NFS | Протокол сетевой файловой системы NFS (Network File System) [http://www.ietf.org/rfc/rfc2224.txt]. Серверы NFS используются в некоторых местах для общего доступа к файлам |
| tel | Telephone (телефон) | Телефонный номер для передачи голоса [http://www.ietf.org/rfc/rfc3966.txt и http://www.ietf.org/rfc/rfc2806.txt]. |
| telnet | Telnet | Ссылка на диалоговые сеансы [http://www.ietf.org/rfc/rfc1738.txt]. В некоторых местах (например, в лабораториях) используются компьютерные терминалы для удаленных сеансов запросов по протоколу Telnet |

Следует учесть, что настоящий стандарт явно ограничивает этот тип данных адресами URL. Универсальные имена ресурсов URN (Universal Resource Name) в ней не разрешены. Имена URN представляют собой вид схемы идентификации, используемой для ресурсов, к которым не обеспечивается доступ. А настоящий стандарт распространяется только на ресурсы, к которым может быть получен доступ и которые могут быть адресованы с помощью URL.

B.2.18.2 Свойство address: ST

Определение: адрес представляет собой строку символов, формат которой полностью определяется значением свойства scheme.

B.2.18.3 Литеральная форма

[1] Свойство scheme имеет тип данных CS и для целей стандартов HL7 имеет нерасширяемый домен (с квалификатором CNE). С учетом того факта, что ни одного официального списка схем URL не существует и нередко используются местные схемы, это может показаться странным. Однако комитет HL7 не может позволить расширение схемы URL с использованием механизма местных альтернативных систем кодирования, поэтому технически свойству scheme присвоен тип данных CS.

В то время как концептуально значения типа URL имеют свойства scheme и address, внешний вид адреса URL представляет собой строковый литерал, сформированный в соответствии со стандартом Internet. Общий синтаксис литерала URL определен следующим образом:

```
URL.literal ST {
   URL : /[a-z0-9+.-]+/ ":" ST    { $.scheme.equal($1);
                                    $.address.equal($3); }
};
```

**Номера факсов и телефонов**

Обратите внимание, что для телефонных номеров нет специального типа данных. Они имеют тип данных TEL и специфицированы как адреса URL.

Адреса URL телефонных номеров определены в документе Интернет RFC 2806 [http://www.ietf.org/rfc/rfc2806.txt]. Это определение кратко излагается в настоящем подразделе. Оно не предназначено для переопределения или изменения каких-либо правил, описанных в указанном выше документе.

Адреса URL голосовых телефонов начинаются с приставки «tel:», а факсимильных телефонов — с приставки «fax:».

Свойство address содержит телефонный номер, соответствующий спецификации ITU-T E.123 Telephone Network and ISDN Operation, Numbering, Routing and Mobile Service: Notation for National and International Telephone Numbers (1993). Пока комитет HL7 не дополняет и не удаляет спецификацию адреса URL, предпочтительное подмножество синтаксиса телефонного номера, передаваемого в свойстве address, имеет следующее описание:

```
protected type TelephoneURL specializes URL {
   literal ST {
      URL                 : /(tel)|(fax)/ ":" address    { $.scheme.equal($1);
                                                           $.address.equal($3); };
      ST address          : "+" phoneDigits
      ST phoneDigits      : digitOrSeparator phoneDigits
                          | digitOrSeparator
      ST digitOrSeparator : digit
                          | separator;
      ST digit            : /[0..9]/;
      ST separator        : /[().-]/;
      };
};
```

Предпочтительными являются абсолютно глобальные телефонные номера, начинающиеся со знака «+» и кода страны. Разделяющие символы служат для удобства чтения, но не придают никакого особого смысла телефонному номеру. Например, строки «tel:+13176307960» и «tel:+1(317)630-7960» описывают один и тот же телефонный номер; строки «fax:+49308101724» и «fax:+49(30)8101-724» описывают один и тот же номер факса.

**B.2.19 Тип данных TelecommunicationAddress (TEL) (специализация типа данных URL)**

Определение: телефонный номер (голосового телефона или факса), адрес электронной почты или иной указатель ресурса, воспринимаемый телекоммуникационным оборудованием. Этот телекоммуникационный адрес указывается как указатель URL, дополненный спецификациями времени и кодами использования, помогающими определить, какой адрес следует использовать в данное время и для данной цели.

Таблица B.19 — Сводка свойств типа данных TelecommunicationAddress

| Имя | Тип | Описание |
|---|---|---|
| useablePeriod | GTS | Указывает периоды времени, в течение которых можно пользоваться данным телекоммуникационным адресом. Для телефонного номера это могут быть периоды времени, в течение которого нужному абоненту можно звонить по этому номеру. Для веб-адреса это могут быть периоды времени, в течение которого содержание сайта предполагается доступным по этому адресу |
| use | SET<CS> | Один или несколько кодов, информирующих систему или пользователя о том, какой из телекоммуникационных адресов следует выбрать для данной цели контакта |

Семантика телекоммуникационного адреса состоит в том, что взаимодействующая по нему сущность (ответчик) ожидает запросы по этому адресу и готова на них ответить, следовательно, инициатор взаимодействия может установить с ним контакт.

Ответчиком по данному телекоммуникационному адресу может быть автоматическая служба, возвращающая необходимую информацию (например, службы FTP или HTTP). В этом случае телекоммуникационный адрес является ссылкой на информацию, доступную по этому адресу. Таким образом, по значению телекоммуникационного адреса можно получить некоторую информацию (в форме инкапсулированных данных типа ED).

```
type TelecommunicationAddress alias TEL specializes URL {
   GTS       useablePeriod;
   SET<CS>   use;
   BL        equal(ANY x);
};
```

Тип данных телекоммуникационного адреса представляет собой расширение типа данных UniversalResourceLocator (URL), определенного в соответствии со стандартом Интернет RFC 2396 [http://www.ietf.org/rfc/rfc2396.txt]. Тип данных URL указывает протокол и точку контакта, определенную этим протоколом для доступа к ресурсу. Наиболее примечательными приложениями типа данных телекоммуникационного адреса являются номера телефонов и факсов, адреса электронной почты, гипертекстовые ссылки, FTP-адреса и т. д.

B.2.19.1 Свойство useablePeriod: GTS

Определение: указывает периоды времени, в течение которых можно пользоваться данным телекоммуникационным адресом. Для телефонного номера это могут быть периоды времени, в течение которого нужному абоненту можно звонить по этому номеру. Для веб-адреса это могут быть периоды времени, в течение которого содержание сайта предполагается доступным по этому адресу.

B.2.19.2 Свойство use: SET<CS>

Определение: один или несколько кодов, информирующих систему или пользователя о том, какой из телекоммуникационных адресов следует выбрать для данной цели контакта.

Таблица B.20 — Домен значений свойства use

| Код | Имя | Определение |
|---|---|---|
| H | home address (домашний адрес) | Домашний телекоммуникационный адрес; попытка контакта по этому адресу для деловых целей может явиться нарушением неприкосновенности личной жизни, и вместо нужного лица может ответить член семьи или иной житель. Обычно используется в экстренных случаях или если других способов контакта нет |
| HP | primary home (основной домашний адрес) | Основной домашний телекоммуникационный адрес. Используется для контакта с лицом во внерабочее время |
| HV | vacation home (домашний на время отпуска) | Домашний телекоммуникационный адрес на время отпуска. Используется для контакта с лицом во время его отпуска |
| WP | work place (служебный) | Служебный адрес. Первый, по которому должны начинаться попытки контакта в рабочее время |
| DIR | direct (прямой) | Адрес рабочего места или телекоммуникационный адрес, по которому можно связаться с лицом без посредников. У телефона такой номер часто называется «прямым» |
| PUB | public (публичный) | «Стандартный» адрес рабочего места или телекоммуникационный адрес, по которому можно связаться со справочной, общим почтовым ящиком или иным посредником, обеспечивающим контакт с нужным лицом |
| BAD | bad address (плохой адрес) | Признак «плохого», бесполезного адреса |
| TMP | temporary address (временный адрес) | Временный адрес, может быть приемлемым для визита или корреспонденции. История смены адресов может представить более детальную информацию |
| AS | answering service (автоответчик) | Автоответчик, используемый для менее срочных контактов, а также в ситуация, когда основной целью контакта является оставить сообщение или прослушать автоматическое объявление |

*Окончание таблицы 20*

| Код | Имя | Определение |
|---|---|---|
| EC | emergency contact (экстренный контакт) | Контакт, специально указанный для связи в экстренных случаях. Первый, по которому должны начинаться попытки контакта в экстренных случаях вне зависимости от наличия адресов с другими кодами использования |
| MC | mobile contact (мобильный контакт) | Мобильное телекоммуникационное устройство, которое владелец носит с собой. Может иметь характеристики других кодов использования, пригоден для срочных контактов, но не является первым, по которому должны начинаться попытки обычных деловых контактов |
| PG | pager (пейджер) | Пейджер, позволяющий инициировать обратный вызов или оставить очень короткое сообщение |

Коды использования телекоммуникаций не являются полной классификацией типов оборудования или местонахождений. Их основной целью является предложить использование конкретного телекоммуникационного адреса или предостеречь от его использования. Не так-то просто предложить точные правила, управляющие выбором телекоммуникационного адреса.

B.2.19.3 Свойство equal: BL (унаследовано от типа данных ANY)

Два значения телекоммуникационного адреса считаются равными, если равны их URL. Свойства use и useablePeriod из проверки на равенство исключаются.

```
invariant(TEL x, y)
      where x.nonNull.and(y.nonNull) {
   x.equal(y).equal(((URL)x).equal((URL)y));
};
```

Типы данных почтового адреса и имен сущностей (фамилия, имя, отчество лица, название организации, тривиальное имя) основаны на расширении строкового типа данных ST (рисунок B.8).

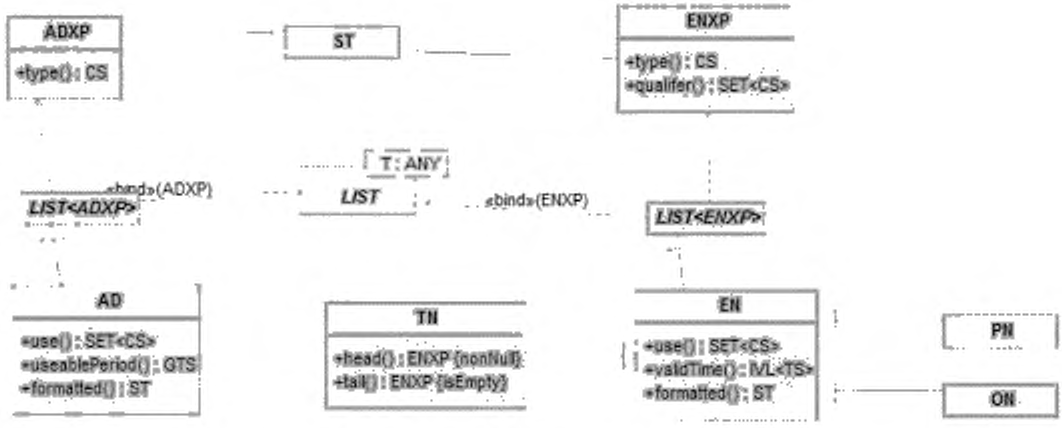


Рисунок B.8 — Типы данных почтового адреса и имен сущностей

**B.2.20 Тип данных AddressPart (ADXP) (специализация типа данных ST)**

Определение: строка символов, которая может иметь маркировку типа, указывающую ее роль в адресе. Типичными компонентами, присутствующими почти в каждом адресе, являются название улицы, номер дома или почтового ящика, почтовый индекс, город, страна. Другие роли могут быть определены на региональном, национальном или даже на ведомственном уровне (например, адреса воинских частей). Обычно адреса разбиваются на строки, обозначаемыми специальными признаками разрыва строки (например, DEL).

Таблица B.21 — Сводка свойств типа данных AddressPart

| Код | Имя | Определение |
|---|---|---|
| partType | CS | Указывает, означает ли компонент адреса улицу, город, страну, почтовый индекс, почтовый ящик и т. д. Если это свойство имеет пустое значение, то компонент адреса является не классифицированным и должен представлять собой неструктурированный адрес |

```
protected type AddressPart alias ADXP specializes ST {
   CS  type;
};
```

B.2.20.1 Свойство partType: CS

Определение: указывает, означает ли компонент адреса улицу, город, страну, почтовый индекс, почтовый ящик и т. д. Если это свойство имеет пустое значение, то компонент адреса является не классифицированным и должен представлять собой неструктурированный адрес.

Таблица B.22 — Домен значений свойства partType

| Код | Имя | Определение |
|---|---|---|
| ADL | additional locator (дополнительный указатель) | Компонент с таким свойством может обозначать некоторую единицу адреса, например, номер квартиры, комнаты или этажа. В одном адресе может быть несколько таких компонентов (например, «3-й этаж, кв. 342»). Это может быть обозначением места, находящегося в стороне от адреса, а не меньшей единицей внутри некоторой большей (например, в Нидерландах обозначение «t.o.» означает «напротив» и используется для указания жилых барж, стоящих напротив фасадов домов, выходящих на улицу) |
| UNID | unit identifier (идентификатор единицы) | Номер или название конкретной части здания или комплекса, присвоенные в этом здании или комплексе |
| UNIT | unit designator (обозначение единицы) | Указывает тип конкретной части здания или комплекса, например, квартира, этаж |
| DAL | delivery address line (строка адреса доставки) | Строка адреса доставки часто используется вместо выделения способа доставки, почтового накопителя и т. д. Обычно в адресе указывают либо строку с адресом улицы, либо строку с адресом доставки, но не обе вместе |
| DINST | delivery installation type (тип почтового накопителя) | Указывает тип почтового накопителя (места, в которое письмо должно быть доставлено перед окончательной передачей с помощью способа доставки). Примеры: почтовое отделение, почтовый узел, центральный телеграф, станция и т. д. |
| DINSTA | delivery installation area (территория почтового накопителя) | Местонахождение почтового накопителя, обычно поселок или город. Требуется только в тех случаях, когда оно отличается от муниципального образования. Территория, на которой осуществляется доставка почты этим накопителем, например, доставка почтальоном, доставка по сельскому или городскому маршруту |
| DINSTQ | delivery installation qualifier (квалификатор почтового накопителя) | Номер, буква или имя, выделяющие почтовый накопитель. Например, для «Станции А» квалификатором, выделяющим этот накопитель, будет буква «А» |
| DMOD | delivery mode (метод доставки) | Тип предлагаемой услуги, метод доставки. Например, почтовый ящик (в почтовом отделении), сельский маршрут, общая доставка и т. д. |
| DMODID | delivery mode identifier (идентификатор метода доставки) | Идентификатор метода доставки, например, номер маршрута доставки. Конкретизирует метод доставки (например, указывает номер почтового ящика или сельского маршрута) |
| SAL | street address line (строка адреса улицы) | |
| BNR | building number (номер здания) | Номер здания, дома или участка, находящегося на улице. Называется также «основным уличным номером». Это скорее номер здания, нежели улицы |

*Окончание таблицы В.22*

| Код | Имя | Определение |
|---|---|---|
| BNN | building number numeric (числовой компонент номера здания) | Числовой компонент номера здания |
| BNS | building number suffix (суффикс номера здания) | Любая буква, дробь или иной текст, который может быть указан после числового компонента номера здания |
| STR | street name (название улицы) | Название улицы |
| STB | street name base (базовое название улицы) | Базовое название проезда по улице, присвоенное муниципалитетом (исключая тип и направление улицы) |
| STTYP | street type (тип улицы) | Тип улицы (например, улица, проспект, площадь и т. д.) |
| DIR | direction (направление) | Направление улицы (например, «Северная», «Южная», «Западная», «Восточная») |
| CAR | care of (через) | Название получателя почтового отправления, который находится по этому адресу и обеспечит передачу отправления его адресату |
| CEN | census tract (переписной район) | Переписной район, выделяемый в демографических целях |
| CNT | country (страна) | Страна |
| CPA | county or parish (графство или приход) | Административная единица штата или провинции [в 49 штатах США используется термин «county» (графство), в Луизиане — «parish» (приход)] |
| CTY | municipality (муниципалитет) | Название города, поселка, деревни или иного муниципального образования |
| DEL | delimiter (разделитель) | Разделители печатаются без окаймления пробельными символами. Если этот компонент пуст, то разделителем служит переход на следующую строку |
| POB | post box (почтовый ящик) | Пронумерованный ящик в почтовом отделении |
| PRE | precinct (избирательный участок) | Структурная единица муниципального образования |
| STA | state or province (штат или провинция) | Административная единица страны, которая в федеральных государствах сохраняет ограниченный суверенитет |
| ZIP | postal code (почтовый индекс) | Почтовый индекс, обозначающий район, обслуживаемый почтовым отделением |

**B.2.21 Тип данных PostalAddress (AD) (специализация типа данных LIST<ADXP>)**

Определение: почтовый, домашний или служебный адрес. Представляет собой последовательность компонентов адреса, например, названия улицы или номера почтового ящика, города, почтового индекса, страны и т. д.

Тип данных AD прежде всего используется для передачи данных, позволяющих печатать этикетки с адресами, позволяющими лицу физически посетить адресата. Тип данных почтового адреса AD не предназначен служить контейнером для дополнительной информации, которая может быть полезной для поиска географического местонахождения (например, для GPS-координат) или для проведения эпидемиологических исследований. Такая дополнительная информация может передаваться в других, более подходящих элементах, определенных в стандартах HL7.

Таблица В.23 — Сводка свойств типа данных PostalAddress

| Имя | Тип | Описание |
|---|---|---|
| use | SET<CS> | Один или несколько кодов, информирующих систему или пользователя о том, какой из адресов следует выбрать для данной цели |

*Окончание таблицы В.23*

| Имя | Тип | Описание |
|---|---|---|
| useablePeriod | GTS | Общая спецификация времени GTS (General Timing Specification), указывающая периоды времени, в течение которых можно использовать данный адрес. Используется для указания разных адресов в зависимости от дня недели или от года |
| isNotOrdered | BL | Булевское значение, указывающее, известен ли порядок компонентов адреса или нет. Хотя компоненты адреса всегда передаются в определенном порядке, тем не менее порядок, в котором они должны быть представлены пользователю, может быть, а может и не быть известен. Если он не известен, то это может быть указано с помощью свойства isNotOrdered |
| formatted | ST | Строковое значение адреса, форматированное по строкам и с нужными пробельными символами. Это только семантическое свойство, определяющее назначение некоторых типов компонентов адреса[1)] |

Адреса концептуально рассматриваются как текст с добавленной логической разметкой. Такая разметка может разбивать адрес на строки и может детально описывать роль каждого компонента адреса, если она известна. Компоненты адреса присутствуют в нем в том порядке, как они печатаются на этикетке почтового отправления. Этот подход похож на разметку текста HTML или XML (но технически не сводится к представлению XML).

По существу, адреса являются последовательностями своих компонентов, но при этом имеют дополнительные свойства use и useablePeriod, указывающие, как и когда адрес может использоваться для конкретной цели.

```
type PostalAddress alias AD specializes LIST<ADXP> {
   SET<CS>  use;
   GTS      useablePeriod;
   BL       isNotOrdered;
   BL       equal(ANY x);
   ST       formatted;
};
```

B.2.21.1 Свойство use: SET<CS>

Определение: один или несколько кодов, информирующих систему или пользователя о том, какой из адресов следует выбрать для данной цели.

Таблица В.24 — Домен значений свойства use типа данных PostalAddress

| Код | Имя | Определение |
|---|---|---|
| H | home address (домашний адрес) | Домашний адрес. Попытка контакта по этому адресу для деловых целей может явиться нарушением неприкосновенности личной жизни, и вместо нужного лица по этому адресу может находиться член семьи или иной житель. Обычно используется в экстренных случаях или если других способов контакта нет |
| HP | primary home (основной домашний адрес) | Основной домашний адрес. Используется для контакта с лицом во внерабочее время |
| HV | vacation home (домашний на время отпуска) | Домашний адрес на время отпуска. Используется для контакта с лицом во время его отпуска |

[1)] Следует учесть, что семантические свойства свободны от любой семантики потока управления. Свойство formatted может быть реализовано как «процедура», «возвращающая» форматированный адрес, но обычно оно не является переменной, которой можно присвоить форматированный адрес. Однако стандарт HL7 определяет не приложения, а только семантику передаваемых значений данных. Поэтому семантическая модель абстрагируется от понятий наподобие «процедуры», «возвращения» и «присваивания», а оперирует только свойством и значением.

*Окончание таблицы В.24*

| Код | Имя | Определение |
|---|---|---|
| WP | work place (служебный) | Служебный адрес. Первый, по которому должны начинаться попытки контакта в рабочее время |
| DIR | direct (прямой) | Адрес рабочего места или телекоммуникационный адрес, по которому можно связаться с лицом без посредников. У телефона такой номер часто называется «прямым» |
| PUB | public (публичный) | «Стандартный» адрес рабочего места или телекоммуникационный адрес, по которому можно связаться со справочной, общим почтовым ящиком или иным посредником, обеспечивающим контакт с нужным лицом |
| BAD | bad address (плохой адрес) | Признак «плохого», бесполезного адреса |
| TMP | temporary address (временный адрес) | Временный адрес, может быть приемлемым для визита или корреспонденции. История смены адресов может представить более детальную информацию |
| Следующие коды указывают различные представления имен. Способ представления может влиять на использование имени (например, для формальных коммуникаций может требоваться идеографическое представление) | | |
| ABC | Alphabetic (алфавитное) | Алфавитное представление имени (romaji в Японии) |
| IDE | Ideographic (идеографическое) | Идеографическое представление имени (kanji в Японии, китайские иероглифы) |
| SYL | Syllabic (силлабическое) | Силлабическая транскрипция имени (kana в Японии, hangul в Корее) |
| PHYS | physical visit address (адрес физического визита) | Используемый в основном для посещения этого адреса |
| PST | postal address (почтовый адрес) | Используется для отправки письма |

Адрес, для которого свойство use не указано, может по умолчанию использоваться для любых целей, но адрес с конкретным значением этого свойства должен предпочитаться для соответствующей цели.

B.2.21.2 Свойство useablePeriod: GTS

Определение: общая спецификация времени GTS (General Timing Specification), указывающая периоды времени, в течение которых можно использовать данный адрес. Используется для указания разных адресов в зависимости от дня недели или от года.

B.2.21.3 Свойство isNotOrdered: BL

Определение: булевское значение, указывающее, известен ли порядок компонентов адреса или нет. Хотя компоненты адреса всегда передаются в определенном порядке, тем не менее порядок, в котором они должны быть представлены пользователю, может быть, а может и не быть известен. Если он неизвестен, то это может быть указано с помощью свойства isNotOrdered.

B.2.21.4 Свойство equal: BL (унаследовано от типа данных ANY)

Два значения адреса считаются равными, если они содержат одинаковые компоненты адреса (без учета порядка следования). Свойства use и useablePeriod из проверки на равенство исключаются.

```
invariant(AD x, y)
      where x.nonNull.and(y.nonNull) {
   x.equal(y).equal((
      forall(ADXP p) where x.contains(p) {
            y.contains(p);
            )}.and.(
           forall(ADXP p) where x.contains(p) {
                 y.contains(p);
                 )});
);
```

B.2.21.5 Свойство formatted: ST

Определение: строковое значение адреса, форматированное по строкам и с нужными пробельными символами. Это только семантическое свойство, определяющее назначение некоторых типов компонентов адреса[1].

Тип данных AD прежде всего используется для передачи данных, по которым можно печатать этикетки с адресами, позволяющими лицу физически посетить адресата. Люди воспринимают адреса в напечатанном виде, например, на этикетках. В определении типа данных AD точно указано, как форматируются адреса[2].

Адреса являются упорядоченными списками компонентов адреса. Каждый компонент адреса печатается слева направо в том порядке, как он указан в списке (или в ином направлении, специфичном для языка; определение этого направления не входит в область применения настоящей спецификации). Печатается значение каждого компонента адреса. Большинство компонентов окаймляются пробельными символами. Следующие шесть правил регламентируют использование пробельных элементов:

1. Пробельные символы не аккумулируются, то есть два смежных пробельных символа эквивалентны одному. Смежные переходы на другую строку могут быть заменены одним переходом. Пробельные символы вокруг перехода на другую строку не являются значащими.

2. Литералы могут явно содержать пробельные элементы, к которым могут применяться те же самые правила сокращения пробелов. В литерале, указанном внутри текста одного компонента адреса, переходы на новую строку лишены смысла.

3. Явные ведущие и концевые пробельные символы не являются значащими во всех компонентах адреса, за исключением компонентов, у которых свойство partType имеет значение «DEL» (разделитель).

4. По умолчанию компонент адреса окаймляется неявными пробельными символами.

5. Компоненты адреса, у которых свойство partType имеет значение «DEL» (разделитель), неявными пробельными символами не окаймляются.

6. Явные ведущие и концевые пробельные символы являются значащими в компонентах адреса, у которых свойство partType имеет значение «DEL» (разделитель).

Это означает, что в общем случае все компоненты адреса окаймляются пробельными символами, но эти символы не аккумулируются. Компоненты разделителей никогда не окаймляются неявными пробельными символами, и каждый пробельный символ, которым заканчивается предшествующий компонент или начинается следующий компонент, удаляется вне зависимости от того, явный он или неявный.

Ниже показаны примеры представления адреса в форме, определяемой спецификацией реализуемой технологии на языке XML[3].

Адрес

```
    1050 W Wishard Blvd,
RG 5th floor,
Indianapolis, IN 46240.
```

может быть закодирован в любой из указанных ниже форм.

Первая форма используется системой, которая хранит адреса в форме свободного текста или в форме списка строк (строка 1, строка 2 и т. д.).

***Пример 3—***

```
<addr use="WP">
   1050 W Wishard Blvd,
```

---

[1] Следует учесть, что семантические свойства свободны от любой семантики потока управления. Свойство formatted может быть реализовано как «процедура», «возвращающая» форматированный адрес, но обычно оно не является переменной, которой можно присвоить форматированный адрес. Однако стандарт HL7 определяет не приложения, а только семантику передаваемых значений данных. Поэтому семантическая модель абстрагируется от понятий наподобие «процедуры», «возвращения» и «присваивания», а оперирует только свойством и значением.

[2] Правила форматирования адреса являются частью семантики адресов, поскольку адреса в первую очередь являются изображаемым или печатаемым текстом, предназначенным для восприятия человеком. Другие использования (например, в эпидемиологии) являются вторичными. Хотя они и не запрещены, использование типа данных AD в этих случаях может оказаться не идеальным решением. В стандартах HL7 определены более лучшие способы удовлетворения таких сценариев. Обратите внимание, что правила форматирования адреса не относятся к вопросам технологий реализации, поскольку они применяются для представления адреса людям, а спецификации технологии реализации описывают адреса для целей обмена между компьютерами.

[3] Кодирование на языке XML в соответствии со спецификацией реализуемой технологии XML показано здесь лишь для того, чтобы избежать описания других нотаций. Это не означает, что функция работает только на языке XML или что XML является предпочтительным представлением.

```
   RG 5th floor,
   Indianapolis, IN 46240
</addr>
```

Вторая форма содержит более специфичные компоненты адреса по сравнению с первой.

***Пример 4***

```
<addr use="WP">
   <streetAddressLine>1050 W Wishard Blvd</streetAddressLine>,
   <streetAddressLine>RG 5th floor</streetAddressLine>,
   <city>Indianapolis</city>,
   <state>IN</state>
   <postalCode>46240</postalCode>
</addr>
```

Эта форма типична для США, где адрес улицы иногда разделяется на компоненты, но город, штат и почтовый индекс всегда разделены.

Третья форма еще более специфична.

***Пример 5—***

```
<addr use="WP">
   <houseNumber>1050</houseNumber>
   <direction>W</direction>
   <streetName>Wishard Blvd</streetName>,
   <additionalLocator>RG 5th floor</additionalLocator>,
   <city>Indianapolis</city>,
   <state>IN</state>
   <postalCode>46240</postalCode>
</addr>
```

Такая форма в США не используется. Однако она полезна в Германии, где многие системы выделяют номер дома в отдельный компонент. Например, германский адрес

```
   Windsteiner Weg 54a,
D-14165 Berlin
```

скорее всего, будет представлен в описанной ниже форме[1)].

***Пример 6—***

```
<addr use="HP">
   <streetName>Windsteiner Weg</streetName>
   <houseNumber>54a</houseNumber>,
   <country>D</country>-
   <postalCode>14165</postalCode>
   <city>Berlin</city>
</addr>
```

**B.2.22 Тип данных EntityNamePart (ENXP) (специализация типа данных ST)**

Определение: строковое значение, представляющее компонент именования сущности (фамилии, имени, отчества лица). Может иметь свойство partType, обозначающее роль этого компонента в полном именовании, и квалификатор qualifier, детализирующий эту роль. Типичными компонентами именований лиц являются имена, фамилии, обращения и т. д.

---

[1)] Этот пример демонстрирует эффективность применения разметки адреса. Типичная германская система, которая хранит номер дома и название улицы в разных полях, должна напечатать сначала название улицы, а затем номер дома. Это будет неправильным в США, поскольку там номер дома пишется перед названием улицы. Разметка адреса позволяет хранить компоненты адреса в естественном порядке и тем не менее понимать их роль.

Таблица В.25 — Сводка свойств типа данных EntityNamePart

| Имя | Тип | Описание |
|---|---|---|
| partType | CS | Тип компонента указывает, является ли он именем, фамилией, префиксом, суффиксом и т. д. |
| qualifier | SET<CS> | Квалификатор содержит ряд кодов, каждый из которых указывает определенную подкатегорию компонента именования, дополняющую основной тип компонента. Например, можно указать, что имя является прозвищем, фамилия является псевдонимом или предназначена для использования в общедоступных записях |

```
protected type EntityNamePart alias ENXP specializes ST {
   CS       type;
   SET<CS>  qualifier;
};
```

В.2.22.1 Свойство partType: CS

Определение: тип компонента partType указывает, является ли он именем, фамилией, префиксом, суффиксом и т. д.

Таблица В.26 — Домен значений свойства partType типа данных EntityName

| Код | Имя | Определение |
|---|---|---|
| FAM | family (фамилия) | Имя семьи, то есть имя, связывающее с генеалогией. В некоторых культурах (например, в Эритрее) фамилией сына является первое имя его отца |
| GIV | given (имя) | Имя (не следует называть его «первым именем», поскольку в написании именования оно не всегда идет первым) |
| PFX | prefix (префикс) | Префикс имеет жесткую связь с компонентом, непосредственно следующим за ним. У префикса нет неявных концевых пробельных символов (хотя он может иметь неявный ведущий пробельный символ). Следует учесть, что префиксы могут быть обращены в суффиксы |
| SFX | suffix (суффикс) | Суффикс имеет жесткую связь с компонентом, непосредственно предшествующим ему. У суффикса нет неявных ведущих пробельных символов (хотя он может иметь неявный концевой пробельный символ). Суффиксы не могут быть обращены в префиксы |
| DEL | delimiter | Разделитель служит только в качестве литерала, печатаемого в данном представлении именования. Он не имеет неявных ведущих и концевых пробельных символов |

Не каждый компонент именования должен иметь код типа. Если этот код неизвестен, неприменим или просто не определен, то это выражается пустым значением (partType.isNull). Например, по именованию «Rogan Sulma» нельзя сделать уверенное заключение, что является именем, а что фамилией. Может даже оказаться, что «Rogan» является титулом.

Концептуально именования сущностей являются размеченным текстом. Разметка может детально описывать роль каждого компонента именования, если она известна. Компоненты именования присутствуют в нем в том порядке, как они печатаются на этикетке почтового отправления. Этот подход похож на разметку текста HTML или XML.

В.2.22.2 Свойство qualifier: SET<CS>

Определение: квалификатор содержит ряд кодов, каждый из которых указывает определенную подкатегорию компонента именования, дополняющую основной тип компонента. Например, можно указать, что имя является прозвищем, фамилия является псевдонимом или предназначена для использования в общедоступных записях.

Таблица В.27 — Домен значений свойства qualifier типа данных EntityNamePart

| Код | Имя | Определение |
|---|---|---|
| LS | legal status (юридическая форма собственности) | Для организаций этот суффикс указывает юридическую форму собственности, например, «Inc.», «Co.», «AG», «GmbH», «B.V.», ««S.A.», «Ltd.» и т. д. |

*Окончание таблицы В.27*

| Код | Имя | Определение |
|---|---|---|
| AC | academic (ученая степень или звание) | Указывает префикс наподобие «Др.», «M.D.» или «Ph.D.», обозначающий ученую степень или звание |
| NB | nobility (дворянин) | В Европе и Азии все еще есть люди с дворянскими титулами (аристократы). Германское «von» в большей мере является титулом, нежели приставкой. Другими примерами служат «Барон» или «Его Величество Король...» и т. д. В наши дни используется редко, но в некоторых системах все еще присутствует |
| PR | professional (профессиональная принадлежность) | Люди, воспитанные культурой Британской империи, нередко используют в качестве суффиксов аббревиатуры своей профессиональной организации |
| VV | voorvoegsel (приставка) | Приставка (по-голландски «voorvoegsel») наподобие «van» или «de» в прошлом могла указывать дворянское происхождение, но в настоящее время это уже не так. Аналогичные приставки существуют и в других языках, например, в испанском, французском и португальском |
| AD | adopted (при усыновлении или удочерении) | Именование лица, данное ему при усыновлении (удочерении) |
| BR | birth (урожденный) | Именование лица, данное ему в краткий период после рождения. Обычно относится к фамилии, но может присваиваться и именам, если они были позже изменены |
| SP | Spouse (супружеское) | Именование, полученное от партнера в семейных отношениях (отсюда «M»). Обычно фамилия супруга. Следует учесть, что из существования супружеской фамилии нельзя сделать определенный вывод относительно пола супруга |
| CL | callme (зовите меня) | Именование (обычно имя), предпочтительное при прямом обращении к лицу |
| IN | initial (инициал) | Указывает, что компонент именования является инициалом. В инициалы не всегда включается заключительная точка, поскольку это может быть не принято в языках, не использующих латинский алфавит. Инициалы могут содержать более одной буквы, например, инициал «Ph.» может соответствовать имени «Philippe», а «Th.» — имени «Thomas» |
| TITLE | title (титул) | Указывает, что префикс или суффикс является титулом, который применяется к полному именованию, а не только к смежному компоненту |

**В.2.23 Тип данных EntityName (EN) (специализация типа данных LIST<ENXP>)**

Определение: именования физического лица, организации, места или предмета. Последовательность компонентов, например, имени или фамилии, префикса, суффикса и т. д. Примерами именований служат «Джим Боб Уолтон мл.», «Health Level Seven, Inc.», «Озеро Тахо» и т. д. Именование может представлять собой простую строку или состоять из нескольких компонентов, например, «Джим», «Боб», «Уолтон» и «мл.», «Health Level Seven» и «Inc.», «Озеро» и «Тахо».

Таблица В.28 — Сводка свойств типа данных EntityName

| Имя | Тип | Описание |
|---|---|---|
| use | SET<CS> | Один или несколько кодов, информирующих систему или пользователя о том, какое из именований следует выбрать для данной цели |
| validTime | IVL<TS> | Интервал времени, в течение которого данное именование используется или было использовано для этой сущности. Отражает тот факт, что именования людей, мест и предметов могут с течением времени изменяться |

*Окончание таблицы В.28*

| Имя | Тип | Описание |
|---|---|---|
| formatted | ST | Строковое значение адреса, форматированное по строкам и с нужными пробельными символами. Это только семантическое свойство, определяющее назначение некоторых типов компонентов адреса.<br>Следует учесть, что семантические свойства свободны от любой семантики потока управления. Свойство formatted может быть реализовано как «процедура», «возвращающая» форматированный адрес, но обычно оно не является переменной, которой можно присвоить форматированный адрес. Однако стандарт HL7 определяет не приложения, а только семантику передаваемых значений данных. Поэтому семантическая модель абстрагируется от понятий наподобие «процедуры», «возвращения» и «присваивания», а оперирует только свойством и значением. |

Именования концептуально рассматриваются как текст с добавленной логической разметкой. Компоненты именования присутствуют в нем в том порядке, как они будут отображаться, а не в порядке, определяемом именем компонента. Упорядочение компонентов именования является важной характеристикой, которая заменяет необходимость в отдельном свойстве «именование для распечатки». Приложения могут изменить этот порядок в целях учета пожеланий своих пользователей. Этот подход похож на разметку текста HTML или XML (но технически не сводится к представлению XML).

По существу, именования являются последовательностями своих компонентов, но при этом имеют дополнительные свойства use и useablePeriod, указывающие, как и когда именование может использоваться для конкретной цели.

```
type EntityName alias EN specializes LIST<ENXP> {
   SET<CS>  use;
   IVL<TS>  validTime;
   BL       equal(ANY x);
   ST       formatted;
};
```

B.2.23.1 Свойство use: SET<CS>

Определение: один или несколько кодов, информирующих систему или пользователя о том, какое из именований следует выбрать для данной цели.

Таблица В.29 — Домен значений свойства use типа данных EntityName

| Код | Имя | Определение |
|---|---|---|
| C | License (вписано в лицензию) | Именование, вписанное в лицензию, запись, сертификат и т. д. (только если отличается от юридического именования) |
| I | Indigenous/Tribal (племенное) | Например, «Вождь Красное Облако» |
| L | Legal (юридическое) | Известно как традиционное, которое можно использовать в юридических отношениях |
| P | pseudonym (псевдоним) | Самоприсвоенное именование, которое лицо использует или использовало |
| A | Artist/Stage (сценическое) | Псевдоним писателя, сценическое именование и т. д. |
| R | Religious (религиозное) | Например, сестра Мэри Браун, брат Джон |
| SRCH | search (поисковое) | Именование, используемое для поиска или определения совпадения |
| PHON | phonetic (фонетическое) | Произношение наименования |
| SNDX | Soundex | Свертка именования в соответствии с алгоритмом SoundEx |
| ABC | Alphabetic (алфавитное) | Алфавитная транскрипция именования (romaji в Японии) |
| SYL | Syllabic (силлабическое) | Силлабическая транскрипция именования (например, kana в Японии, hangul в Корее) |

*Окончание таблицы В.29*

| Код | Имя | Определение |
|---|---|---|
| IDE | Ideographic (идеографическое) | Идеографическое представление именования (kanji в Японии, китайские иероглифы) |

Именование, для которого свойство use не указано, может по умолчанию использоваться для любых целей, но именование с конкретным значением этого свойства должно предпочитаться для соответствующей цели.

В.2.23.2 Свойство validTime: IVL<TS>

Определение: интервал времени, в течение которого данное именование используется или было использовано для этой сущности. Отражает тот факт, что именования людей, мест и предметов могут с течением времени изменяться.

Тип данных EN соответствует расширению типа данных HistoryItem (HXIT).

В.2.23.3 Свойство equal: BL (унаследовано от типа данных ANY)

Два значения именования считаются равными, если они содержат одинаковые компоненты именования (без учета порядка следования). Свойства use и validTime из проверки на равенство исключаются.

```
invariant(EN x, y)
      where x.nonNull.and(y.nonNull) |
   x.equal(y).equal((
      forall(ENXP p| where x.contains(p) |
            y.contains(p);
            |).and.(
           forall(ENXP p| where x.contains(p) |
                 y.contains(p);
                 |));
```

В.2.23.4 Свойство formatted: ST

Определение: строковое значение адреса, форматированное по строкам и с нужными пробельными символами. Это только семантическое свойство, определяющее назначение некоторых типов компонентов адреса[1)].

Тип данных EN прежде всего используется для передачи именований физических лиц, мест и предметов (сущностей), позволяющих устно и письменно использовать их и ссылаться на них. Люди воспринимают именования в напечатанном виде, например, на этикетках. Поэтому в определении типа данных EN точно указано, как форматируются именования[2)].

Именования являются упорядоченными списками компонентов именования. Каждый компонент именования печатается слева направо в том порядке, как он указан в списке (или в ином направлении чтения, специфичном для языка). Печатается значение каждого компонента именования. Большинство компонентов окаймляются пробельными символами. Следующие шесть правил регламентируют использование пробельных элементов:

1. Пробельные символы не аккумулируются, то есть два смежных пробельных символа эквивалентны одному. Смежные переходы на другую строку могут быть заменены одним переходом. Пробельные символы вокруг перехода на другую строку не являются значащими.

2. Литералы могут явно содержать пробельные элементы, к которым могут применяться те же самые правила сокращения пробелов.

3. Явные ведущие и концевые пробельные символы не являются значащими во всех компонентах именования, за исключением компонентов, у которых свойство partType имеет значение «PFX» (префикс), «SFX» (суффикс), «DEL» (разделитель).

---

1) Следует учесть, что семантические свойства свободны от любой семантики потока управления. Свойство formatted может быть реализовано как «процедура», «возвращающая» форматированное именование, но обычно оно не является переменной, которой можно присвоить форматированное именование. Однако стандарт HL7 определяет не приложения, а только семантику передаваемых значений данных. Поэтому семантическая модель абстрагируется от понятий наподобие «процедуры», «возвращения» и «присваивания», а оперирует только свойством и значением.

2) Правила форматирования именования являются частью семантики именований, поскольку компоненты именований в первую очередь являются изображаемым или печатаемым текстом, предназначенным для восприятия человеком. Обратите внимание, что правила форматирования именований не относятся к вопросам технологий реализации, поскольку они применяются для представления именований людям, а спецификации технологии реализации описывают именования для целей обмена между компьютерами.

4. Компоненты именования, у которых свойство partType имеет значение «DEL» (разделитель), неявными пробельными символами не окаймляются. Ведущие и концевые явные пробельные символы являются в этих компонентах значащими.

5. Компонент именования, у которого свойство partType имеет значение «PFX» (префикс), может иметь только ведущий неявный пробельный символ, но не концевой. Явный концевой пробельный символ в этом компоненте является значащим.

6. Компонент именования, у которого свойство partType имеет значение «SFX» (суффикс), может иметь только концевой неявный пробельный символ, но не ведущий. Явный ведущий пробельный символ в этом компоненте является значащим.

Это означает, что в общем случае все компоненты именования окаймляются пробельными символами, но эти символы не аккумулируются. Компоненты разделителей никогда не окаймляются неявными пробельными символами, и каждый пробельный символ, которым заканчивается предшествующий компонент или начинается следующий компонент, удаляется вне зависимости от того, явный он или неявный.

B.2.23.5 Примеры

Ниже показан очень простой пример кодирования именования «Adam A. Everyman».

***Пример 7***

```
<name>
   <given>Adam</given>
   <given>A.</given>
   <family>Everyman</family>
</name>
```

Не должен упоминаться никакой специальный квалификатор, если его значение неизвестно или неприменимо. Ниже показано интенсивное применение нескольких имен, префиксов, суффиксов, ученых степеней, дворянских титулов, приставок («van») и обозначений профессиональной принадлежности.

***Пример 8 —***

```
<name>
   <prefix qualifier="AC">Dr. phil. </prefix>
   <given>Regina</given>
   <given>Johanna</given>
   <given>Maria</given>
   <prefix qualifier="NB">Grafin </prefix>
   <family qualifier="BR">Hochheim</family>-<family qualifier="SP">Weilenfels</family>
   <suffix qualifier="PR">NCFSA</suffix>
</name>
```

В следующем примере название организации «Health Level Seven, Inc.» показано в форме простой строки.

***Пример 9 —***

```
<name>Health Level Seven, Inc.</name>
```

В следующем примере это же название приведено в полностью структурированной форме.

***Пример 10 —***

```
<name>Health Level Seven, <suffix qualifier="LS">Inc.</suffix></name>
```

В следующем примере японские имя и фамилия показаны в трех формах: идеографической (Kanji), силлабической (Hiragana) и алфавитной (Romaji).

***Пример 11 —***

```
<name use="IDE">
   <family>木村</family>
   <given>通男</given>
</name>
<name use="SYL">
   <family>きむら</family>
   <given>みちお</given>
</name>
<name use="ABC">
```

```
    <family>KIMURA</family>
    <given>MICHIO</given>
</name>
```

**B.2.24 Тип данных TrivialName (TN) (специализация типа данных EN)**

Определение: ограничение типа данных EN, представляющее собой простую строку именования предметов и мест.

Тип данных тривиального именования TN представляет собой вариант типа данных EN, состоящий только из одного компонента именования, у которого нет ни типа компонента, ни какого-либо квалификатора. Поэтому тип данных TN и его единственный компонент именования эквивалентен простой строке символов. Эта эквивалентность выражается в форме понижающего приведения до типа данных ST и повышающего приведения от типа данных ST.

```
type TrivialName alias TN specializes EN {
   demotion   ST;
   promotion  TN (ST x);
};

invariant(TN x) where x.nonNull {
   x.head.nonNull;
   x.tail.isEmpty;
   x.formatted.equal(x.head);
};

invariant(ST x) {
   ((TN)x).head.equal(x);
};
```

Тривиальные именования обычно используются для мест и предметов, например, «Озеро Эри» или «Национальный аэропорт Вашингтона им. Рональда Рейгана».

***Пример 12 —***

```
<name>Озеро Эри</name>
<name>Национальный аэропорт Вашингтона им. Рональда Рейгана</name>
```

**B.2.25 Тип данных PersonName (PN) (специализация типа данных EN)**

Определение: ограничение типа данных EN, используемое, когда именованной сущностью является физическое лицо. Представляет собой последовательность компонентов именования лица, например, имя или фамилия, префикс, суффикс и т. д. Компонент именования лица является ограничением компонента именования сущности, при котором используются только те квалификаторы компонента, которые применимы к именованиям лиц. Поскольку структура типа данных EN в основном определялась требованиями к именованиям лиц, это ограничение весьма минимально.

Так как большая часть функциональности типа данных EN относилась к именованиям лиц, то в целом тип данных PN представляет собой минимальное ограничение квалификаторов компонентов типа данных EN.

```
type PersonName alias PN specializes EN;

invariant(PN this) {
   forall(ENXP part)
         where this.contains(part) {
      part.qualifier.contains("LS").not;
      }
};
```

**B.2.26 Тип данных OrganizationName (ON) (специализация типа данных EN)**

Определение: ограничение типа данных EN, используемое, когда именованной сущностью является организация. Представляет собой последовательность компонентов именования.

Название организации, например, «Health Level Seven, Inc.». Название организации состоит только из нетипизированных частей наименования, префиксов, суффиксов и разделителей.

```
type OrganizationName alias ON specializes EN;

invariant(ON this) {
   forall(ENXP part)
```

```
        where this.contains(part) {
      part.type.implies("FAM").not;
      part.type.implies("GIV").not;
      }
};
```

B.2.26.1 Примеры

Ниже показан пример названия организации «Health Level Seven, Inc.» в форме простой строки.

***Пример 13 —***

```
<name>Health Level Seven, Inc.</name>
```

В следующем примере юридическая форма собственности «Inc.» вынесена в отдельный компонент именования.

***Пример 14 —***

```
<name>Health Level Seven, <suffix qualifier="LS">Inc.</suffix></name>
```

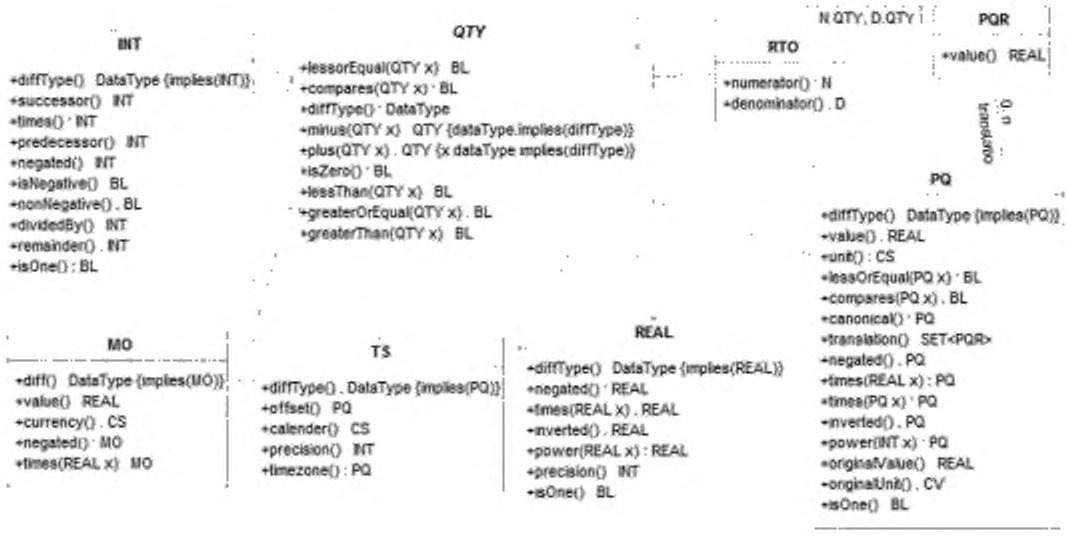


Рисунок B.9 — Типы данных количества

**B.2.27 Абстрактный тип данных Quantity (QTY) (специализация типа данных ANY)**

Определение: тип данных Quantity (количество) представляет собой абстрактное обобщение всех типов данных, у которых: 1) набор значений упорядочен (имеет отношение lessOrEqual) и 2) для всех полностью упорядоченных подмножеств значений типа данных определена операция вычитания. Абстрактный тип количества необходим для определения некоторых других типов, например, интервала и распределения вероятности.

```
abstract type Quantity alias QTY specializes ANY {
   BL    lessOrEqual(QTY x);
   BL    compares(QTY x);
   TYPE  diffType;
   QTY   minus(QTY x);
   QTY   plus(QTY x);
   BL    isZero;
   BL    lessThan(QTY x);
   BL    greaterOrEqual(QTY x);
   BL    greaterThan(QTY x);
};
```

B.2.27.1 Свойство упорядочения lessOrEqual: BL

Определение: предикат, выражающий рефлексивное, симметричное и транзитивное отношение порядка между двумя значениями количества.

Отношение lessOrEqual определено на полностью упорядоченном подмножестве типа данных Quantity, то есть подмножестве, для всех элементов которого задан определенный порядок (например, целые и вещественные числа полностью упорядочены).

Напротив, в частично упорядоченном множестве некоторые, но не все пары элементов сопоставимы с помощью отношения порядка (например, структура дерева или множество физических величин представляют собой частично упорядоченные множества). Два значения данных *x* и *y* упорядоченного типа данных сопоставимы (*x*.compares(*y*)), если между ними существует отношение lessOrEqual (меньше или равно) в ту или иную сторону ($x \le y$ или $y \le x$).

Отношение частичного порядка генерирует полностью упорядоченные подмножества, объединение которых составляет полное множество (например, множество всех величин длины является полностью упорядоченным подмножеством во множестве всех физических величин).

Например, структура дерева частично упорядочена, если корень считать меньше или равным листу, но при этом отношения порядка между листьями может не быть. Физические величины также частично упорядочены, поскольку отношение порядка определено только для величин одной размерности (например, между двумя длинами, но не между длиной и временем). Полностью упорядоченное подмножество дерева представляет собой путь, который транзитивно соединяет лист с корнем. Физическая размерность времени представляет собой полностью упорядоченное подмножество физических величин.

```
invariant (QTY x, y, z)
   where x.nonNull.and(y.nonNull).and(z.nonNull) {
      x.lessOrEqual(x);                                  /* рефлексивное */
      x.lessOrEqual(y).implies(y.lessOrEqual(x)).not; /* асимметричное */
      x.lessOrEqual(y).and(y.lessOrEqual(z))
       .implies(x.lessOrEqual(z))                        /* транзитивное */
};
```

B.2.27.2 Свойство equal: BL (унаследовано от типа данных ANY)

Определение: равенство является рефлексивным, симметричным и транзитивным отношением между двумя значениями данных. Равенство возможно только между правильными значениями, пустые значения никогда не равны (даже если имеют одинаковую причину пустоты).

```
invariant(Quantity x, y, z)
      where x.nonNull.and(y.nonNull).and(z.nonNull) {
   x.equal(x);                                          /* reflexivity */
   x.equal(y).equal(y.equal(x));                        /* symmetry */
   x.equal(y).and(y.equal(z)).implies(x.equal(z))       /* transitivity */
   x.equal(y).implies(x.dataType.equal(y.dataType);
};
```

Способ установления равенства должен быть определен для каждого типа данных. Если иное не указано, то два значения данных равны, если они не различимы, то есть если у них нет различающихся семантических свойств. Это общее определение равенства может быть «переопределено» в типе данных с помощью указания собственного отношения равенства. Такое переопределение отношения равенства может быть использовано для исключения семантических свойств из проверки на равенство. Если в типе данных какие-то семантические свойства исключены из его определения равенства, это означает, что определенные свойства (или аспекты свойств), не ставшие частью проверки на равенство, не существенны для смысла значения.

Например, физическая величина имеет два семантических свойства: (1) вещественное число, (2) кодированная единица измерения. Однако при проверке на равенство необходимо учитывать тот факт, что, к примеру, 1 метр равен 100 сантиметрам. Таким образом, независимые равенства двух семантических свойств являются слишком строгим критерием равенства двух величин. Поэтому в определении типа данных физической величины необходимо переопределить отношение равенства.

B.2.27.3 Свойство compares: BL

Определение: предикат, указывающий, что данное значение и операнд являются сравнимыми и можно определить, какое из них больше другого.

Две величины являются сравнимыми, если они являются элементами общего полностью упорядоченного подмножества пространства значений их типов данных. Определение сравнения основано на свойстве lessOrEqual.

```
invariant (QTY x, y, z)
      where x.nonNull.and(y.nonNull) {
   x.compares(y).equal(x.lessOrEqual(y).or(y.lessOrEqual(x)));
};
```

B.2.27.4 Свойство diffType: TYPE

Определение: свойство diffType определяет тип разности между двумя значениями конкретного типа данных QTY.

```
invariant(QTY x) {
   x.diffType.implies(QTY)
};
```

Тип разности представляет собой некоторый тип данных, который далее специализирует тип данных QTY.

B.2.27.5 Свойство minus: QTY

Определение: Величина, выражающая «дистанцию» данной величины от величины операнда, которая должна быть сравнимой. Тип данных величины разности связан с типом данных операнда, но не обязан быть тем же самым.

```
invariant(QTY x, y) {
   x.minus(y).implies(x.diffType);
};
```

Результат вычитания minus имеет тип данных, возвращаемых в свойстве diffType данной величины.

Разность определена в упорядоченном множестве, если семантически значимо утверждать, что Δ представляет собой разность значений *x* и *y*. Разность Δ должна быть значимой независимо от значений величин *x* и *y*. Такая независимость существует, если для всех значений *u* можно осмысленно вывести такое значение *v*, что Δ будет также разностью между *u* и *v*. Суждение о том, что значимо, не может быть определено формально[1)].

Свойство minus имеет тип данных, способный представить разность двух значений, между которыми существует отношение упорядоченности (то есть эти значения являются элементами общего полностью упорядоченного подмножества). Например, разность двух целых чисел имеет тип данных целого числа, а разность двух моментов времени является физической величиной, имеющей размерность времени. Тип данных разности имеет полностью упорядоченный домен значений.

Разность двух значений *x* и *y* должна быть определена для всех пар *x* и *y* в полностью упорядоченном подмножестве значений типа данных. Нуль представляет собой разность значения с самим собой.

```
invariant(QTY x, y)
      where x.compares(y) {
   x.minus(y).nonNull;
   x.minus(x).isZero;
};
```

Если значения *x* и *y* не сравнимы, то их разность должна иметь пустое значение.

```
invariant(QTY x, y)
      where x.compares(y).Not {
   x.minus(y).notApplicable;
};
```

B.2.27.6 Свойство plus: QTY

Определение: сумма данного количества и значения операнда. Операнд должен иметь тип данных, который может выразить разность между двумя значениями этого типа данных физической величины.

```
invariant(QTY x, y)
      where x.compares(y) {
```

[1)] Абстракция типа данных количества отвечает понятию шкал разности в противовес порядковым шкалам и шкалам отношений (Guttman и Stevens). Тип данных, в котором определены только требования к упорядочению, но не к разности значений, является порядковым. В настоящее время для порядковых типов данных специальный тип данных не определен. Обычно порядковыми величинами являются кодированные значения, взятые из систем кодирования, в которых определено отношение порядка. Но пока что семантика упорядочения не отражена в семантике типов данных, определенных в стандартах HL7.

```
      x.plus(y.minus(x)).equal(y);
      y.dataType.implies(x.diffType);
};
```

Вопрос: как соотносятся друг с другом выражения {y.dataType.implies(x.diffType)} и {x.compares(y)}?

Если тип значения *y* не является допустимым для разности значений, имеющих тип значения *x*, то результатом операции должно быть пустое значение.

```
invariant(QTY x, y)
      where y.dataType.implies(x.diffType).not {
   x.plus(y).notApplicable;
};
```

B.2.27.7 Свойство isZero: BL (является нулем)

Определение: нейтральный элемент операций разности и сложения, то есть если величина равна нулю, то сложение с ней или ее вычитание из другой сравнимой величины должно быть равно этой другой величине.

```
invariant(QTY x) {
   x.minus(x).isZero;
};
```

B.2.27.8 Свойство отношения порядка lessThan: BL (меньше)

Определение: предикат, указывающий асимметричное и транзитивное отношение упорядочения между данной величиной и другой величиной. Это отношение упорядочения то же самое, что и для свойства lessOrEqual, но не является рефлексивным.

```
invariant (QTY x, y, z)
      where x.nonNull.and(y.nonNull) {
   x.lessThan(y).equal(x.lessOrEqual(y)
                .and(x.equal(y).not));
};
```

B.2.27.9 Свойство отношения порядка greaterOrEqual: BL (больше или равно)

Определение: предикат, указывающий асимметричное и транзитивное отношение упорядочения между данной величиной и другой величиной. Оно является обращением отношения lessOrEqual.

```
invariant (QTY x, y, z)
      where x.nonNull.and(y.nonNull) {
   x.greaterOrEqual(y).equal(y.lessOrEqual(x));
};
```

B.2.27.10 Свойство отношения порядка greaterThan : BL (больше)

Определение: предикат, указывающий асимметричное и транзитивное отношение упорядочения между данной величиной и другой величиной. Оно является обращением отношения lessThan.

```
invariant (QTY x, y, z)
      where x.nonNull.and(y.nonNull) {
   x.greaterThan(y).equal(y.lessThan(x));
};
```

**B.2.28 Целочисленный тип данных IntegerNumber (INT) (специализация типа данных QTY)**

Определение: целые числа (-1, 0, 1, 2, 100, 3398129 и т. д.), являющиеся точными числами, полученными в результате подсчета и перечисления. Целые числа являются дискретными, множество целых чисел бесконечно, но счетно. На диапазон целых чисел не накладывается никакого произвольного ограничения. Для положительной и отрицательной бесконечности предусмотрены отдельные причины пустоты.

```
type IntegerNumber alias INT specializes QTY {
           INT  successor;
           INT  times(INT x);
           INT  predecessor;
           INT  negated;
```

```
            BL   isNegative;
            BL   nonNegative;
            INT  dividedBy(INT x);
            INT  remainder(INT x);
            BL   isOne;
  literal   ST;
};
```

Так как тип данных IntegerNumber имеет всю семантику математического понятия целого числа, то определены базовые операции plus (сложение) и times (умножение). Эти операции определены здесь как характеризующие операции в смысле ИСО 11404 по той причине, что они необходимы в других частях настоящей спецификации, а именно для определения семантики литеральной формы.

Традиционное рекурсивное определение сложения и умножения восходит к Грассману и использует понятие «следующее» (successor)[1].

```
invariant(INT x, o, i)
      where x.nonNull.and(o.isZero) {
   x.lessThan(x.successor);
   x.plus(o).equal(x);
   x.plus(y.successor).equal(x.plus(y).successor);
   x.times(o).equal(o);
   x.times(y.successor).equal(x.times(y)).plus(x);
};
```

B.2.28.1 Свойство successor: INT (следующее)

Определение: следующим называется такое значение типа INT, большее данного значения типа INT, для которого не существует никакого значения типа INT, находящегося между данным и следующим значением.

```
invariant(INT x, y)
      where x.successor(y) {
   x.lessThan(y).and.not(exists(INT z) {
      x.lessThan(z);
      z.lessThan(y);
      });
};
```

B.2.28.2 Свойство типа данных разности diffType: TYPE (унаследовано от типа данных QTY)

```
invariant(INT x) {
   x.diffType.implies(INT);
};
```

Тип данных разности (diffType) двух значений типа INT также является типом данных INT.

B.2.28.3 Свойство сложения plus: INT (унаследовано от типа данных QTY)

```
invariant(INT x, y, o)
      where x.nonNull.and(y.nonNull).and(o.isZero) {
   x.plus(o).equal(x);
   x.plus(y.successor).equal(x.plus(y).successor);
};
```

B.2.28.4 Свойство умножения times: INT

Определение: результат умножения данного целого значения на операнд, эквивалентный повторениям сложения этого значения.

---

[1] *H. Grassman*. Lehrbuch der Arithmetik. 1861. Исходным аксиомам, предложенным Грассманом, отдается предпочтение перед аксиомами Пеано, поскольку аксиомы Грассмана охватывают все целые числа, а не только натуральные. Также «довольно хорошо известно, что, по собственному признанию Пеано, он позаимствовал свои аксиомы у Дедекинда и при их разработке существенно использовал труды Грассмана» (Hao Wang. The Axiomatization of Arithmetic. J. Symb. Logic; 1957:22(2); p. 145).

```
invariant(INT x, y, 1, 0)
      where x.compares(y).and(0.isZero).and(1.isOne) {
   x.times(0).equal(0);
   x.times(1).equal(x);
   x.times(y.successor).equal(x.times(y)).plus(x);
};
```

B.2.28.5 Свойство predecessor: INT (предыдущее)

Определение: свойство, обратное свойству successor.

```
invariant(INT x, y)
      where x.successor(y) {
   x.successor.predecessor.equal(x);
};
```

B.2.28.6 Свойство negated: INT (отрицание)

Определение: элемент, обратный значению типа INT, то есть такое другое значение типа INT, которое при сложении с данным значением дает нулевой результат (нейтральный элемент).

```
invariant(INT x)
      where x.nonNull {
   x.plus(x.negated).isZero;
};
```

B.2.28.7 Свойство nonNegative: BL (неотрицательное)

Определение: предикат, указывающий, является ли нулевое значение типа INT (нейтральный элемент) меньшим данного значения типа INT или равным ему.

```
invariant(INT x, 0)
      where x.nonNull.and(0.isZero) {
   x.nonNegative.equal(0.lessOrEqual(x));
};
```

B.2.28.8 Свойство isNegative: BL (отрицательное)

Определение: предикат, указывающий, что данное значение типа INT меньше нуля (не является неотрицательным).

```
invariant(INT x)
      where x.nonNull {
   x.isNegative.equal(x.nonNegative.not);
};
```

B.2.28.9 Свойство dividedBy: INT (деление)

Определение: целочисленным делением данного целого значения (делимого — dividend) на другое целое значение (делитель — divisor) является целое значение, произведение которого на делитель укладывается в делимое.

```
invariant(INT dividend, divisor, 0, 1)
      where divisor.isZero.not.and(0.isZero) {
   dividend.isZero.implies(dividend.dividedBy(divisor).equal(0));
   dividend.isZero.not.implies(dividend.dividedBy(divisor).equal(
      absolute(dividend).minus(absolute(divisor)).dividedBy(absolute(divisor))
                        .successor.times(sign(dividend)
                        .times(sign(divisor))));
};
```

B.2.28.10 Свойство reminder: INT (остаток)

Определение: остаток от целочисленного деления.

```
invariant(INT x, y)
      where x.nonNull.and(y.nonNull) {
```

```
    x.remainder(y).equal(x.minus(x.dividedBy(z).times(y)));
};
```

Данное определение остатка совпадает с определениями, приведенными в языках программирования C и Java.

B.2.28.11 Свойство isOne: BL (нейтральный элемент умножения)

Определение: предикат, указывающий, что данное значение является единицей, то есть нейтральным элементом умножения. Этим свойством обладает ровно одно целое значение.

```
invariant(INT x, y)
      where x.nonNull.and(y.nonNull) {
   x.isOne.and(y.isOne).implies(x.equal(y));
   x.isOne.and(y.isZero).implies(x.equal(y).not);
};
```

B.2.28.12 Литеральная форма

Литеральной формой целого значения является простое десятичное число, то есть строка десятичных цифр.

```
INT.literal ST {
   INT digit : "0"          { $.isZero; }
             | "1"          { $.equal(0.successor); }
             | "2"          { $.equal(1.successor); }
             | "3"          { $.equal(2.successor); }
             | "4"          { $.equal(3.successor); }
             | "5"          { $.equal(4.successor); }
             | "6"          { $.equal(5.successor); }
             | "7"          { $.equal(6.successor); }
             | "8"          { $.equal(7.successor); }
             | "9"          { $.equal(8.successor); };

  INT uint   : digit        { $.equal($1); }
             | uint digit   { $.equal($1.times(9.successor).plus($2)); };

  INT        : uint         { $.equal($1); }
             | "+" uint     { $.equal($2); }
             | "-" uint     { $.equal($2.negated); };
};
```

**B.2.29 Вещественный тип данных RealNumber (REAL) (специализация типа данных QTY)**

Определение: дробные числа. Обычно получаются при измерении и оценке физических величин, а также в результате вычислений, проводимых над другими вещественными числами. Обычно представляются в десятичной форме, где число значащих десятичных разрядов именуется как точность.

В настоящей спецификации термин «вещественное число» используется для обозначения дробных значений, не подразумевая под ними полное множество математических вещественных чисел, включающее в себя иррациональные числа, например, p, число Эйлера и т. д.[1)]

Примечание — В настоящей спецификации вещественный тип данных трактуется в максимально широком смысле. Однако из этого не следует, что любая спецификация реализуемой технологии или реализация, соответствующая стандарту, должна представлять полный диапазон вещественных чисел, что невозможно для любой конечной реализации. В настоящее время сценарии использования вещественных чисел в стандартах HL7 ограничиваются измеряемыми и оцениваемыми физическими величинами, а также денежными суммами. В этих сценариях может использоваться ограниченное пространство вещественных значений, и даже только очень ограниченное множество десятичных чисел (масштабируемых целых значений). Однако при этом объявляются такие представления пространства вещественных значений, как числа с плавающей точкой, рациональные числа, масштабируемые целые числа, строки десятичных цифр, и различные ограничения этих представлений не входят в область применения настоящей спецификации.

В настоящей спецификации предлагаются две альтернативы числового типа данных. Выбор между ними осуществляется следующим образом: числовой атрибут считается вещественным, если точно неизвестно, что он

[1)] Термин «вещественный» применительно к типу данных дробных чисел восходит к языкам программирования Algol и Pascal, где он прочно укоренился.

является целым. Число является целым, если оно всегда является результатом подсчета, обычно представляя порядковый номер. В тех возможных сценариях, когда число является результатом оценки или усреднения, оно не всегда имеет целое значение, и следовательно, необходимо использовать тип данных REAL.

```
type RealNumber alias REAL specializes QTY {
             REAL  negated;
             REAL  times(REAL x);
             REAL  inverted;
             BL    isOne;
             REAL  power(REAL x);
  literal    ST;
             INT   precision;
  demotion   INT;
  promotion  REAL  (INT x);
  promotion  PQ;
  promotion  RTO;
};
```

Указанные здесь алгебраические операции определены как характеризующие операции в смысле ИСО 11404 по той причине, что они необходимы в других частях настоящей спецификации.

В отличие от целых значений семантика вещественных значений не конструируется методом индукции, а только интуитивно описывается на основе соответствующих аксиом об их алгебраических свойствах. Полнота аксиоматики намеренно оставлена в стороне, чтобы не делать никаких утверждений об иррациональных числах.

B.2.29.1 Свойство compares: BL (унаследовано от типа данных QTY)

Множество значений типа REAL является полностью упорядоченным.

```
invariant(REAL x, y)
      where x.nonNull.and(y.nonNull) {
   x.compares(y);
};
```

B.2.29.2 Свойство типа данных разности diffType TYPE (унаследовано от типа данных QTY)

```
invariant(REAL x) {
   x.diffType.implies(REAL);
};
```

Тип данных разности (diffType) двух значений типа REAL также является типом данных REAL.

B.2.29.3 Свойство сложения plus: REAL (унаследовано от типа данных QTY)

```
invariant(REAL x, y, z, o)
      where x.nonNull.and(y.nonNull).and(z.nonNull).and(o.isZero) {
   x.plus(o).equal(x);                                  /* neutral element */
   x.plus(y).plus(z).equal(x.plus(y.plus(z)));          /* associative */
   x.plus(y).equal(y.plus(x));                          /* commutative */
   z.lessOrEqual(x).and(z.lessOrEqual(y)
                  .implies(z.lessOrEqual(x.plus(y));
   x.lessOrEqual(y).implies(x.plus(z)
                  .lessOrEqual(y.plus(z)));
};
```

B.2.29.4 Свойство negated (обратное значение к сложению): REAL

Определение: элемент, обратный значению типа REAL, то есть такое другое значение типа REAL, которое при сложении с данным значением дает нулевой результат (нейтральный элемент сложения).

```
invariant(REAL x)
      where x.nonNull {
   x.plus(x.negated).isZero;
};
```

B.2.29.5 Свойство isOne: BL (нейтральный элемент умножения)

Определение: предикат, указывающий, что данное значение является единицей, то есть нейтральным элементом умножения. Этим свойством обладает ровно одно целое значение.

```
invariant(REAL x, y)
      where x.nonNull.and(y.nonNull) {
   x.isOne.and(y.isOne).implies(x.equal(y));
   x.isOne.and(y.isZero).implies(x.equal(y).not);
};
```

B.2.29.6 Свойство умножения times: REAL

Определение: операция над множеством значений REAL, формирующая абелеву группу и связанная со сложением правилом дистрибутивности.

```
invariant(REAL x, y, z, i, o)
      where x.nonNull.and(y.nonNull).and(z.nonNull)
            .and(i.isOne).and(o.isZero) {
   x.times(o).equal(o);
   x.times(i).equal(x);                                          /* neutral element */
   x.times(y).times(z).equal(x.times(y.times(z)));               /* associative */
   x.times(y).equal(y.times(x));                                 /* commutative */
   x.times(y.plus(z)).equal(x.times(y).plus(x.times(z));         /* distributive */
   o.lessOrEqual(x).and(o.lessOrEqual(y).implies(o.lessOrEqual(x.times(y));
};
```

B.2.29.7 Свойство inverted (обратное значение): REAL

Определение: значение типа REAL, которое при умножении на другое значение типа REAL дает единицу (нейтральный элемент умножения). Нуль (нейтральный элемент сложения) не имеет обратного значения.

```
invariant(REAL x, i)
      where x.isZero.not.and(i.isOne) {
   x.times(x.inverted).equal(i);
};
```

B.2.29.8 Гомоморфизм типа данных INT в тип данных REAL: INT

Определение: типы данных INT и REAL связаны гомоморфизмом, преобразующим каждое значение типа INT в значение типа REAL с сохранением алгебраических операций типа данных INT. Это означает, что целое значение может быть приведено к вещественному, а вещественное может быть понижено до целого числа с помощью округления дробной части.

```
invariant(INT n, m)
      where n.nonNull.and(m.nonNull) {
   ((REAL)n.plus(m)).equal(((REAL)n).plus((REAL)m));
   ((REAL)n.times(m)).equal(((REAL)n).times((REAL)m));
};
```

B.2.29.9 Свойство возведения в степень power: REAL

Определение: основой возведения в степень является повторение умножения вещественного значения, расширенное до рациональной степени с помощью обратной операции извлечения корня.

Ниже перечислены только некоторые общие свойства возведения в степень.

```
invariant(REAL x, y, z, o, i)
      where x.nonNull.and(y.nonNull).and(z.nonNull)
            .and(o.isZero).and.(i.isOne) {
   forall(INT n)
         where n.nonNull {
      n.greaterThan(o).implies(
         x.power(n).equal(x.times(x.power(n.predecessor))));
      n.lessThan(o).implies(
         x.power(n).equal(x.power(n.negated).inverted);
      }
```

```
        x.power(0).equal(1);
        x.power(1).equal(x);
        x.power(y).power(z).equal(x.power(y.times(z)));
        x.power(y).times(x.power(z)).equal(x.power(y.plus(z)));
        x.power(y).inverted.equal(x.power(y.negated));
        x.power(y).power(y.inverted).equal(x);
};
```

B.2.29.10 Литеральная форма

Литеральными формами вещественного значения являются строка десятичных цифр с необязательным ведущим знаком «+» или «–» и необязательной десятичной точкой и необязательная экспоненциальная нотация, в которой используется символ «e» (нечувствительный к регистру) между мантиссой и экспонентой. Число значащих цифр должно соответствовать свойству точности precision.

```
REAL.literal ST {
   REAL            : mantissa                  { $.equal($1); }
                   | mantissa /[eE]/ INT       { $.equal($1.times(10.power($3))); };

   REAL mantissa   : /0*/ 0                    { $.isZero;
                                                 $.precision.equal(1); }
                   | /0*/ "." /0*/             { $.isZero;
                                                 $.precision.equal($3.length.successor); }
                   | /0*/ "." /0*/ fractional  { $.equal($4);
                                                 $.precision.equal($4.precision); }
                   | integer                   { $.equal($1); }
                   | integer "." fractional    { $.equal($1.plus($2));
                                                 $.precision.equal(
                                                 $1.precision.plus($3.precision)); };

   REAL integer    : uintval                   { $.equal($2); }
                   | "+" uintval               { $.equal($1.times($2)); }
                   | "-" uintval               { $.equal($1.times($2).negated); };

   REAL uintval    : /0*/ uint                 { $.equal($2); };

   REAL uint       : digit                     { $.equal($1);
                                                 $.precision.equal(1); }
                   | uint digit                { $.equal($1.times(10).plus($2));
                                                 $.precision.equal(
                                                    $1.precision.successor); };

   REAL fractional : digit                     { $.equal($1.times(10.inverted));
                                                 $.precision.equal(1); }
                   | digit fractional          { $.equal(
                                                 $1.plus($2.times(10.inverted));
                                                 $.precision.equal(
                                                    $1.precision.successor); };

   INT digit       : /[0-9]/                   { $.equal($1); }
};
```

Примерами литералов вещественного значения «две тысячи» служат 2000, 2000., 2e3, 2.0e+3, +2.0e+3.

Следует обратить внимание, что литеральная форма не несет информации о типе данных. Например, «2000» является допустимым представлением как вещественного, так и целого числа. Для отличия от целых значений концевая десятичная точка не используется. Спецификация реализуемой технологии, использующая эту литеральную форму, должна получать информацию о типе значения из других источников.

B.2.29.11 Свойство precision для десятичной формы: INT

Определение: число значащих цифр в десятичном представлении.

Формальное определение точности основано на литеральной форме.

Атрибут precision характеризует только точность десятичного представления, а не точность вещественного значения.

Назначением свойства precision вещественного типа данных является достоверная передача всей информации, представляемой человеку в форме числа. Количество показываемых десятичных цифр несет информацию о степени определенности измеряемого значения (например, число значащих цифр и точность).

Примечание — Точность представления не зависит от степени неопределенности (степени точности) результата измерений. Если важно знать степень неопределенности результата измерений, то этот результат надо передавать как значение типа PPD.

Действуют следующие правила определения значащих цифр:

1) все ненулевые цифры являются значащими;

2) все нули справа от значащей цифры являются значащими;

3) если все цифры числа являются нулями, то первый нуль слева от десятичной точки является значащим (и в соответствии с правилом 2 все следующие нули также являются значащими).

Примечание — Эти правила определения значащих цифр несколько отличаются от более привычных правил, преподаваемых в школе. А именно: здесь все концевые нули перед десятичной точкой последовательно считаются значащими. Иначе, например, в числе 2000, нельзя определить, какие нули являются значащими. Это отклонение от привычного правила предназначено для обеспечения однозначной коммуникации.

Таблица В.30 — Примеры точности литералов вещественных чисел

| Литерал | Число значащих цифр |
|---|---|
| 2000 | 4 значащие цифры |
| 2e3 | 1 значащая цифра, само число могло бы быть записано как «2000», но точность составляет только один десятичный разряд |
| 0.001 | 1 значащая цифра |
| 1e-3 | 1 значащая цифра, само число могло бы быть записано как «0.001», но точность составляет только один десятичный разряд |
| 0 | 1 значащая цифра |
| 0.0 | 2 значащие цифры |
| 000.0 | 2 значащие цифры |
| 0.00 | 3 значащие цифры |
| 4.10 | 3 значащие цифры |
| 4.09 | 3 значащие цифры |
| 4.1 | 2 значащие цифры |

Точность представления должна совпадать с неопределенностью значения. Однако точность представления и неопределенность значения представляют собой два независимых понятия. Детальное обсуждение неопределенности вещественных значений приведено в подразделе PPD<REAL>.

Например, представление «0.123» имеет 3 значащие цифры, но неопределенность значения может быть в любом показанном или не показанном десятичном разряде, например, неопределенность может быть 0.123 ± 0.0005, 0.123 ± 0.005 или 0.123 ± 0.00005 и т. д. Следует учесть, что спецификация реализуемой технологии должна согласовывать свою точность представления с неопределенностью значения. Но поскольку точность строки цифр кратна 0.5 от наименьшего значащего разряда, то неопределенность принимать любое значение между этими «пределами» и 0.123 ± 0.005 будет также адекватным представлением значения, находящегося в диапазоне от 0.118 до 0.128.

Примечание — в спецификации реализуемой технологии, основанной на представлении чисел в форме десятичных строк, точность не обязательно должна быть явным атрибутом. В этом случае подобная спецификация должна предусматривать правила однозначного определения значащих цифр. Представление числа должно обеспечивать не больше, но и не меньше значащих цифр, изначально имевшихся у этого числа. Соответствие можно проверить с помощью циклического преобразования кодирование — декодирование — кодирование.

**B.2.30 Тип данных Ratio (RTO) (специализация типа данных QTY)**

Определение: величина, представленная как отношение величины, указанной в числителе, к величине, указанной в знаменателе. Общие множители числителя и знаменателя автоматически не сокращаются. Тип данных

RTO обеспечивает представление титров (например, «1:128») и других величин, измеряемых в лабораториях и действительно представляющих отношения. Такие отношения не являются просто «структурированными числами», в частности результаты измерения артериального давления (например, «120/60») не являются отношениями. Во многих случаях вместо типа данных RTO можно использовать тип данных REAL.

Отношения отличаются от рациональных чисел тем, что в них общие множители числителя и знаменателя никогда не сокращаются. Отношение двух вещественных или целых чисел автоматически не понижается до вещественного числа.

Таблица B.31 — Сводка свойств типа данных Ratio

| Имя | Тип | Описание |
|---|---|---|
| numerator | N | Числитель — значение, представляющее собой делимое в отношении. По умолчанию является целым числом 1 (один) |
| denominator | D | Знаменатель — значение, представляющее собой делитель в отношении. По умолчанию является целым числом 1 (один). Делитель не должен равняться нулю |

Примечание — Этот тип данных определен не для общего представления рациональных чисел. Он используется только в том случае, когда общие множители числителя и знаменателя не предполагается сокращать. Такое происходит не часто. При передаче результатов измерений отношения встречаются почти исключительно в титрах.

```
type Ratio<QTY N, QTY D> alias RTO specializes QTY {
   N          numerator;
   D          denominator;
   demotion   REAL;
   demotion   PQ;
};
```

По умолчанию свойства numerator и denominator имеют целое значение 1 (один). Свойство denominator не может иметь нулевое значение.

Примечание — Этот тип данных определен как параметризованный (см. B.2.9.5 «Параметризованные типы данных»), но обсуждается в контексте других типов данных, связанных с величинами. Причина определения типа данных отношения RTO как параметризованного состоит в том, что в этом случае можно точно ограничить типы данных числителя и знаменателя.

B.2.30.1 Свойство numerator (числитель): N

Определение: величина, представляющая собой делимое в отношении. По умолчанию является целым числом 1 (один).

B.2.30.2 Свойство denominator (знаменатель): D

Определение: величина, представляющая собой делитель в отношении. По умолчанию является целым числом 1 (один). Делитель не должен равняться нулю.

```
invariant(RTO x)
      where x.nonNull {
   x.denominator.isZero.not;
};
```

B.2.30.3 Литеральная форма

Литеральная форма существует для всех отношений, у которых и числитель, и знаменатель имеют литеральные формы. Литерал отношения представляет собой литерал числителя, за которым в качестве разделителя следует двоеточие, а за ним литерал знаменателя. Если двоеточие и знаменатель отсутствуют, то по умолчанию в качестве знаменателя предполагается целое число 1.

```
RTO.literal ST {
   RTO : QTY           { $.numerator.equal($1);
                         $.denominator.equal((INT)1); };
       | QTY ":" QTY   { $.numerator.equal($1);
                         $.denominator.equal($3); };
};
```

Например, значение титра антител к вирусу краснухи 1:64 может быть представлено, используя литерал «1:64».

**B.2.31 Тип данных PhysicalQuantity (PQ) (специализация типа данных QTY)**

Определение: размерностная величина, представляющая результат измерения.

Таблица B.32 — Сводка свойств типа данных PhysicalQuantity

| Имя | Тип | Описание |
|---|---|---|
| value | REAL | Количественное значение, измеренное в соответствующих единицах физической величины |
| unit | CS | Единица измерения, указанная согласно системе Унифицированных кодов единиц измерения UCUM (Unified Code for Units of Measure) [http:///unitsofmeasure.org] |
| translation | SET<PQR> | Альтернативное представление той же самой физической величины, выраженной в других единицах или в другой системе кодирования единиц, возможно, с другим количеством |
| canonical | PQ | Физическая величина, представленная в канонических единицах. В любой конкретной системе единиц каждой физической размерности может быть присвоена одна каноническая единица. Определение канонических единиц не входит в область применения настоящей спецификации, утверждается лишь, что такие канонические единицы существуют (и могут быть произвольно выбраны) для каждой физической величины. Абстрактная физическая величина равна своей канонической форме |
| diffType | TYPE | Тип данных разности двух значений конкретного типа данных QTY |
| toPQ | REAL | |

```
type PhysicalQuantity alias PQ specializes QTY {
            REAL        value;
            CS          unit;
            BL          equal(ANY x)
            BL          lessOrEqual(PQ x);
            BL          compares(PQ x);
            PQ          canonical;
            SET<PQR>    translation;

            PQ          negated;
            PQ          times(REAL x);
            PQ          times(PQ x);
            PQ          inverted;
            PQ          power(INT x);
            BL          isOne;

  literal   ST;
  demotion  REAL;

            REAL        originalValue;
            CV          originalUnit;
};
```

B.2.31.1 Свойство value: REAL

Определение: количество величины, измеренное в терминах единиц.

B.2.31.2 Свойство unit: CS

Определение: единица измерения, указанная согласно системе Унифицированных кодов единиц измерения UCUM (Unified Code for Units of Measure) [http:///unitsofmeasure.org].

Примечание — Для равенства физических величин не требуется, чтобы у них соответственно совпадали свойства value и unit. Эти свойства относятся только к способу представления физических величин. К примеру, 1 м равен 100 см. Хотя различны и единицы, и количества, эти физические величины равны. Таким образом, надо

не рассчитывать на то, что физическая величина имеет конкретную единицу измерения, а обеспечивать автоматическое преобразование разных сопоставимых единиц.

B.2.31.3 Свойство translation: SET<PQR>

Определение: альтернативное представление той же самой физической величины, выраженной в других единицах или в другой системе кодирования единиц, возможно, с другим количеством.

Семантически физические величины являются результатами действий измерения. Хотя эти величины представляются как пары «значение — единица измерения», семантически физическая величина не ограничивается этим. Для определения, являются ли две физические величины равными, недостаточно провести независимое сравнение их значений и единиц измерения. К примеру, 1 м равен 100 см, но различны и единицы, и количества. Для установления равенства вводится понятие канонической формы.

B.2.31.4 Свойство canonical: PQ

Определение: физическая величина, представленная в канонических единицах. В любой конкретной системе единиц каждой физической размерности может быть присвоена одна каноническая единица. Определение канонических единиц не входит в область применения настоящей спецификации, утверждается лишь, что такие канонические единицы существуют (и могут быть произвольно выбраны) для каждой физической величины. Абстрактная физическая величина равна своей канонической форме.

```
invariant(PQ x, y)
      where x.nonNull.and(y.nonNull) {
   x.canonical.equal(x);
};
```

Например, в системе, основанной на Международной системе единиц (СИ), можно определить каноническую форму как: а) произведение только базовых единиц, б) без приставок, в) с использованием только умножения и возведения в степень (без операций деления), г) с определенным порядком указания базовых единиц (например, m, s, g, ...). В этом случае 1 мм рт. ст. можно представить как 133322 $м^{-1}$ $с^{-2}$. Как видно, правила образования канонической формы могут быть довольно сложными. Однако для семантической спецификации не имеет значения ни как конструируется каноническая форма, ни как выбирается конкретная каноническая форма. Важно лишь, что некоторая каноническая форма может быть определена.

B.2.31.5 Свойство equal: BL (унаследовано от типа данных ANY)

Две физические величины равны, если у их канонических форм равны значения и единицы измерения.

```
invariant(PQ x, y)
      where x.nonNull.and(y.nonNull) {
   x.equal(y).equal(x.canonical.value
            .equal(y.canonical.value).and(
               x.canonical.unit.equal(y.canonical.unit)));
};
```

B.2.31.6 Свойство compares: BL (унаследовано от типа данных QTY)

Две физические величины сравнимы (упорядочены и имеют разность), если у их канонических форм равны единицы измерения.

```
invariant(PQ x, y)
      where x.nonNull.and(y.nonNull) {
   x.compares(y).equal(x.canonical.unit.equal(y.canonical.unit));
};
```

B.2.31.7 Свойство diffType: TYPE (унаследовано от типа данных QTY)

```
invariant(PQ x) {
   x.diffType.implies(PQ);
};
```

Разностью двух физических величин является другая физическая величина с теми же самыми единицами измерения.

```
invariant(PQ x, y)
      where x.compares(y) {
   x.minus(y).canonical.unit.implies(x.canonical.unit);
};
```

B.2.31.8 Свойство isOne: BL (нейтральный элемент умножения)

Определение: предикат, указывающий, что данное значение равно числу один, то есть нейтральному элементу умножения. Существует ровно одна физическая величина с таким свойством, называемая единицей.

```
invariant(PQ x, y)
      where x.nonNull.and(y.nonNull) {
   x.isOne.and(y.isOne).implies(x.equal(y));
   x.isOne.and(y.isZero).implies(x.equal(y).not);
};
```

B.2.31.9 Свойство times: PQ (умножение)

Определение: произведением двух физических величин является произведение их количеств, умноженное на произведение их единиц.

```
invariant(PQ x, y, z, i, o)
      where x.nonNull.and(y.nonNull).and(z.nonNull)
            .and(o.isZero).and(i.isOne) {
   x.times(o).equal(o);
   x.times(i).equal(x);              /* нейтральный элемент */
   x.times(y).times(z).equal(
          x.times(y.times(z)));     /* ассоциативность */
   x.times(y).equal(y.times(x));     /* коммутативность */
   o.lessOrEqual(x).and(o.lessOrEqual(y).implies(o.lessOrEqual(x.times(y));
};
```

B.2.31.10 Свойство inverted: PQ (обратная величина)

Определение: значение типа PQ, которое, будучи умноженным на другое значение типа PQ, дает единицу (нейтральный элемент умножения). Нуль (нейтральный элемент сложения) не имеет обратной величины. Отношение двух сравнимых величин сравнимо с единицей (единица измерения 1).

```
invariant(PQ this, that, one)
      where this.nonNull.and(that.nonNull).and(one.isOne) {
   this.times(this.inverted).equal(one);
   this.compares(that).implies(this.times(that.inverted).equal(one));
};
```

B.2.31.11 Свойство times: PQ (умножение на вещественное число)

Определение: при умножении на вещественное число образуется масштабированная величина. Эта величина сравнима с ее исходной величиной.

Если две величины $Q_1$ и $Q_2$ сравнимы друг с другом, то существует вещественное число r, для которого r**1** = $Q_1 / Q_2$.

```
invariant(PQ x; REAL r)
      where x.nonNull.and(r.nonNull) {
   x.times(r).value.equal(x.value.times(r));
   x.times(r).compares(x);
};
```

B.2.31.12 Гомоморфизм значения типа REAL в величину типа PQ: REAL

Значение типа REAL может быть преобразовано в значение типа PQ с единицей **1** (один). Аналогично физическая величина, сравнимая с единицей, может быть преобразована в вещественное значение.

```
invariant(PQ x, unity)
      where x.nonNull.and.unity.isOne.and(x.compares(unity)) {
   unity.times((REAL)x).equal(x);
};
```

B.2.31.13 Свойство power: PQ (возведение в степень)

Определение: физическая величина может быть возведена в целую степень.

```
invariant (PQ x, i; INT n, o)
      where x.nonNull.and(i.isOne).and(n.nonNull.and(o.isZero) {
```

```
      x.power(0).equal(1);
      n.greaterThan(0).implies(
         x.power(n).equal(x.times(x.power(n.predecessor)))};
      n.lessThan(0).implies(
         x.power(n).equal(x.power(n.negated).inverted);
};
```

B.2.31.14 Свойство plus: PQ (сложение)

Определение: две сравнимые физические величины могут быть сложены.

```
invariant (PQ x, y)
      where x.compares(y) {
   x.canonical.plus(y.canonical).value.equal(
      x.canonical.value.plus(y.canonical.value));
};
```

B.2.31.15 Литеральная форма

Литеральной формой физической величины является литерал вещественного числа, за которым следуют необязательный пробельный символ и строка символов, представляющая допустимый код, взятый из системы Унифицированных кодов единиц измерения UCUM (Unified Code for Units of Measure) [http:///unitsofmeasure.org].

```
PQ.literal ST {
   PQ       : REAL unit     { $.value.equal($1);
                              $.unit.equal($2); }
   CS  unit : ST            { $.value.equal($1);
                              $.codeSystem.equal(2.16.840.1.113883.6.8); };
};
```

Например, литеральная форма 20 минут имеет вид «20 min».

**B.2.32 Тип данных PhysicalQuantityRepresentation (PQR) (специализация типа данных CV)**

Определение: расширение типа данных кодированного значения (CV), представляющее физическую величину, у которой единица берется из какой-либо системы кодирования. Используется для альтернативного представления физической величины.

Таблица B.33 — Сводка свойств типа данных PhysicalQuantityRepresentation

| Имя | Тип | Описание |
|---|---|---|
| value | REAL | Количественное значение, измеренное в единицах физической величины, указанных в конкретной системе кодирования |
| code | ST | Символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9 |
| codeSystem | UID | Указывает систему кодирования, в которой определен код |
| codeSystemName | ST | Общее имя системы кодирования |
| codeSystemVersion | ST | Если применим, дескриптор версии, определенный специально для данной системы кодирования |
| displayName | ST | Имя или название кода, под которым система-отправитель показывает значение кода своим пользователям |
| originalText | ED | Текст или фраза, используемые в качестве основы для кодирования |

```
type PhysicalQuantityRepresentation alias PQR specializes CV {
   REAL  value;
};
```

B.2.32.1 Свойство value: REAL

Определение: измеренное количество величины в терминах единиц, указанных в свойстве code.

B.2.32.2 Свойство code: ST (унаследовано от типа данных CV)

Определение: символ кода, определенный в системе кодирования. Например, «784.0» является символом кода головной боли «784.0», определенным в системе кодирования МКБ-9.

Неисключительное значение типа данных PQR имеет непустое свойство code, значением которого является строка символов, определенная в системе кодирования, идентифицированной в свойстве codeSystem. И наоборот, значение типа PQR, у которого свойство code не имеет значения или имеет значение, не принадлежащее указанной системе кодирования, считается исключительным значением (пустым значением (NULL) с типом пустоты «other»).

```
invariant(PQR x)
      where x.nonNull {
   x.code.nonNull;
};
```

B.2.32.3 Свойство codeSystem: UID (унаследовано от типа данных CV)

Определение: указывает систему кодирования, в которой определено значение свойства code.

Идентификатор системы кодирования должен иметь тип данных UID, позволяющий однозначно указать стандартные системы кодирования HL7, другие стандартные системы кодирования, а также местные системы кодирования. Комитет HL7 должен присваивать идентификатор типа UID каждой из своих таблиц кодов, а также внешним стандартным системам кодирования, которые используются в стандартах HL7. На местах должны использовать свои объектные идентификаторы ИСО (тип данных OID), с помощью которых можно сконструировать глобально уникальные идентификаторы местных систем кодирования.

Под ветвью комитета HL7, 2.16.840.1.113883, подветви 5 и 6 содержат соответственно идентификаторы стандартных систем кодирования HL7 и внешних систем кодирования. Эти ветви ведутся техническим комитетом HL7 Vocabulary Technical Committee.

Неисключительное значение типа PQR (то есть значение типа PQR с непустым свойством code) имеет непустое свойство codeSystem, указывающее систему понятий, в которой определено значение свойства code. Другими словами, если есть код, должна быть и система кодирования.

Примечание — Хотя для каждого непустого значения типа PQR определена конкретная система кодирования, в некоторых обстоятельствах представление значения типа PQR в соответствии со спецификацией реализуемой технологии не нуждается в явном упоминании системы кодирования. Например, когда контекст подразумевает одну и только одну систему кодирования, то ее явное указание стало бы избыточным. Однако в таком случае свойство codeSystem принимает контекстно-зависимое значение по умолчанию и не является пустым.

```
invariant(PQR x)
      where x.code.nonNull {
   x.codeSystem.nonNull;
};
```

Причина пустоты «other» у исключительного значения типа PQR указывает, что понятие не может быть закодировано в указанной системе кодирования. Эта система кодирования, в которой нет такого исключительного понятия, должна быть указана в свойстве codeSystem.

Некоторые домены кодов квалифицированы таким образом, что они могут включать в себя некоторые части подходящей местной системы кодирования, не являющиеся парафразами стандартной системы кодирования (coded with extensibility, CWE — кодированные с расширением). Если поле с квалификатором «CWE» действительно содержит такой местный код, то в свойстве системы кодирования должен быть указан идентификатор местной системы кодирования, из которой взят этот код. Однако в доменах с квалификатором «CWE» местный код является допустимым членом домена, поэтому использование местного кода не является ни ошибкой, ни исключительным значением (с причиной пустоты «other») в смысле настоящей спецификации.

```
invariant(PQR x)
      where x.other {
   x.code.other;
   x.codeSystem.nonNull;
};
```

B.2.32.4 Свойство codeSystemName: ST (унаследовано от типа данных CV)

Определение: общее имя системы кодирования.

Имя системы кодирования не используется для вычислений. Оно может быть указано для облегчения интерпретации человеком значений свойств code и codeSystem. Предполагается, хотя и не является обязательным, что

в спецификациях реализуемой технологии будет предусматриваться указание значения свойства codeSystemName в качестве аннотации к идентификатору UID, рассчитанной на восприятие человеком.

Системы, соответствующие стандарту HL7, не должны функционально полагаться на значение свойства codeSystemName. Это значение не должно модифицировать значение свойства codeSystem и не может существовать без значения свойства codeSystem.

```
invariant(PQR x) {
   x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.32.5 Свойство codeSystemVersion: ST (унаследовано от типа данных CV)

Определение: дескриптор версии, определенный специально для данной системы кодирования (если применим).

Для каждой внешней системы кодирования в стандарте HL7 будет указано, как формируется строка со значением версии. Если для конкретной системы кодирования такое указание в стандарте отсутствует, то для такой системы обозначение версии не имеет определенного значения.

Различные версии одной и той же системы кодирования должны быть совместимыми. Если система кодирования изменена несовместимым образом, то она представляет собой другую систему кодирования, а не другую версию, как бы издатель это ни называл.

Например, издатель классификаций МКБ-9 и МКБ-10 назвал эти системы кодирования «9-м пересмотром» и «10-м пересмотром». Однако МКБ-10 представляет собой полное изменение кодов МКБ, и не является обратно совместимой. Поэтому в целях настоящей спецификации МКБ-9 и МКБ-10 рассматриваются как разные системы кодирования, а не как разные версии одной системы кодирования. Напротив, когда версия «1.0j» системы кодирования LOINC была обновлена до версии «1.0k», комитет HL7 рассматривал это как смену версий, поскольку новая версия была обратно совместимой с предыдущей.

```
invariant(PQR x) {
   x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

B.2.32.6 Свойство displayName: ST (унаследовано от типа данных CV)

Определение: имя или название кода, под которым система-отправитель показывает значение кода своим пользователям.

Свойство displayName включено как для удобства интерпретации человеком значения кода, так и для документирования имени, используемого для изображения понятия пользователю. Оно не имеет функционального значения, не может существовать без кода и никогда не должно модифицировать смысл кода.

Примечания

1 В своих словарных доменах стандарты HL7 предусматривают атрибут «печатаемого имени» (print name). Значения этого атрибута могут использоваться в качестве значений свойства displayName.

2 Имена, предусмотренные для кодов, не могут менять смысл кодированного значения. Поэтому они не должны предоставляться пользователю прикладной системы-получателя, пока не будет уверенности в том, что такое имя адекватно отражает понятие, соответствующее кодированному значению. При коммуникациях нельзя просто полагаться на имя кодированного значения. Основная цель этого имени в обеспечении возможности отладки единиц данных в протоколах HL7 (например, сообщений).

```
invariant(PQR x) {
   x.displayName.nonNull.implies(x.code.nonNull);
};
```

B.2.32.7 Свойство originalText: ED (унаследовано от типа данных CV)

Определение: текст или фраза, используемые в качестве основы для кодирования.

Исходный текст появляется в тех случаях, когда код был присвоен информации не ее источником, а специальным кодировщиком уже после ее появления (кодирование постфактум). Тогда при создании дескриптора понятия исходный текст может существовать без кода.

Примечание — Хотя кодирование постфактум часто осуществляется по информации, представленной как свободный текст, например в форме документов, сканированных изображений или диктовки, мультимедийные данные явным образом не разрешены в качестве исходного текста. Кроме того, свойство исходного текста originalText не может означать ссылку на весь исходный документ. Связи между различными артефактами медицинской информации (например, документом и кодированным результатом) не входят в область применения настоящей спецификации и рассматриваются в других стандартах HL7. Исходный текст должен представлять собой

извлечение из оригинального источника, а не точную копию его содержания или указатель на этот источник. Поэтому исходный текст должен представляться в неформатированном виде.

Значение типа PQR могут иметь непустое свойство исходного текста, даже если свойство code имеет пустое значение. В этом случае свойство originalText является именем или описанием понятия, которое не было закодировано. Такие исключительные значения типа PQR могут также содержать преобразования translation. Такие преобразования обеспечивают непосредственное кодирование понятия, описанного в свойстве originalText.

Тип данных PQR может быть понижен до типа данных ST. В этом случае значение типа ST будет представлено только свойством originalText.

```
invariant(PQR x)
      where x.originalText.nonNull {
   ((ST)x).equal(x.originalText);
};
```

**B.2.33 Тип данных MonetaryAmount (MO) (специализация типа данных QTY)**

Определение: тип данных MO является величиной, представляющей денежную сумму в некоторой валюте, то есть в денежных единицах, используемых в различных экономических регионах. В то время как денежная сумма представляет собой простой вид величины (деньги), курс обмена валют является переменным. В этом и состоит принципиальная разница между типами данных PQ и MO, по которой денежные единицы не являются физическими единицами.

Таблица B.34 — Сводка свойств типа данных MonetaryAmount

| Имя | Тип | Описание |
|---|---|---|
| value | REAL | Денежная сумма в единицах некоторой денежной системы |
| currency | CS | Код денежной единицы в соответствии с ИСО 4217 |
| diffType | TYPE | Тип данных разности между двумя значениями конкретного типа данных QTY |

```
type MonetaryAmount alias MO specializes QTY {
           REAL  value;
           CS    currency;
           MO    negated;
           MO    times(REAL x);
   literal ST;
};
```

B.2.33.1 Свойство value: REAL

Определение: денежная сумма в терминах денежной единицы currency.

Примечание — Значения типа MO обычно имеют точность до 0.01 (один цент, пенс, пайса и т. д.). Для больших сумм важно не хранить значения типа MO в форме вещественных чисел с плавающей точкой, поскольку это может привести к потере точности. Однако настоящий стандарт не определяет внутреннее представление значений типа REAL в форме чисел с фиксированной или плавающей точкой.

Свойство REAL.precision означает точность десятичного представления вещественного значения, а не точность самого значения. В определении типа данных REAL отсутствует понятие неопределенности или точности. Например, результат произведения «1.99 USD» (точность 3) на 7 равен «13.93 USD» (точность 4) и не должен округляться до «13.9» для сохранения исходной точности.

B.2.33.2 Свойство currency: CS

Определение: код денежной единицы в соответствии со стандартом ИСО 4217.

Таблица B.35 — Домен значений свойства currency

| Код | Имя | Определение |
|---|---|---|
| ARS | Argentine Peso | Аргентинское песо, денежная единица Аргентины |
| AUD | Australian Dollar | Австралийский доллар, денежная единица Австралии |

*Окончание таблицы В.35*

| Код | Имя | Определение |
|---|---|---|
| BRL | Brazilian Real | Бразильский реал, денежная единица Бразилии |
| CAD | Canadian Dollar | Канадский доллар, денежная единица Канады |
| CHF | Swiss Franc | Швейцарский франк, денежная единица Швейцарии |
| CLF | Unidades de Formento | Единица развития, денежная единица Чили |
| CNY | Yuan Renminbi | Юань, денежная единица Китая |
| DEM | Deutsche Mark | Немецкая марка, денежная единица Германии |
| ESP | Spanish Peseta | Испанская песета, денежная единица Испании |
| EUR | Euro | Евро, денежная единица Европейского Союза |
| FIM | Markka | Марка, денежная единица Финляндии |
| FRF | French Franc | Французский франк, денежная единица Франции |
| GBP | Pound Sterling | Фунт стерлингов, денежная единица Великобритании |
| ILS | Shekel | Шекель, денежная единица Израиля |
| INR | Indian Rupee | Индийская рупия, денежная единица Индии |
| JPY | Yen | Йена, денежная единица Японии |
| KRW | Won | Вона, денежная единица Кореи (Южной) |
| MXN | Mexican Nuevo Peso | Мексиканский новый песо, денежная единица Мексики |
| NLG | Netherlands Guilder | Голландский гульден, денежная единица Нидерландов |
| NZD | New Zealand Dollar | Новозеландский доллар, денежная единица Новой Зеландии |
| PHP | Philippine Peso | Филлипинский песо, денежная единица Филлипин |
| RUR | Russian Ruble | Российский рубль, денежная единица Российской Федерации |
| THB | Baht | Бат, денежная единица Таиланда |
| TRL | Lira | Лира, денежная единица Турции |
| TWD | Taiwan Dollar | Тайваньский доллар, денежная единица Тайваня |
| USD | US Dollar | Доллар США, денежная единица США |
| ZAR | Rand | Ранд, денежная единица Южной Африки |

В таблице В.35 показано только представительное подмножество кодов, определенных в стандарте ИСО 4217. Все коды из ИСО 4127 являются допустимыми значениями этого атрибута.

B.2.33.3 Свойство equal: BL (унаследовано от типа данных ANY)

Два значения типа MO равны, если совпадают значения их атрибутов value и currency.

```
invariant(MO x, y)
      where x.nonNull.and(y.nonNull) {
   x.equal(y).equal(x.value.equal(y.value)
            .and(x.unit.equal(y.unit))};
};
```

B.2.33.4 Свойство compares: BL (унаследовано от типа данных QTY)

Два значения типа MO сравнимы (упорядочены и имеют разность), если равны их свойства currency.

Если их свойства currency не идентичны, то нельзя сравнивать значения их свойств values. Конверсия валют не входит в область применения настоящей спецификации. На практике обменные курсы валют чрезвычайно изменчивы не только в течение длительных и кратких периодов времени, но даже зависят от места и доступности обменных пунктов.

```
invariant(MO x, y)
      where x.nonNull.and(y.nonNull) {
   x.compares.equal(x.currency.equal(y.currency));
};
```

B.2.33.5 Тип данных diffType: TYPE (унаследован от типа данных QTY)

```
invariant(INT x) {
   x.diffType.implies(MO);
};
```

Разность между двумя значениями типа MO имеет тип данных MO.

B.2.33.6 Свойство plus: MO

Определение: два значения типа MO можно сложить, если у них равны значения свойство currency.

```
invariant (MO x, y)
      where x.currency.equal(y.currency) {
   x.plus(y).currency.equal(x.currency);
   x.plus(y).value.equal(x.value.plus(y.value));
};
```

B.2.33.7 Свойство times: MO (умножение на значение типа REAL)

Определение: при умножении на значение типа REAL образуется масштабированная сумма. Она сравнима с исходной суммой.

```
invariant(MO x; REAL r)
      where x.nonNull.and(r.nonNull) {
   x.times(r).value.equal(x.value.times(r));
   x.times(r).currency.equal(x.currency);
};
```

B.2.33.8 Литеральная форма

Литеральная форма типа данных MO состоит из кода валюты, необязательного пробельного символа и литерала значения REAL, представляющего величину суммы.

```
MO.literal ST {
   MO           : currency value  { $.currency.equal($1); }
                                    $.value.equal($2);
   CS  currency : ST               { $.currency.value.equal($1);
                                     $.currency.codeSystem
                                     .equal(2.16.840.1.113883.6.9); }
   REAL value   : REAL            { $.value.equal($1); }
};
```

Например, строка «USD189.95» является литералом суммы 189.95 долларов США.

**B.2.34 Тип данных Calendar (CAL) (специализация типа данных SET<CLCY>)**

Определение: календарь представляет собой понятие измерения различных циклов времени. Такими циклами являются годы, месяцы, дни, часы, минуты, секунды и недели. Некоторые из этих циклов синхронизированы, а некоторые — нет (например, недели и месяцы не синхронизированы).

После «сворачивания оси времени» в эти циклы календарь представляет момент времени как последовательность целых значений счетчиков циклов, например, год, месяц, день, час и т. д. Точкой отсчета календаря является некоторый согласованный момент времени, называемый «эрой».

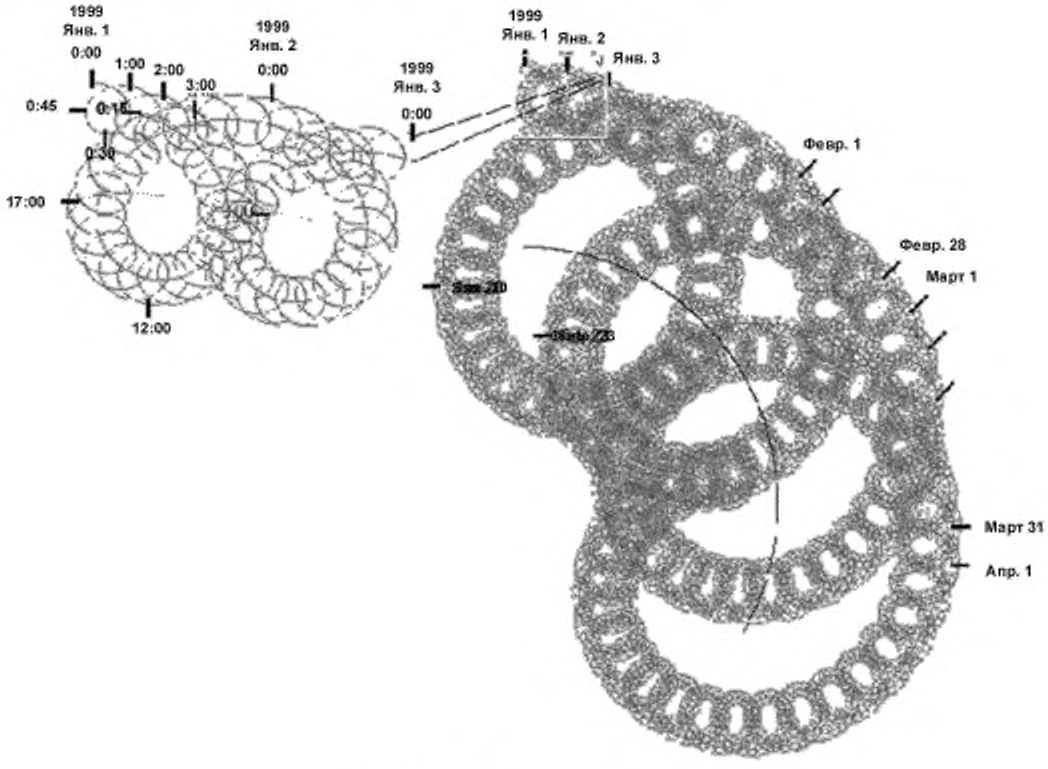


Рисунок B.10 — Календарные циклы

Календарь «сворачивает» ось времени в сложную спираль в соответствии с периодами календаря: год (синий), месяц (желтый), день (зеленый), час (красный) и т. д. Разные циклы могут не совпадать по фазе, например, неделя (не показана) не совпадает по фазе с месяцем[1)].

Тип данных календаря Calendar имеет имя и код и определяется как множество календарных циклов. Свойство Calendar.head представляет наибольший календарный цикл, который в календарном выражении указан крайним справа. Свойство epoch является началом отсчета календаря, то есть моментом начала всех календарных циклов.

```
private type Calendar alias CAL specializes SET<CLCY>  {
   CS    name;
   CLCY  head;
   TS    epoch;
};

invariant(CAL c)
```

[1)] Представьте себе специальные часы, измеряющие эти циклы, в которых все стрелки не имеют общей оси, а каждая стрелка прикреплена к концу другой стрелки, измеряющей следующий больший цикл.

```
        where c.nonNull {
    c.name.nonNull;
    c.contains(c.head);
};
```

В таблице В.36 показано определение современного григорианского календаря. В каждой строке этой таблицы показан календарный цикл. Календарные единицы зависят друг от друга и показаны в графе «Имя». В графе «Порядок» показан порядок следования свойств. Другие графы показаны в соответствии с формальным определением календарного цикла[1)].

Таблица В.36 — Домен календарных циклов

| Имя | Код 1 | Код 2 | Порядок | Цифры | Начало | Условие |
|---|---|---|---|---|---|---|
| year (год) | Y | CY | 1 | 4 | 0 | |
| month of the year (месяц года) | M | MY | 2 | 2 | 1 | |
| month (continuous) (непрерывный месяц) | | CM | | | 0 | |
| week (continuous) (непрерывная неделя) | W | CW | | | 0 | |
| week of the year (неделя года) | | WY | | 2 | 1 | |
| day of the month (день месяца) | D | DM | 3 | 2 | 1 | |
| day (continuous) (непрерывный день) | | CD | | | 0 | |
| day of the year (день года) | | DY | | 3 | 1 | |
| day of the week (begins with Monday) (день недели, начиная с понедельника) | J | DW | | 1 | 1 | |
| hour of the day (час дня) | H | HD | 4 | 2 | 0 | |
| hour (continuous) (непрерывный час) | | CH | | | 0 | |
| minute of the hour (минута часа) | N | NH | 5 | 2 | 0 | |
| minute (continuous) (непрерывная минута) | | CN | | | 0 | |
| second of the minute (секунда минуты) | S | SN | 6 | 2 | 0 | |
| second (continuous) (непрерывная секунда) | | CS | | | 0 | |

**В.2.35 Тип данных календарного цикла CalendarCycle (CLCY) (специализация типа данных ANY)**

Определение: календарный цикл определяет одну группу цифр в календарном выражении. Примерами календарных циклов служат год, месяц, день, час, минута, секунда и неделя.

Календарный цикл имеет имя и два кода, один состоит из одной буквы, а другой — из двух букв. Свойство ndigits указывает число десятичных цифр, занимаемых в календарном выражении. Свойство start указывает, с какого значения начинается отсчет (например, с 0 или 1). Свойство next указывает следующий по порядку меньший цикл в календарном выражении. Свойство max(t) представляет максимальное число циклов за время t (зависит от времени t с учетом високосных лет и секунд). Свойство value(t) представляет целое число циклов, показанных в календарном выражении для времени t. Свойство sum(t, n) представляет собой результат сложения n календарных циклов с временем t.

```
pr.vate type CalendarCycle alias CALCY specializes ANY {
    CE      name;
```

[1)] В настоящее время свойства sum и value календарного цикла формально не определены. Вычисление цифр календаря включает в себя довольно сложный алгоритм, описание которого в настоящей спецификации было бы сложным для понимания и оценки правильности. К несчастью, не существует стандарта, содержащего формальное определение связей между календарными выражениями и временем, прошедшим от начала эры. Спецификации языка ACH.1, типов данных XML-схемы и языка SQL92 ссылаются на стандарт ISO 8601, однако этот стандарт описывает только синтаксис календарных выражений григорианского календаря, а не их семантику. В настоящем стандарте формально определены и синтаксис, и семантика, однако семантика свойств sum и value в нем не описана.

```
    INT     ndigits;
    INT     start;
    CALCY   next;
    INT     max(TS);
    TS      sum(TS t, REAL r);
    INT     value(TS t);
);

invariant(CALCY c)
      where c.nonNull {
   c.name.nonNull;
   c.start.equal(0).or(c.start.equal(1));
   c.digits.greaterThan(0);
);
```

**B.2.36 Тип данных момента времени PointInTime (TS) (специализация типа данных QTY)**

Определение: величина, указывающая момент на оси естественного времени. Момент времени чаще всего представляется как календарное выражение.

Однако семантически время не зависит от календарей и лучше всего может быть описано с помощью связи с прошедшим временем (измеряемым как физическая величина, имеющая размерность времени). Сумма значения типа TS с прошедшим временем дает другое значение типа TS. И наоборот, вычитание значения типа TS из другого значения типа TS дает прошедшее время.

Поскольку никто не знает, когда началось время, тип данных TS концептуализируется как количество времени, прошедшее с некоторого произвольного момента отсчета, называемого началом эры. Так как на оси времени нет абсолютной (нулевой) точки отсчета, то естественное время является разностной величиной, для которой определена только разность, но не отношение (например, нельзя сказать, что какой-то момент времени «вдвое больше» другого момента времени).

Задав некоторую произвольную точку отсчета времени, любой момент времени можно представить в терминах времени, прошедшего от этой точки. Такая произвольная точка отсчета называется началом эры. В настоящей спецификации используется форма смещения от начала эры, но при этом не требуется, чтобы в каждой системе тип данных TS был реализован именно таким способом. Системам, которым не требуется вычислять дистанции между значениями типа TS, достаточно ограничиваться литеральными представлениями календарных выражений.

```
type PointInTime alias TS specializes QTY {
           PQ   offset;
           CS   calendar;
           INT  precision;
           PQ   timezone;
           BL   equal(ANY x);
  literal  ST;
);
```

B.2.36.1 Свойство offset: PQ

Определение: время, прошедшее от любого постоянного начала эры и измеряемое как физическая величина, размерностью которой является время (например, сравнимо с одной секундой).

```
invariant(TS x)
      where x.nonNull {
   x.offset.compares(1 s);
);
```

Настоящий стандарт не требует определения канонического начала эры; семантика остается той же самой для любой эры при условии, что ее начало постоянно.

Примечание — Свойство offset может трактоваться как чисто семантическое свойство, которое не обязано быть представлено иначе как в литерале календарного выражения. Однако в спецификации реализуемой технологии может быть выбрано собственное постоянное начало эры, и значения типа TS могут быть в ней представлены как смещение времени относительно начала эры. Но в такой технологии, использующей другие календари, тем не менее существует потребность в передаче кода календарного цикла и точности календарного представления.

B.2.36.2 Свойство equal: BL (унаследовано от типа данных QTY)

Два значения типа данных TS равны в том и только том случае, если равны их свойства offset (отсчитываемые от одного и того же начала эры).

```
invariant(TS x, y)
      where x.nonNull.and(y.nonNull) {
   x.equal(y).equal(x.offset.equal(y.offset));
};
```

B.2.36.3 Свойство calendar: CS

Определение: код, указывающий календарь, используемый для литерального представления данного значения типа TS[1].

Таблица B.37 — Домен значений свойства calendar

| Код | Имя | Определение |
|---|---|---|
| GREG | Gregorian | Григорианский календарь используется в большинстве стран, находящихся под влиянием христианской веры примерно с 1582 года. Он заменил юлианский календарь |

Свойство calendar определено в основном для целей достоверного определения, что пользователь должен видеть или вводить в системе, оперирующей данным значением типа TS. Оно также служит рекомендацией, какой календарь следует использовать системе, отображающей литеральную форму значения типа TS. Однако его значение является лишь рекомендацией, и система, отображающая литеральную форму значений типа TS, может выбрать любой календарь и литеральную форму, необходимую ее пользователям, а не обязательно тот календарь, что указан в свойстве calendar. Таким образом, свойство calendar не является константой при передаче данных из одной системы в другую и поэтому не является частью проверки на равенство.

Для целей определения отношения между календарным выражением и значением в форме начала эры/смещения введены два типа данных, а именно CAL и CLCY, имеющих область видимости private. Они не могут использоваться вне данной спецификации.

B.2.36.4 Свойство precision: INT (точность календарной литеральной формы)

Определение: число значащих цифр в представлении календарного выражения.

Свойство precision формально определено на основе определения свойства literal (см. подраздел B.2.31.9).

Свойство precision характеризует только точность десятичного представления, а не точность самого значения типа TS.

Это свойство предназначено для достоверной оценки информации, представленной человеку в календарном выражении. Показываемое ему число цифр содержит информацию о неопределенности (а именно степени точности) измеренного значения типа TS.

Примечание — Точность представления не зависит от степени неопределенности (степени точности) результата измерений. Если важно знать степень неопределенности результата измерений, то этот результат надо передавать как значение типа PPD.

Значение свойства precision зависит от значения свойства calendar. Данное значение свойства precision, относящееся к одному календарю, не обязательно будет иметь тот же смысл в календаре с другими периодами.

Например, строка «20000403» имеет 8 значащих цифр в представлении, но значение типа TS, имеющего это представление, может быть неопределенным в любой показанной или не показанной цифре, например, не иметь точного дня, недели или часа. Точность внешнего представления должна коррелировать с точностью самого значения. Однако точность строки цифр зависит от календаря и может быть определена только в терминах календарных периодов, а точность значения может не укладываться в эти периоды (например, строка «2000040317» является адекватным представлением значения, находящегося в границах «2000040305» и № 2000040405»).

Примечание — В спецификации реализуемой технологии, основанной на литеральном представлении календарных выражений, точность не обязательно должна быть явным атрибутом. В этом случае подобная спецификация должна предусматривать правила однозначного определения значащих цифр. Представление значения типа TS

[1] В настоящее время никакие другие календари, кроме григорианского, не определены. Однако понятие календаря как произвольного соглашения об абсолютном времени является важным для правильного определения семантики времени и типов данных, относящихся к времени. Кроме того, для облегчения использования стандартов HL7 другими культурами может потребоваться поддержка иных календарей.

должно обеспечивать не больше, но и не меньше значащих цифр, изначально имевшихся у этого значения. Соответствие можно проверить с помощью циклического преобразования кодирование — декодирование — кодирование.

B.2.36.5 Свойство timezone: PQ (часовая зона)

Определение: разность между местным временем в данной часовой зоне и Всемирным координируемым временем (Universal Coordinated Time, UTC), которое ранее именовалось средним временем по Гринвичу (Greenwich Mean Time, GMT). Часовая зона имеет тип данных PQ с размерностью времени (то есть сравнима с одной секундой). Нулевая часовая зона означает UTC. Значение часовой зоны не позволяет сделать вывод о географической долготе или о согласованном названии часовой зоны.

Например, значение 200005121800-0500 может быть Восточным стандартным временем (Eastern standard time, EST) в Индианаполисе (Индиана) или Центральным летним временем (Central daylight savings time, CDT) в Декатуре (Иллинойс). В других странах, имеющих иную долготу, часовые зоны могут иметь другие именования.

```
invariant(TS x, y)
      where x.nonNull.and(y.nonNull) {
   x.timezone.compares(1 s);
};
```

Если свойство timezone имеет пустое значение (причина пустоты «UNK» — неизвестно), то подразумевается «местное время». Однако местное время всегда относится к какому-то месту, и без знания этого места часовая зона неизвестна. Поэтому местное время не может быть переведено в значение UTC. Чтобы избежать значительной потери точности при сравнении значений типа TS, свойство timezone должно быть задано для всех значений типа TS. Разность двух значений местного времени, если неизвестно, к каким местам они относятся, может иметь ошибку ±12 часов.

В контексте административных данных некоторые значения времени не должны иметь часовую зону. Например, для административной даты рождения указание часовой зоны является некорректным, поскольку при преобразовании в другую часовую зону эта дата может измениться. У таких административных данных свойство timezone должно быть пустым с причиной пустоты «NA» (неприменимо).

B.2.36.6 Свойство diffType: TYPE (унаследовано от QTY)

```
invariant(TS x) {
   x.diffType.implies(PQ);
};
```

Свойство diffType означает тип данных разности между двумя значениями типа TS, а именно тип PQ с размерностью времени.

B.2.36.7 Свойство plus: TS (унаследовано от типа данных QTY)

Определение: свойство plus определяет сложение значения типа TS со значением прошедшего времени (имеющего тип данных PQ с размерностью времени), имеющее тип данных TS.

```
invariant(TS x, PQ t)
      where x.nonNull.and(t.compares(1 s)) {
   x.plus(t).offset.equal(x.offset.plus(t));
};
```

B.2.36.8 Свойство minus: QTY (унаследовано от типа данных QTY)

Определение: свойство minus представляет собой разность двух значений типа TS, являющуюся прошедшим временем.

```
invariant(TS x)
      where x.nonNull {
   x.minus(y).offset.equal(
      x.offset.plus(y.offset.negated));
};
```

B.2.36.9 Литеральная форма

Литералы типа данных TS представляют собой простые календарные выражения, составленные на основе таблицы определения календаря. По умолчанию должен использоваться западный (григорианский) календарь (таблица B.36).

Согласно настоящей спецификации, литералы календарных выражений григорианского календаря, используемого по умолчанию, соответствуют ограничению ИСО 8601, определенному в разделе 32 (всемирное время) ИСО 8824 (ASN.1) и в описании типа данных TS, приведенному в стандарте HL7 Версии 2.

Литеральные выражения являются последовательностями целых чисел, упорядоченных в соответствии с графой «Порядок» таблицы В.36. Периоды с меньшими порядковыми номерами записываются слева от периодов с большими порядковыми номерами. Периоды, которым порядковые номера не присвоены, не должны включаться в календарное выражение типа данных TS.

В графе «Цифры» таблицы В.36 указано точное число цифр в числе, задающем конкретный период.

Таким образом, в соответствии с таблицей В.36 календарные выражения западного календаря начинаются с четырех цифр года (отсчет начинается с нуля), за которыми следуют две цифры месяца (отсчет начинается с 1), за ними две цифры дня месяца (начиная с 1), две цифры часа дня (начиная с нуля) и т. д. Например, строка «200004010315» является допустимым выражением для 3 ч 15 м 1 апреля 2000 г.

Опуская правые компоненты, можно получить календарные выражения разной точности.

Например, строка «20000401» является календарным выражением с точностью только до дня месяца.

Наименьший определенный период календаря (например, секунда) может быть записан как число типа REAL, в котором указаны цифры целой части секунд, за ними десятичная точка и любое число цифр дробной части.

Например, строка «20000401031520.34» означает 20,34 с 3 ч 15 м 1 апреля 2000 г.

Если когда-нибудь будут использоваться другие календари, то перед западным (григорианским) календарным выражением может быть указан префикс «GREG:», отличающий его от других календарей. Каждый календарь должен иметь собственный префикс. Но если префикс не указан, то по умолчанию используется западный календарь.

В современном григорианском календаре (и во всех календарях, у которых время дня основано на Всемирном координируемом времени UTC) календарное выражение может содержать суффикс часовой зоны. Он начинается со знака плюса (+) или минуса (–), за которым указаны цифры циклов часов и минут. Часовая зона UTC обозначается как смещение «+00» or «–00»; суффикс «Z», предложенный для этой зоны в стандартах ИСО 8601 и ИСО 8824, не разрешен.

```
TS.literal ST {
   TS    : cal timestamp($1)                  { $.equal($2); }
         | timestamp(GREG)                    { $.equal($1); };

   TS timestamp(Calendar C)
         : cycles(C.head, C.epoch) zone(C)    { $.equal($1.minus($2)); }
                                                $.timezone.equal($2); }
         | cycles(C.head, C.epoch)            { $.equal($1);
                                                $.timezone.unknown; };
   Calendar cal
         : /[a-zA-Z_][a-zA-Z0-9_]*:/          { $.equal($1); };

   TS cycles(CalendarCycle c, TS t)
         : cycle(c, t) cycles(c.next, $1)     { $.equal($2); }
         | cycle(c, t) "." REAL.fractional    { $.equal(c.sum($1, $3));
                                                $.precision.equal(
                                                   t.precision.plus($3.precision)); }
         | cycle(c, t)                        { $.equal($1); };

   TS cycle(CalendarCycle c, TS t)
         : /[0-9]{c.ndigits}/                 { $.equal(c.sum(t, $1));
                                                 $.precision.equal(
                                                    t.precision.plus(c.ndigits)); };

   PQ zone(Calendar C)
         : "+" cycles(C.zonehead, C.epoch)    { $.equal($2.minus(C.epoch)); }
         | "-" cycles(C.zonehead, C.epoch)    { $.equal(C.epoch.minus($2)); };
};
```

### B.3 Общие коллекции

Диаграмма классов типов данных общих коллекций приведена на рисунке B.11.

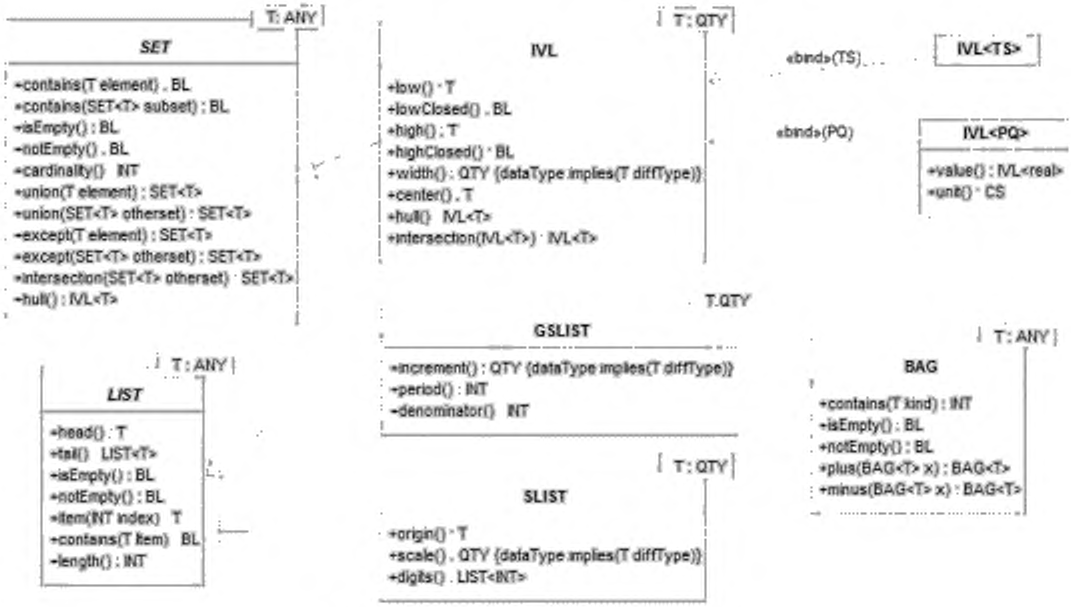


Рисунок B.11 — Типы данных общих коллекций

В этом разделе описаны типы данных, представляющие собой «коллекции» других данных, а именно множество SET, список LIST, «мешок» BAG и интервал IVL[1]. Эти типы коллекций определены как параметризованные типы данных. Понятие параметризованного типа данных описано в подразделе B.1.9.5 «Параметризованные типы данных».

#### B.3.1 Тип данных множества (SET) (специализация типа данных ANY)

Определение: значение, представляющее собой неупорядоченную совокупность различных однотипных значений.

```
template<ANY T>
type Set<T> alias SET<T> specializes ANY {
            BL      contains(T element);
            BL      isEmpty;
            BL      notEmpty;
            BL      contains(SET<T> subset);
            INT     cardinality;
            SET<T>  union(SET<T> otherset);
            SET<T>  union(T element);
            SET<T>  except(T element);
            SET<T>  except(SET<T> otherset);
            SET<T>  intersection(SET<T> otherset);
  literal   ST;
  promotion SET<T>  (T x);
            IVL<T>  hull;
};
```

[1] В некоторых языках программирования под типами «коллекций» понимаются контейнеры индивидуально перечисляемых типов данных, поэтому интервал (нижняя граница — верхняя граница) в них не считался бы коллекцией. Однако такая узкая интерпретация «коллекции» существенно зависит от представления/реализации. С позиции семантики типа данных не существенно, «действительно ли элемент коллекции содержится в коллекции». Поэтому элементы коллекции не обязательно являются индивидуально перечисляемыми.

B.3.1.1 Свойство contains: BL

Определение: отношение множества к своим элементам; принимает значение «true», если данное значение является элементом множества.

Это примитивное свойство множества, с помощью которого определяются все другие свойства.

Множество может содержать только непустые различные элементы. Исключительные (пустые) значения не могут быть элементами множества.

```
invariant(SET<T> s, T n)
      where s.nonNull.and(n.isNull) {
   s.contains(n).not;
};
```

B.3.1.2 Свойство contains: BL

Определение: отношение множества к своим подмножествам, при котором каждый элемент подмножества является также элементом этого множества.

```
invariant(SET<T> superset, subset)
      where superset.nonNull.and(subset.nonNull)
         superset.contains(subset).equal(
      forall(T element) where subset.contains(element) {
         superset.contains(element);
            });
};
```

Отсюда вытекает, что пустое множество является подмножеством каждого множества, в том числе самого себя.

B.3.1.3 Свойство notEmpty: BL

Определение: предикат, указывающий, что данное множество содержит элементы.

```
invariant(SET<T> set)
      where set.nonNull {
   set.notEmpty.equal(exists(T element) {
      set.contains(element);
      });
};
```

B.3.1.4 Свойство isEmpty: BL

Определение: предикат, указывающий, что данное множество не содержит элементов (отрицание свойства notEmpty). Пустое множество является допустимым значением множества, а не исключительным.

```
invariant(SET<T> set)
      where set.nonNull {
   set.isEmpty.equal(notEmpty.not);
};
```

B.3.1.5 Свойство cardinality: INT

Определение: свойство cardinality (мощность) указывает число различных элементов множества.

```
invariant(SET<T> set)
      where set.nonNull {
   exists(T element) where set.contains(element) {
      set.cardinality.equal(set.except(element)
                     .cardinality.successor);
         };
};
```

Это определение мощности не является достаточным, поскольку оно не применимо к несчетным бесконечным множествам (значения типов REAL, PQ и т. д.) и не закончено для бесконечных множеств. Кроме того, определение целочисленного типа данных, приведенное в настоящей спецификации, является неполным для таких случаев, поскольку не учитывает бесконечности. Наконец, значение мощности служит примером, когда надо различать мощность $\aleph_0$ (алеф-0) счетных бесконечных множеств (например, значений типа INT) от мощности $\aleph_1$ (алеф-1) несчетных множеств (например, значений типов REAL, PQ).

B.3.1.6 Свойство union: SET<T>

Определение: свойство, описывающее объединение двух множеств (компонентов множества), при котором каждый элемент объединения является также элементом как минимум одного компонента объединения.

```
invariant(SET<T> x, y, z)
      where x.nonNull.and(y.nonNull).and(z.nonNull) {
   x.union(y).equal(z).equal(forall(T e) {
      z.contains(e).equal(x.contains(e).or(y.contains(e)));
      });
};
```

B.3.1.7 Свойство union: SET<T>

Определение: свойство, описывающее добавление элемента к множеству.

```
invariant(SET<T> set, singletonset, T element)
      where set.nonNull.and(element.nonNull)
               .and(singletonset.cardinality.isOne)
               .and(singletonset.contains(element)) {
   set.union(element).equal(set.union(singleton));
};
```

B.3.1.8 Свойство except: SET<T>

Определение: свойство, описывающее разность между данным множеством и вычитаемым, содержащую все элементы данного множества, которые не являются элементами вычитаемого множества.

```
invariant(SET<T> x, y, z)
      where x.nonNull.and(y.nonNull).and(z.nonNull) {
   x.except(y).equal(z).equal(forall(T e) {
      z.contains(e).equal(x.contains(e).and(y.contains(e).not));
      });
};
```

B.3.1.9 Свойство except: SET<T>

Определение: свойство, описывающее удаление элемента из данного множества и представляющее собой множество, состоящее из всех элементов данного множества, кроме удаляемого. Если удаляемый элемент не входит в данное множество, то результирующим множеством является данное множество.

```
invariant(SET<T> x, z; T d)
      where z.nonNull.and(z.nonNull).and(d.nonNull) {
   x.except(d).equal(z).equal(forall(T e) {
      z.contains(e).equal(x.contains(e).and(d.equal(e).not));
      });
};
```

B.3.1.10 Свойство intersection: SET<T>

Определение: свойство, описывающее пересечение двух множеств, содержащее те и только те элементы, которые входят в каждое из этих множеств.

```
invariant(SET<T> x, y, z)
      where x.nonNull.and(y.nonNull).and(z.nonNull) {
   x.intersection(y).equal(z).equal(forall(T e) {
      z.contains(e).equal(x.contains(e).and(y.contains(e)));
      });
};
```

B.3.1.11 Литеральная форма

Если элементы типа T имеют литеральную форму, то множество элементов типа T имеет литеральную форму, в которой элементы множества перечислены внутри фигурных скобок, отделяемые друг от друга точками с запятой.

```
SET<T>.literal ST {
   SET<T>          : "{" elements "}"   { $.equal($2); };
```

```
   SET<T> elements : elements ";" T      | $.except($2).equal($1); |
                   | T                   | $.contains($1);
                                           $.except($1).isEmpty; };
};
```

Примечание — Эта литеральная форма практична только для относительно малых перечисляемых множеств, но это не означает, что все множества являются относительно малым перечислением элементов.

Таблица В.38 — Примеры литералов множеств

| Литерал | Значение |
|---|---|
| {1; 3; 5; 7; 19} | Множество целых или вещественных чисел |
| {3; 1; 5; 19; 7} | То же самое множество целых или вещественных чисел |
| {1.2 м; 2.67 м; 17.8 м} | Дискретное множество физических величин |
| {яблоко; апельсин; банан} | Множество строк символов |

Примечание — Если спецификация реализуемой технологии, основанной на символьном представлении данных, имеет более естественную форму литералов таких коллекций, то она должна использовать ее для литералов множеств.

В.3.1.12 Преобразование значения элемента во множество: SET<T>

Значение типа T может быть преобразовано в тривиальное множество, содержащее это значение как единственный элемент.

```
invariant(T x) {
   ((SET<T>)x).contains(x);
   ((SET<T>)x).except(x).isEmpty;
};
```

В.3.1.13 Выпуклая оболочка полностью упорядоченных множеств: IVL<T>

Множества величин могут быть полностью упорядоченными, если между любыми двумя элементами множества определено отношение порядка. Следует учесть, что понятие «упорядоченного множества» не тождественно списку (тип данных LIST). Например, множество {3; 2; 4; 88; 1} является упорядоченным. Позиции элементов множества не имеют значения, но можно провести сравнение элементов и записать их в порядке возрастания (1; 2; 4; 88).

Для полностью упорядоченных множеств можно построить выпуклую оболочку. Выпуклой оболочкой полностью упорядоченного множества S является наименьший интервал, содержащий множество S. Важность этого понятия будет обсуждаться позже.

```
type Set<QTY> alias SET<QTY> {
   BL           totallyOrdered;
   IVL<T>  hull;
};

invariant(SET<QTY> s)
      where s.nonNull {
   s.totallyOrdered.equal(forall(QTY x, y)
         where s.contains(x).and(s.contains(y)) {
      x.compares(y); });
};

invariant(SET<QTY> s)
      where s.totallyOrdered {
   s.hull.contains(s);
   forall(T e)
         where s.contains(e) {
      s.hull.low.lessOrEqual(e);
      e.lessOrEqual(s.hull.high);
      };
};
```

Следует обратить внимание, что выпуклая оболочка определена в том и только том случае, если действительное множество является полностью упорядоченным. Значения типа данных его элементов не обязаны быть полностью упорядоченным. Например, значения типа данных PQ упорядочены только частично (поскольку сравнивать можно только величины одного типа), но множество типа SET<PQ> тем не менее может быть полностью упорядоченным (если содержит только сравнимые величины). Например, выпуклой оболочкой множества {4 с; 20 с; 55 с} является интервал [4 с; 55 с]; выпуклая оболочка множества {"яблоко"; "апельсин"; "банан"} не определена, поскольку это множество не является упорядоченным; выпуклая оболочка множества {2 м; 4 м; 8 с} также не определена, поскольку и оно не является полностью упорядоченным (секунды не сравнимы с метрами).

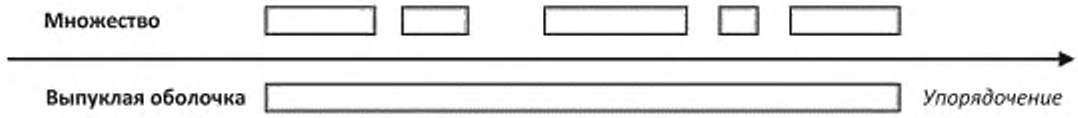


Рисунок В.12 — Выпуклая оболочка полностью упорядоченного множества

**В.3.2 Тип данных списка LIST (специализация типа данных ANY)**

Определение: значение, содержащее другие дискретные (но не обязательно различные) значения в определенной последовательности.

```
template<ANY T>
type Sequence<T> alias LIST<T> specializes ANY {
             T        head;
             LIST<T>  tail;
             BL       isEmpty;
             BL       notEmpty;
             T        item(INT index);
             BL       contains(T item);
             INT      length;
  literal    ST;
  promotion  LIST<T>  (T x);
};
```

Список может не содержать ни одного элемента.

В.3.2.1 Свойство head: T

Определение: первый элемент списка. Свойство head является определяющим для семантики списка.

В.3.2.2 Свойство tail: LIST<T>

Определение: список элементов, следующих за первым элементом списка. Свойство tail является определяющим для семантики списка.

В.3.2.3 Свойство isEmpty: BL

Определение: предикат, имеющий значение «true», если список пуст, то есть не содержит элементов.

Обратите внимание на отличие пустого списка от пустого значения: пустой список является допустимым списком, а не исключительным значением.

```
invariant(LIST<T> x)
      where x.isEmpty {
   x.head.isNull;
   x.tail.isNull;
};
```

Следует обратить внимание, что пустота элемента head и списка tail является необходимым, но не достаточным условием пустоты списка, поскольку список может содержать в качестве элементов пустые значения, это условие может означать, что данный список имеет единственный элемент, имеющий пустое значение.

В.3.2.4 Свойство notEmpty: BL

Определение: предикат, имеющий значение «true», если список не является пустым. Является отрицанием свойства isEmpty.

```
invariant(LIST<T> x)
      where x.nonNull {
   x.notEmpty.equal(x.isEmpty.not);
};
```

B.3.2.5 Свойство item: T

Определение: значением этого свойства является элемент, находящийся в заданной позиции списка (имеющий заданный индекс). Нулевой индекс задает первый элемент списка (значение свойства head).

```
invariant(LIST<T> list; INT index)
      where list.nonNull.and(index.nonNegative) {
   list.isEmpty.implies(list.item(index).isNull);
   list.notEmpty.and(index.isZero)
       .implies(list.item(index).equal(list.head));
   list.notEmpty.and(index.nonZero)
       .implies(list.item(index).equal(
          list.tail.item(index.predecessor)));
};
```

B.3.2.6 Свойство contains: BL

Определение: предикат, имеющий значение «true», если данный список содержит заданное значение.

```
invariant(LIST<T> list; T item)
      where list.nonNull {
   list.isEmpty.implies(list.contains(item).not);
   list.nonEmpty.and(item.nonNull).implies(list.contains(item).equal(
      list.head.equal(item).or(list.tail.contains(item))));
      list.notEmpty.and(item.isNull).implies(list.contains(item).equal(
         list.head.isNull.or(list.tail.contains(item)));
};
```

B.3.2.7 Свойство length: INT

Определение: число элементов списка. Пустые элементы включаются в подсчет наравне с обычными элементами списка.

```
invariant(LIST<T> list)
      where x.nonNull {
   list.isEmpty.equal(list.length.isZero);
   list.notEmpty.equal(list.length.equal(
      list.tail.length.successor));
};
```

B.3.2.8 Свойство equal: BL (унаследовано от типа данных ANY)

Два списка равны в том и только том случае, если либо они оба пусты, либо у них одинаковы свойства head и свойства tail.

```
invariant(LIST<T> x, y)
      where x.nonNull.and(y.nonNull) {
   x.isEmpty.and(y.isEmpty).implies(x.equal(y));
   x.notEmpty.and(y.notEmpty).and(x.head.nonNull)
      .implies(x.equal(y).equal(
         x.head.equal(y.head).and(x.tail.equal(y.tail))));
   x.notEmpty.and(y.notEmpty).and(x.head.isNull)
             .implies(x.equal(y).equal(
                y.head.isNull.and(x.tail.equal(y.tail))));
};
```

B.3.2.9 Литеральная форма

Если элементы типа T имеют литеральную форму, то список элементов типа T имеет литеральную форму, в которой элементы списка пронумерованы и перечисляются внутри круглых скобок, разделяемые точками с запятой.

```
LIST<T>.literal ST {
   LIST<T> : "(" elements ")"        { $.equal($2); }
           | "(" ")"                 { $.isEmpty; };
   LIST<T> elements
```

```
        : T ";" elements          { $.head.equal($1);
                                    $.tail.equal($3); }
        | T                       { $.head.equal($1);
                                    $.tail.isEmpty; };
};
```

Таблица В.39 — Примеры литералов списков

| Литерал | Значение |
|---|---|
| (1; 3; 5; 7; 19) | Список целых или вещественных чисел |
| (3; 1; 5; 19; 7) | Другой список целых или вещественных чисел |
| (1.2 м; 2.67 м; 17.8 м) | Дискретный список физических величин |
| («яблоко»; «апельсин»; «банан») | Список строк символов |

Примечание — Если спецификация реализуемой технологии, основанной на символьном представлении данных, имеет более естественную форму литералов таких коллекций, то она должна использовать ее для литералов списков.

В.3.2.10 Преобразование значения элемента в список: LIST<T>

Значение типа T может быть преобразовано в тривиальный список, содержащий это значение как единственный элемент.

```
invariant(T x) {
   ((LIST<T>)x).head.equal(x);
   ((LIST<T>)x).tail.isEmpty;
};
```

**В.3.3 Тип данных GeneratedSequence (GLIST) (специализация типа данных LIST)**

Определение: периодическая или монотонная последовательность значений, генерируемая с помощью небольшого числа параметров вместо перечисления. Используется для задания регулярной последовательности считывания биосигналов.

```
type GeneratedSequence<QTY T> alias GLIST specializes LIST<T> {
   T    head;
   QTY  increment;
   INT  period;
   INT  denominator;
};
```

Таблица В.40 — Сводка свойств типа данных GeneratedSequence

| Имя | Тип | Описание |
|---|---|---|
| head | T | Первый элемент данной последовательности. Свойство head является определяющим для семантики последовательности |
| increment | QTY | Разность между значением и предшествующим ему отличающимся значением. Например, при генерации последовательности (1; 4; 7; 10; 13; ...) свойство increment равно 3; аналогично, при генерации последовательности (1; 1; 4; 4; 7; 7; 10; 10; 13; 13; ...) свойство increment также равно 3 |
| period | INT | Если это свойство не пусто, то оно определяет чередование последовательности, то есть через заданное в нем число различных элементов значение элемента последовательности возвращается к начальному значению. Например, последовательность (1; 2; 3; 1; 2; 3; 1; 2; 3; ...) имеет период 3; последовательность (1; 1; 2; 2; 3; 3; 1; 1; 2; 2; 3; 3; ...) также имеет период 3 |
| denominator | INT | Целов число, на которое делится индекс элемента последовательности. Задает число повторений того же самого значения элемента последовательности перед тем, как перейти к следующему значению. Например, при генерации последовательности (1; 1; 1; 2; 2; 2; 3; 3; 3; ...) свойство denominator равно 3 |

Вычисление элемента последовательности с заданным индексом $i$ ($x_i$) осуществляется следующим образом: индекс $i$ делится нацело на значение свойства denominator ($d$) и определяется остаток от деления результата на значение свойства period ($p$). Полученное значение умножается на значение свойства increment ($\Delta x$) и прибавляется к значению свойства head ($x_0$):

$$x_i = x_0 + \Delta x \cdot (i/d) \bmod p$$

```
invariant(GLIST<T> list, INT index)
      where list.nonNull.and(index.nonNull) {
   list.period.nonNull.implies(list.item(index).equal(
      list.head.plus(item.dividedBy(list.increment.denominator)
         .remainder(list.period)).times(increment)));
   list.period.isNull.implies(list.item(index).equal(
      list.head.plus(item.dividedBy(list.increment.denominator))
         .times(increment)));
);
```

B.3.3.1 Свойство head: T (унаследовано от типа данных LIST)

Первый элемент генерируемой последовательности.

B.3.3.2 Свойство increment: QTY

Определение: разность между значением и предшествующим ему отличающимся значением. Например, при генерации последовательности (1; 4; 7; 10; 13; ...) свойство increment равно 3; аналогично при генерации последовательности (1; 1; 4; 4; 7; 7; 10; 10; 13; 13; ...) свойство increment также равно 3.

```
invariant(GLIST<T> x) {
   x.increment.dataType.implies(T.diffType);
};
```

B.3.3.3 Свойство period: INT

Определение: если это свойство не пусто, то оно определяет чередование последовательности, то есть через заданное в нем число различных элементов значение элемента последовательности возвращается к начальному значению. Например, последовательность (1; 2; 3; 1; 2; 3; 1; 2; 3; ...) имеет период 3; последовательность (1; 1; 2; 2; 3; 3; 1; 1; 2; 2; 3; 3; ...) также имеет период 3.

Задание свойства period позволяет периодически повторять группу значений. «Сигнал» такого периодического генератора всегда является «пилой» наподобие изображения линейной функции на осциллографе[1)].

B.3.3.4 Свойство denominator: INT

Определение: целое число, на которое делится индекс элемента последовательности. Задает число повторений того же самого значения элемента последовательности перед тем, как перейти к следующему значению. Например, при генерации последовательности (1; 1; 1; 2; 2; 2; 3; 3; 3; ...) свойство denominator равно 3.

Свойство denominator используется для генерирования нескольких последовательностей, используемых для периодического сканирования многомерного пространства. Например, (абстрактный) экран телевизора использует два таких генератора для строк и столбцов изображения. Скажем, если на экране отображается развертка 200 строк с 320 растровыми столбцами, то у генератора столбцов свойство denominator равно 1, а у генератора строк это свойство равно 320.

Таблица B.41 — Примеры генерируемых последовательностей

| Свойство head | Свойство increment | Свойство denominator | Свойство period | Описание последовательности |
|---|---|---|---|---|
| 0 | 1 | 1 | ∞ | Последовательность идентификаторов, в которой каждый элемент равен своему индексу |
| 198706051900 | 2 hour | 1 | ∞ | Последовательность значений времени, начинающаяся с 19:00 5 июня 1987 г. и каждый раз возрастающая на два часа: 21:00, 23:00, 1:00 (6 июня), 3:00, 5:00 и т. д. |

[1)] Обратите внимание на отличие типа данных GTS. Тип данных GTS представляет собой генератор для типа SET<TS>, а не LIST<TS>. Последовательность дискретных значений из непрерывного домена не имеет особого смысла, кроме как для приложений оцифровки биосигналов. Однако тип данных SET<TS> можно представлять себе как последовательность значений интервалов IVL<TS>, которая тем не менее отличается от типа данных LIST<TS>.

*Окончание таблицы В.41*

| Свойство head | Свойство increment | Свойство denominator | Свойство period | Описание последовательности |
|---|---|---|---|---|
| 0 V | 1 mV | 1 | 100 | Считывание линейно возрастающего напряжения на входе цифрового осциллографа в пределах от 0 до 100 мВ за 100 шагов по 1 мВ. Частота опроса по этим данным не определяется, поскольку интервал времени между последовательными считываниями неизвестен |
| 2002072920300 | 100 us | 1 | ∞ | Временной ряд, значения которого начинаются в 20:30 29 июня 2002 г. и возрастающие на 100 мкс на каждом шаге. Если скомбинировать его с предыдущим генератором в качестве моментов считывания сигнала, то напряжение будет меняться на 1 мВ каждые 100 мкс. Поскольку период составляет 100 шагов, то его длительность равна 10 мс, что эквивалентно частоте сигнала 100 Гц |
| 0 В | 1 mV | 100 | 100 | Сочетание этого генератора с двумя предыдущими может описывать трехмерное пространство считываний с двумя осями напряжения и осью времени. Каждое приращение напряжения данного генератора также составляет 100 мв с периодичностью 100 шагов, однако изменение напряжения происходит только через 100 считываний, поэтому первый генератор напряжения успевает сделать полный цикл за одно приращение напряжения в данном генераторе. Эти два генератора напряжения можно представить себе как «строки» и «столбцы» считываемого «кадра». Если в качестве таймера используется предыдущий генератор, то частота кадров 100 × 100 мВ составит 1 Гц |

**В.3.4 Тип данных SampledSequence (SLIST) (специализация типа данных LIST)**

Определение: последовательность считываемых величин, получаемая из списка целых значений с помощью масштабирования и сдвига. Используется для описания считываемых биосигналов.

```
type SampledSequence<QTY T> alias SLIST specializes LIST<T> {
   T          origin;
   QTY        scale;
   LIST<INT>  digits;
};
```

Таблица В.42 — Сводка свойств типа данных SampledSequence

| Имя | Тип | Описание |
|---|---|---|
| origin | T | Начало шкалы значений элементов списка, то есть значение величины, которой соответствует нулевое значение элемента списка |
| scale | QTY | Величина, определяющая масштаб значений, указанных в списке |
| digits | LIST<INT> | Последовательность целых значений считываний. Обычно это исходные значения, получаемые от аналого-цифрового преобразователя |

Значение физической величины, соответствующее определенному индексу ($i$) последовательности, вычисляется с помощью умножения значения элемента списка digits ($d_i$) с этим же индексом на масштаб scale ($s$) и прибавления к результату значения начала шкалы origin ($x_0$):

$$x_i = x_0 + s \cdot d_i$$

```
invariant(SLIST<T> list, INT index)
      where list.nonNull.and(index.nonNegative) {
```

```
   list.item(index).equal(
      list.scale.times(digits.item(index))
         .plus(list.origin));
};
```

B.3.4.1 Свойство origin: T

Определение: начало шкалы значений элементов списка, то есть значение величины, которой соответствует нулевое значение элемента списка.

B.3.4.2 Свойство scale: QTY

Определение: величина масштаба, на которую умножаются значения элементов списка.

```
invariant(SLIST<T> x) {
   x.scale.dataType.implies(T.diffType);
};
```

B.3.4.3 Свойство digits: LIST<INT>

Определение: последовательность целых значений считываний. Обычно это исходные значения, получаемые от аналого-цифрового преобразователя.

**B.3.5 Тип данных BAG (специализация типа данных ANY)**

Определение: неупорядоченная коллекция значений, в которой каждое значение может присутствовать более одного раза (мультимножество).

```
template<ANY T>
type Bag<T> alias BAG<T> specializes ANY {
               INT      contains(T kind);
               BL       isEmpty;
               BL       notEmpty;
               BAG<T>   plus(BAG<T> x);
               BAG<T>   minus(BAG<T> x);
   promotion   BAG<T>   (T x);
};
```

Примечание — значение типа BAG может быть представлено двумя способами: как перечисление элементов, включая повторяющиеся, или как «сжатое мультимножество», содержание которого представляется парами «значение элемента» — «число значений». Гистограмма, показывающая абсолютные количества, может служить примером сжатой формы представления значения типа BAG. Поэтому тип данных BAG полезен для передачи необработанных статистических данных.

B.3.5.1 Свойство contains: INT

Определение: число элементов с данным значением, присутствующих в данном мультимножестве.

Свойство contains является примитивным семантическим свойством типа данных BAG, исходя из которого определяются все остальные свойства.

```
invariant(BAG<T> bag; T item)
      where bag.nonNull.and(item.nonNull) {
   bag.contains(item).nonNegative;
   bag.isEmpty.equal(bag.contains(item).isZero);
};
```

B.3.5.2 Свойство notEmpty: BL

Определение: предикат, указывающий, что данное мультимножество содержит элементы.

```
invariant(BAG<T> bag)
      where bag.nonNull {
   bag.notEmpty.equal(exists(T item) {
      bag.contains(item);
      });
};
```

B.3.5.3 Свойство isEmpty: BL

Определение: предикат, указывающий, что данное мультимножество не содержит элементов (отрицание предиката notEmpty). Пустое мультимножество является допустимым, а не исключительным значением.

```
invariant(BAG<T> bag)
      where bag.nonNull {
   bag.isEmpty.equal(notEmpty.not);
};
```

B.3.5.4 Свойство plus: BAG<T>

Определение: мультимножество, содержащее все элементы операндов-мультимножеств, то есть количество элементов с каждым значением увеличивается.

```
invariant(BAG<T> x, y, z)
      where x.nonNull.and(y.nonNull) {
   x.plus(y).equal(z).equal(
      forall(T e)
            where e.nonNull {
         z.contains(e).equal(x.contains(e)
                     .plus(y.contains(e)));
         });
};
```

B.3.5.5 Свойство minus: BAG<T>

Определение: мультимножество, содержащее все элементы данного мультимножества (уменьшаемого) за вычетом элементов другого мультимножества BAG (вычитаемого). Мультимножества не могут иметь дефицит. Если вычитаемое содержит больше экземпляров одного и того же значения, то разность содержит нуль экземпляров этого значения.

```
invariant(BAG<T> x, y, z)
      where x.nonNull.and(y.nonNull) {
   x.minus(y).equal(z).equal(
      forall(T e)
            where e.nonNull {
               exists(INT n)
                  where n.equal(x.contains(e).minus(y.contains(e))) {
         n.nonNegative.equal(z.contains(e));
         n.isNegative.equal(z.contains(e).isZero);
         };
      });
};
```

B.3.5.6 Преобразование значения элемента в мультимножество: BAG<T>

Определение: значение типа T может быть преобразовано в тривиальное мультимножество, содержащее это значение как единственный элемент.

```
invariant(T x) {
   ((BAG<T>)x).contains(x).equal(1);
   forall(T y) {
      ((BAG<T>)x).contains(y)
                 .implies(x.equal(y)) };
};
```

**B.3.6 Тип данных IVL (специализация типа данных SET)**

Определение: множество последовательных значений упорядоченного базового типа данных.

Любой упорядоченный тип данных может быть базовым для типа данных IVL; является ли он дискретным или непрерывным, значения не имеет. Если базовый тип данных упорядочен только частично, то все элементы типа IVL должны принадлежать полностью упорядоченному подмножеству этого типа данных.

Например, тип данных PQ (физическая величина) считается упорядоченным. Однако он упорядочен только частично; полная упорядоченность определена только для сопоставимых физических величин (имеющих одну и ту же размерность). В то время как между 2 и 4 метрами существуют интервалы, нельзя задать интервал между 2 метрами и 4 секундами.

Интервалы являются множествами (тип данных SET) и обладают всеми свойствами множеств. Однако объединение и разность интервалов может не быть интервалом, поскольку значения элементов, составляющих результат этих операций, могут не быть последовательными. Пересечения интервалов всегда являются интервалами.

```
template<QTY T>
type Interval<T> alias IVL<T> specializes SET<T> {
                T       low;
                BL      lowClosed;
                T       high;
                BL      highClosed;
                QTY     width;
                T       center;
                IVL<T>  hull;
                IVL<T>  hull(IVL<T> x);
   literal      ST;
   promotion    IVL<T>  (T x);
   demotion     T;
};
```

B.3.6.1 Свойство low: T

Определение: нижняя граница интервала.

```
invariant(IVL<T> x; T e)
      where x.nonNull.and(x.contains(e)) {
   x.low.lessOrEqual(e);
};
```

B.3.6.2 Свойство high: T

Определение: верхняя граница интервала.

```
invariant(IVL<T> x; T e)
      where x.nonNull.and(x.contains(e)) {
   e.lessOrEqual(x.high);
};
```

B.3.6.3 Свойство width: QTY

Определение: ширина интервала, то есть разность граничных значений high и low. Свойство width выделяется в целях симметричной обработки всех случаев, когда интервал задан не полностью. В любом представлении значения типа IVL достаточно указать только два из трех свойств high, low и width, а третье из них выводится.

Если известны две границы интервала, то ширина интервала width может быть вычислена как разность между значениями свойств high и low. Если известна одна граница и ширина width, то известна и другая граница. Если ни одна граница не известна, то значение свойства width тем не менее может быть известно. Например, известно, что некоторая деятельность занимает около 30 минут, но не известно, когда она начинается.

Следует учесть, что тип данных свойства width не всегда совпадает с типом данных границ интервала. Для величин, измеряемых по относительной шкале (типы данных REAL, PQ, MO), он совпадает. Для величин, измеряемых по разностной шкале (например, тип данных TS), свойство width имеет тип данных разности (например, тип данных PQ с размерностью времени в случае интервала с границами типа TS). Для дискретных элементов (тип данных INT) ширина width определяется как число элементов в интервале, деленное на 2. Поэтому она может иметь тип данных REAL.

```
invariant(IVL<T> x) {
   x.low.lessOrEqual(x.high);
   x.width.equal(x.high.minus(x.low));
};

invariant(IVL<T> x) {
   x.width.dataType.implies(T.diffType);
};
```

B.3.6.4 Свойство center: T

Определение: это свойство задает центр интервала, то есть арифметическое среднее его границ (сумма значений свойств low и high, деленная на 2). Свойство center выделяется как семантическое для преобразования интервалов в значения точек и обратно.

Следует учесть, что свойство center существует не у каждого интервала. Например, интервалы, не ограниченные с одной стороны, не имеют центра. То же относится и к интервалам дискретных значений, содержащим

четное число элементов. Если одна или обе границы интервала неизвестны, свойство center тем не менее может быть задано. В действительности это свойство в основном предназначено для использования, когда границы интервала неизвестны.

```
invariant(IVL<T> x)
      where x.low.nonNull.and(x.high.nonNull) {
   x.center.equal(x.low.plus(x.width.times(0.5)))};
};

invariant(IVL<T> x)
      where x.low.isNull.or(x.high.isNull) {
   x.center.notApplicable;
};
```

B.3.6.5 Свойство lowClosed: BL

Определение: указывает, включается ли нижняя граница low в значение типа IVL (закрытая граница) или исключается из него (открытая граница).

```
invariant(IVL<T> x)
      where x.nonNull {
   x.low.nonNull.implies(x.lowClosed.equal(x.contains(x.low)));
   x.low.isNull.implies(x.lowClosed.not);
};
```

B.3.6.6 Свойство highClosed: BL

Определение: указывает, включается ли верхняя граница high в значение типа IVL (закрытая граница) или исключается из него (открытая граница).

```
invariant(IVL<T> x)
      where x.nonNull {
   x.high.nonNull.implies(x.highClosed.equal(x.contains(x.high)));
   x.high.isNull.implies(x.highClosed.not);
};
```

B.3.6.7 Литеральная форма

Литеральные формы типа данных IVL определены таким образом, чтобы они по возможности были интуитивно очевидными для человека. Определены следующие пять различных форм[1]:

1. Форма интервала с квадратными скобками, например, «[3.5; 5.5[»;
2. Форма с дефисом, например, «3.5-5.5»;
3. Форма с оператором сравнения, используя символы операторов отношения, например, «<5.5»;
4. Форма с центром и шириной, например, «4.5[2.0[»;
5. Форма только с шириной и квадратными скобками, например, «[2.0[».

```
IVL<T>.literal ST {
   IVL<T> range : interval              { $.equal($1); }
                | dash                  { $.equal($1); }
                | comparator            { $.equal($1); }
                | center_width          { $.equal($1); }
                | width                 { $.equal($1); };
```

[1] Наличие столь большого числа вариантов заслуживает разъяснений. В принципе, достаточно было бы иметь форму интервала и форму только с шириной. Однако форма интервала воспринимается чужеродной во многих приложениях медицинской информатики. Одной из важных целей литеральных форм является искоренение несоответствия за счет облегчения соответствия, не входящего в противоречие со значением понятий.

Далее, различные литеральные формы имеют свою силу и свою слабость. Сила формы интервала и формы с центром и шириной в том, что они более точны и позволяют указать открытость и закрытость границ. Но их слабость в том, что для обозначения бесконечных границ требуются специальные символы, что не требуется для формы с оператором сравнения. Форма с центром и шириной вообще не позволяет задавать интервалы с бесконечными границами. Но форма с оператором сравнения может представлять только интервалы с одной границей (то есть когда другая граница бесконечна или неизвестна). Форма с дефисом слабее всех, но при этом наиболее интуитивна для интервалов с двумя границами.

```
    IVL<T> interval
                : open T ";" T close;       { $.low.equal($2);
                                              $.high.equal($4);
                                              $.lowClosed.equal($1);
                                              $.highClosed.equal($5); };
    BL open     : "["                       { $.equal(true); }
                | "]"                       { $.equal(false); };
    BL close    : "]"                       { $.equal(true); }
                | "["                       { $.equal(false); };
    IVL<T> width
                : open T.diffType close     { $.width.equal($2);
                                              $.lowClosed.equal($1);
                                              $.highClosed.equal($3); };
    IVL<T> center_width
                : T width                   { $.center.equal($1);
                                              $.width.equal($2.width);
                                              $.lowClosed.equal($2.lowClosed);
                                              $.highClosed.equal($2.highClosed); };
    IVL<T> dash : T "-" T;                  { $.low.equal($2);
                                              $.high.equal($4);
                                              $.lowClosed.equal(true);
                                              $.highClosed.equal(true); };
    IVL<TS> comparator
                : "<"  T                    { $.high.equal(T);
                                              $.high.closed(false);
                                              $.low.negativelyInfinite; }
                | ">"  T                    { $.low.equal(T);
                                              $.low.closed(false);
                                              $.high.positivelyInfinite; }
                | "<=" T                    { $.high.equal(T);
                                              $.high.closed(true);
                                              $.low.negativelyInfinite; }
                | ">=" T                    { $.low.equal(T);
                                              $.low.closed(true);
                                              $.high.positivelyInfinite; };
};
```

B.3.6.8 Преобразование значений элементов в интервалы: IVL<T>

Величина типа T может быть преобразована в тривиальный интервал типа IVL, у которого свойства low и high равны и границы закрыты.

```
invariant(T x) {
   ((IVL<T>)x).low.equal(x);
   ((IVL<T>)x).high.equal(x);
   ((IVL<T>)x).highClosed;
   ((IVL<T>)x).lowClosed;
};
```

B.3.6.9 Приведение интервала к представительному значению элемента типа T

Значение типа IVL может быть приведено к одной величине типа T, представляющей весь интервал. Если обе границы финитны, этой величиной является значение свойства center. Если одна из границ бесконечна, этой величиной является другая граница. Если обе границы бесконечны, приведение к одной точке не применимо.

```
invariant(IVL<T> x)
      where x.nonNull {
   x.low.nonNull.and(x.high.nonNull).implies(((T)x).equal(x.center));
   x.high.nonNull.and(x.low.isNull).implies(((T)x).equal(x.high));
   x.low.nonNull.and(x.high.isNull).implies(((T)x).equal(x.low));
   x.low.isNull.and(x.high.isNull).implies(((T)x).notApplicable);
};
```

B.3.6.10 Свойство hull: IVL<T> (унаследовано от типа данных SET)

Определение: это свойство задает выпуклую, или «интервальную» оболочку двух значений типа IVL, представляющую наименьший интервал, содержащий оба операнда.

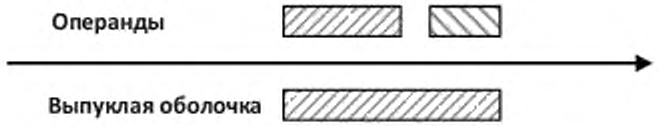


Рисунок B.13 — Выпуклая оболочка двух интервалов

```
invariant(IVL<T> h, IVL<T> i, j)
      where h.equal(i.hull(j)) {
   i.low.lessOrEqual(j.low).implies(h.low.equal(i.low));
   j.low.lessOrEqual(i.low).implies(h.low.equal(j.low));
   i.high.lessOrEqual(j.high).implies(h.high.equal(j.high));
   j.high.lessOrEqual(i.high).implies(h.high.equal(i.high));
};
```

**B.3.7 Тип данных физических величин IVL<PQ> (специализация типа данных IVL)**

Определение: множество последовательных значений физических величин.

Тип данных интервала физических величин строится по общему типу интервала. Но с учетом того, что единицы измерения могут быть взяты от границ интервала, в определение типа данных IVL<PQ> добавлены новая семантика и отдельная литеральная форма. Дополнительным представлением интервала физических величин является интервал вещественных чисел с одной единицей измерения.

```
type Interval<PQ> alias IVL<PQ> {
   IVL<REAL> value;
   CS  unit;
};
```

Единица измерения применяется как к нижней, так и верхней границе интервала.

```
invariant(IVL<PQ> x)
      where x.nonNull {
   x.value.nonNull;
   x.low.value.equal(x.value.low);
   x.low.unit.equal(x.unit);
   x.lowClosed.equal(x.value.lowClosed);
   x.high.value.equal(x.value.high);
   x.high.unit.equal(x.unit);
   x.highClosed.equal(x.value.highClosed);
};
```

Специальная литеральная форма представляет собой просто интервал вещественных чисел, пробел и единицу измерения.

```
IVL<PQ>.literal ST {
   IVL<PQ>  : IVL<REAL> " " unit     { $.value($1);
                                       $.unit.equal($3); }
            | IVL<REAL>              { $.equal($1); };
   CS unit  : ST                     { $.value.equal($1);
                                       $.codeSystem(2.16.840.1.113883.3.2); };
};
```

Например, строки «[0;5] mmol/L» или «<20 mg/dL» являются допустимыми литеральными формами интервалов физических величин. Допускается также общая литеральная форма, например, «[50 nm; 2 m]».

**B.3.8 Тип данных интервала времени IVL<TS> (специализация типа данных IVL)**

Определение: множество последовательных значений штампов даты и времени.

Общий тип данных интервала позволяет также определить интервалы моментов времени. Однако для этих интервалов имеются определенные особенности литеральных представлений и преобразования, описанные в настоящем подразделе.

```
type Interval<TS> alias IVL<TS> {
   literal    ST;
   promotion  IVL<TS> (TS x);
};
```

B.3.8.1 Преобразование моментов времени в интервалы: IVL<TS> (унаследовано от типа данных IVL)

Значение типа TS может быть преобразовано в интервал типа IVL<TS>, у которого нижняя граница представляет собой само это значение, а свойство ширины width выводится из точности значения типа TS и длительности наименее значащего календарного периода, указанного в этом значении. Верхняя граница интервала открыта. Например, литерал «200009» значения типа TS преобразуется в интервал типа IVL<TS> с нижней границей 200009 и шириной 30 дней, что эквивалентно интервалу «[200009;200010[».

B.3.8.2 Литеральная форма

Литеральная форма интервала моментов времени является исключительной:

- форма с дефисом не разрешена для интервалов моментов времени;
- вместо нее используется «форма оболочки».

Чтобы избежать синтаксических конфликтов с часовой зоной и несколько отличающихся профилей, предусмотренных в стандарте ИСО 8601 и используемых в некоторых платформах реализуемых технологий, форма с дефисом для значений типа IVL<TS> не разрешена. Предпочтительной является форма интервала с квадратными скобками.

***Пример — Литералом интервала от 20:00 до 21:40 12 мая 1987 г. служит строка «[198705122000;198705122130[».***

Примечание — Точность границы интервала несущественна для самого интервала. Можно было бы ошибочно предположить, что интервал «[19870901;19870930]» охватывает весь сентябрь 1987 г. вплоть до завершения суток 30 сентября. Однако это не так. Правильный способ задания полного календарного цикла (например, час, день, месяц, год и т. д.) состоит в использовании нотации с открытой верхней границей. Например, весь сентябрь 1987 г. может быть задан литералом «[198709;198710[»[1].

«Форма оболочки» литерала определяется как выпуклая оболочка (см. B.3.6.10 «Свойство hull: IVL<T> (унаследовано от типа данных SET)») интервалов, полученных в результате преобразования двух значений моментов времени в интервалы.

```
   IVL<TS> hull : TS ".." TS      { $.equal(((IVL<TS>)$1)
                                     .hull((IVL<TS>)$3));
};
```

Например, строка «19870901..19870930» является допустимым литералом в «форме оболочки». Она эквивалентна форме интервала «[19870901;19871001[»[2].

Форма оболочки позволяет сокращение, при котором в литерале верхней границы не требуется повторять левые цифры, одинаковые с литералом нижней границы. Два литерала моментов времени выравниваются вправо и недостающие левые цифры копируются из литерала нижней границы в литерал верхней. Эта простая строковая операция здесь формально не определяется.

Например, интервал с 12 мая 1987 по 23 мая 1987 г. может быть записан как «19870512..23». Однако интервал с 12 мая 1987 г. по 2 июня 1987 г. должен быть записан как «19870512..0602», а не «20000512..02».

**B.4 Параметризованные расширения типов данных**

Параметризованные расширения типов данных представляют собой параметризованные типы данных, у которых в качестве параметра используется один тип данных и которые расширяют (специализируют) этот параметр.

---

[1] Может возникнуть впечатление, что это утверждение противоречит правилу преобразования значения типа TS в интервал типа IVL<TS>. Однако никакого противоречия нет. Точность границы значения не имеет, но точность отдельного момента времени (в отличие от границы интервала) существенна, когда момент времени преобразуется в интервал.

[2] Форма оболочки может показаться излишней для представления простого интервала. Однако она становится важной для нотации периодических интервалов, поскольку сокращает запись и (хотя это и может быть спорным) делает запись более сложных структур периодов времени более интуитивной.

На формальном языке определения типов данных параметризованные расширения типов данных могут быть описаны следующим шаблоном: «**template**<ANY T> **type***GenericTypeExtensionName***specializes** T { ... };». Эти расширения наследуют большинство свойств своего базового типа данных и добавляют к ним некоторые специфичные свойства. Поскольку параметризованное расширение типа данных является специализацией своего базового типа, то значение расширения типа данных может быть использовано в качестве значения своего базового типа данных.

Примечание — Значения расширенных типов могут быть подставлены вместо значений своего базового типа. Однако в спецификации реализуемой технологии могут быть наложены некоторые ограничения на то, какие расширения являются приемлемыми. В частности, расширения не должны определяться для тех компонентов, которые содержат значения свойств типов данных. Таким образом, хотя любые значения данных могут быть аннотированы вне спецификации типа данных, спецификация реализуемой технологии может не предусматривать способ аннотирования свойства типа данных. В настоящее время комитет HL7 не позволяет использовать параметризованные расширения типов данных за исключением тех ситуаций, когда такие расширения явно разрешены (в настоящей спецификации или в других спецификациях комитета HL7) для тех вариантов использования, где важна такая сложная функциональность. В этих случаях экземпляры указанных параметризованных расширений типов данных должны быть специально и явно отражены в моделях HL7 RIM, MIM, RMIM и HMD (если применимы) в результате утверждения техническим комитетом[1)].

**B.4.1 Тип данных компонента истории HXIT (специализация типа данных T)**

Определение: параметризованный тип данных, добавляющий интервал времени к любому значению любого типа данных. Этот интервал указывает время, в течение которого информация, представленная данным значением, действительна или была действительна.

Если базовый тип данных T не обладает свойством validTime (время действительности), то тип данных HXIT добавляет эту функциональность к базовому типу. Если же базовый тип данных T обладает свойством времени действительности (в настоящее время это свойство имеется только у типа данных EN), то это свойство отображается на свойство времени действительности, предусмотренное в типе данных HXIT[2)].

```
template<ANY T>
type HistoryItem<T> alias HXIT<T> specializes T {
   IVL<TS> validTime;
};
```

B.4.1.1 Свойство validTime: IVL<TS>

Определение: интервал времени, в течение которого данная информация была действительной, либо является действительной, либо ожидается действительной. Любой конец интервала может быть открытым или закрытым, бесконечным или неопределенным.

**B.4.2 Тип данных истории HIST (специализация типа данных SET<HXIT>)**

Определение: множество значений данных, имеющих свойство времени действительности и тем самым соответствующих типу данных HXIT. Информация об истории не ограничена прошлым, в ней могут присутствовать ожидаемые будущие значения.

Тип данных HIST предназначен для хранения действительных исторических (и будущих) значений элемента, а не для журнализации значений, которые конкретная система присваивала этому элементу.

```
template<ANY T>
type History<T> alias HIST<T> specializes SET<HXIT<T>> {
            HXIT<T>  earliest;
            HIST     exceptEarliest;
            HXIT<T>  latest;
            HIST     exceptLatest;
```

[1)] Настоящий стандарт накладывает на себя определенные ограничения, обеспечивающие существующим системам удобный переход на него. Однако в формальной спецификации параметризованные расширения сохранены как подставляемые вместо своих базовых типов. В будущем это самоограничение может быть отменено. В новых разработках рекомендуется предусмотреть некоторую обобщенную поддержку этих параметризованных расширений типов данных.

[2)] Следует учесть, что типы данных являются спецификациями абстрактных свойств значений. Настоящий стандарт не указывает, как эти значения должны быть представлены в спецификации реализуемой технологии или реализованы в прикладной программе. В частности, он не указывает, как представленные компоненты должны быть поименованы или в каком порядке они должны следовать. Кроме того, семантическая иерархия генерализации может отличаться от иерархии классов, выбранной для реализации (если технология реализации обеспечивает наследование). Следует помнить о различии между типом (интерфейсом) и реализацией (конкретной структурой данных, классом). Спецификация реализуемой технологии должна содержать отображение свойств любого определенного в ней типа данных на семантические свойства, определенные в настоящем стандарте.

```
  demotion  HXIT<T>;
};
```

Семантика этого типа данных не содержит принципиального запрета на пересечение интервалов времени действительности. Но если у двух элементов истории границы IVL.low и IVL.high интервалов действительности совпадают, то нет возможности утверждать, который их них должен считаться более ранним, а какой — более поздним.

```
invariant(HIST x)
      where x.nonNull {
   x.notEmpty;
   ((T)x).equal(x.latest);
};
```

B.4.2.1 Свойство earliest: HXIT<T>

Определение: элемент множества интервалов действительности, у которого граница IVL.low (время начала действительности) меньше или равна (то есть более ранняя), чем у любого другого элемента истории из этого множества.

```
invariant(HIST x; HXIT<T> e)
      where x.contains(e) {
   x.earliest.validTime.low.lessOrEqual(e.validTime.low);
};
```

B.4.2.2 Свойство latest: HXIT<T>

Определение: элемент множества интервалов действительности, у которого граница IVL.high (время начала действительности) больше или равна (то есть более поздняя), чем у любого другого элемента истории из этого множества.

```
invariant(HIST x; HXIT<T> e)
      where x.contains(e) {
   x.latest.validTime.high.greaterOrEqual(e.validTime.high);
};
```

B.4.2.3 Свойство exceptEarliest: HIST<T>

Определение: производная история, из которой исключен самый ранний элемент.

```
invariant(HIST x)
      where x.nonNull {
   x.exceptEarliest.equal(x.except(x.earliest));
};
```

B.4.2.4 Свойство exceptLatest: HIST<T>

Определение: производная история, из которой исключен самый поздний элемент.

```
invariant(HIST x)
      where x.nonNull {
   x.exceptLatest.equal(x.except(x.latest));
};
```

B.4.2.5 Приведение истории к одному компоненту: HXIT<T>

Преобразование, при котором значение типа HIST преобразуется в значение типа HXIT. Это преобразование выбирает самый поздний элемент истории.

Такое преобразование полезно в тех случаях, когда поставщик информации передает не одно значение, а историю изменения значений, а потребитель информации, ожидающий получения не истории, а единственного значения, выбирает из нее более позднее.

Следует учесть, что согласно определению тип данных HXIT семантически специализирует тип данных T. Это означает, что потребитель информации, ожидающий значение типа T, но получивший значение типа HXIT, не увидит для себя никакой разницы (подстановка специализации).

**B.4.3 Тип данных вероятностного неопределенного значения UVP (специализация типа данных T)**

Определение: параметризованное расширение типа данных, используемое для указания вероятности, которую, по мнению поставщика информации, имеет данное значение.

Способ вычисления вероятности не входит в область применения настоящей спецификации.

Вероятности субъективны и (как и любое значение данных) должны интерпретироваться в индивидуальном контексте, например, если появилась новая информация, то значение вероятности может измениться. Таким образом, в любом сообщении (в документе или в ином представлении информации) информация, и в частности вероятности, отражает то, что поставщик информации полагал адекватным цели сообщения (документа) в момент создания сообщения (документа).

Например, в начале бейсбольного сезона 2000 г. (май) в Лас-Вегасе ставки на то, что команда «Нью-Йорк Янкиз» выиграет Мировую серию, могли приниматься букмекерами в отношении 1 к 10 (то есть с вероятностью 0,100). Во время написания настоящей спецификации команды «Нью-Йорк Янкиз» и «Нью-Йорк Метс» выиграли в своих подгруппах, и должна была начаться Мировая серия. Очевидно, что в этот момент вероятность, что «Нью-Йорк Янкиз» выиграют Мировую серию, стала ощутимо больше, скажем, 6 к 10 (вероятность 0,600). Все в мире зависти от контекста, в частности, от его даты.

Поскольку вероятности являются субъективными мерами веры, то о них надо говорить не как о «правильных» или «неправильных», а как о «точных» или «неточных». Примечательно, что на проведение экспериментов по частоте появления некоторого результата, позволяющих оценить его вероятность, нельзя рассчитывать. Действительно, какие бы утверждения о людях и событиях ни делались, невозможно подтвердить их вероятность экспериментами «частоты» соответствующих событий.

В условиях предыдущего примера букмекеры Лас-Вегаса не могут настаивать, чтобы «Янкиз» и «Метс» провели 1000 пробных игр до начала Мировой серии. Даже если бы это было возможно, эти игры не носили бы столь ожесточенный характер, как реальная Мировая серия, и вероятность бы не была достаточно точной. Поэтому букмекеры должны оценивать вероятность на основании предыдущих игр, статистики игроков, сведений о травмах и т. д.

```
template<ANY T>
type UncertainValueProbabilistic<T> alias UVP<T> specializes T {
   REAL  probability;
};
```

Формальные ограничения на тип данных T не накладываются. Теоретически дискретные вероятности могут быть приписаны только дискретным значениям данных. Поэтому параметризованный тип данных UVP не должен использоваться для базовых типов данных REAL, PQ или MO.

B.4.3.1 Свойство probability: REAL

Определение: вероятность, приписанная значению и представляющая собой десятичное значение между 0 (невероятно) и 1 (определенно) включительно.

```
invariant(UVP<T> x)
      where x.nonNull.and(x.probability.nonNull) {
   ((IVL<REAL>){0;1}).contains(x.probability);
};
```

Значение вероятности «по умолчанию», которое можно было бы использовать, если вероятность на задана, не существует. Поэтому невозможно провести семантическое различие между значением типа UVP, вероятность которого не указана, и значением типа T. Использование типа данных UVP не означает «неопределенность», а использование простого типа данных T не означает «определенность». Действительно, вероятность значения типа UVP может составлять 0,999 или 1, то есть это значение имеет высокую степень определенности, в то время как значение типа T может быть очень смутной догадкой.

**B.4.4 Тип данных непараметрического распределения вероятности (NPPD) (специализация типа данных SET<UVP>)**

Определение: множество значений типа UVP с вероятностями (известное как гистограмма). Все элементы множества рассматриваются как альтернативные и ранжируются по своей вероятности, выражающей степень веры (или частоту) каждого из этих значений.

Тип данных NPPD в основном предназначен для обеспечения статистических отчетов по измерениям, полученным от многих субъектов и консолидированных в гистограмме. Подобные отчеты встречаются в эпидемиологии, ветеринарии, лабораторном деле, а также в контроле цены и конструировании деловых процессов.

Семантически информация об объявленном значении существует по контрасту с дополнительным множеством необъявленных возможных значений. Таким образом, семантически значение типа NPPD содержит все возможные значения, каждому из которых приписана определенная вероятность.

Наиболее простым способом визуализации является гистограмма наподобие показанной на рисунке B.14.

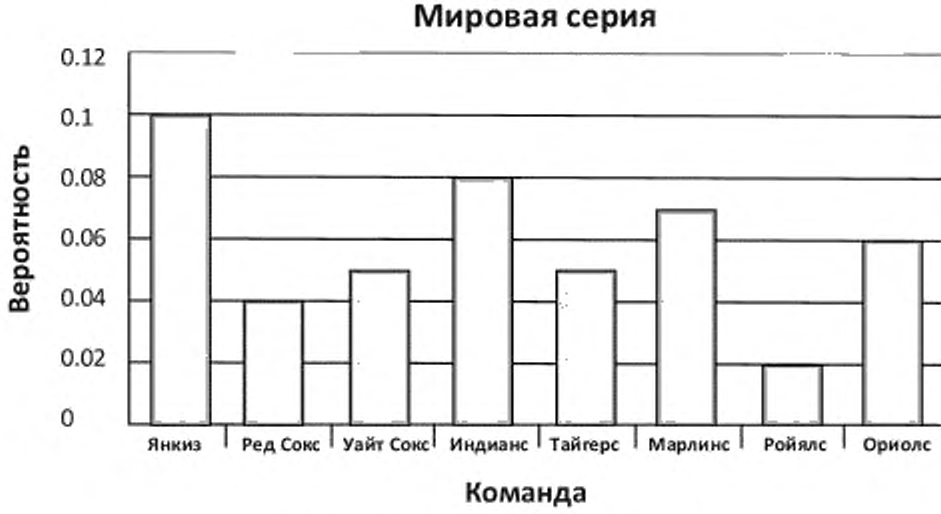


Рисунок В.14 — Пример гистограммы

Этот пример иллюстрирует вероятность выигрыша Мировой серии избранными бейсбольными командами основной лиги (до старта сезона). Команды взаимно исключительны, и если бы на диаграмме были показаны все команды, то сумма вероятностей была бы равна 1 (то есть определенно какая-нибудь команда выиграет).

Примечание — Хотя семантически в типе данных NPPD вероятности присвоены всем возможным значениям, не все значения должны быть представлены явно. Вероятности тех значений, которые не были упомянуты, будут равны равномерному распределению остатка вероятности. Например, если множество значений имеет элементы {A; B; C; D}, но в описании типа данных NPPD указаны только значения {(B; 0,5); (C; 0,25)}, то остаток вероятности составит 1 – 0,75 = 0,25 и будет распределен равномерно по элементам дополняющего множества: {(A; 0,125); (D; 0,125)}. Семантически тип данных NPPD представляет собой объединение объявленного распределения вероятности и необъявленного дополнения с равномерным распределением остатка вероятности.

```
template<ANY T>
type NonParametricProbabilityDistribution<T> alias NPPD<T>
      specializes SET<UVP<T>> {
   SET<UVP<T>> mostLikely(INT n);
};
```

Как и в случае типа данных UVP, на тип данных T не накладываются формальные ограничения, несмотря на то что не все типы данных разумно использовать. Обычно тип данных NPPD используется для неупорядоченных базовых типов T, если только «небольшому» множеству его значений явно приписаны вероятности или если распределение вероятности не может (или не должно) описываться параметрическим методами. В других случаях предпочтительнее использовать тип данных PPD.

В.4.4.1 Свойство mostLikely: UVP (наиболее вероятное значение)

```
invariant(NPPD<T> x)
      where x.nonNull {
   x.notEmpty;
   x.contains(x.mostLikely(n));
   x.mostLikely(n).
   forall(UVP<T> d, e; SET<UVP<T>> m; INT n)
         where x.contains(d).and(m.equal(x.mostLikely(n)))
               .and(m.contains(e)) {
      e.greaterOrEqual(d).or(m.contains(d));
      };
};
```

### B.5 Спецификации моментов и интервалов времени

Обзор типов данных спецификации моментов и интервалов времени показан на рисунке B.15.

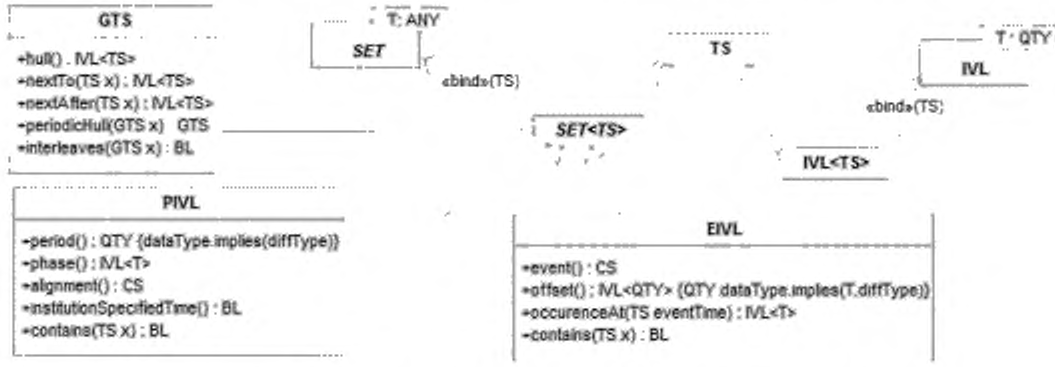


Рисунок B.15 — Обзор типов данных спецификации моментов и интервалов времени

Комплекс спецификаций моментов и интервалов времени используется для указания сложных сроков событий и действий, которые, к примеру, случаются в системах управления заказами и ведения расписаний. Он также поддерживает циклические шаблоны действительности, которые могут применяться к определенным видам информации, например к телефонным номерам (вечерним, дневным), часам работы, или к адресам так называемых «перелетных птиц» (людям, которые предпочитают зимой быть поближе к экватору, а летом подальше от него).

Спецификации моментов и интервалов времени включают в себя момент времени (тип данных TS), интервал времени (тип данных IVL<TS>), а также дополнительные типы данных, специально сконструированные для повторяющихся расписаний. К ним относятся типы данных PIVL, EIVL и, наконец, собственно тип данных GTS. Все эти типы данных описывают распределение моментов времени повторения состояний или событий.

#### B.5.1 Тип данных периодического интервала времени PIVL (специализация типа данных SET)

Определение: периодически повторяющийся интервал времени. Тип данных PIVL имеет два свойства: phase (фаза) и period (период). Свойство phase задает «прототип интервала», повторяющийся с периодичностью, указанной в свойстве period.

Таблица B.43 — Сводка свойств типа данных PIVL

| Имя | Тип | Описание |
|---|---|---|
| phase | IVL<T> | Прототип повторяющегося интервала, задающий длительность каждого повторения и привязку последовательности компонентов типа данных PIVL к определенному моменту времени |
| period | T.diff | Длительность, указанная как обратная величина частоты повторения компонентов типа данных PIVL |
| alignment | CS | Указывает, должны ли повторения привязываться к календарным циклам, и если да, то каким образом (например, чтобы отличать «каждые 30 дней» от «5-го числа каждого месяца»). Непривязанное значение типа PIVL описывает повторения независимо от календаря. Привязанное значение типа PIVL синхронизируется с календарем |
| institutionSpecifiedTime | BL | Указывает, оставляется ли точное время события на усмотрение стороны, которая ведет расписание (например, чтобы отличать «каждые 8 часов» от «3 раза в день») |

Например, расписание «каждые 8 ч в течение 2 мин.» можно представить как значение типа PIVL, у которого интервал IVL.width равен двум минутам, а период повторения period равен 8 ч.

Свойство phase также обозначает привязку всей серии периодически повторяющихся интервалов к определенному моменту времени. Повторения, описываемые типом данных PIVL, не имеют ни начала, ни конца. Они бесконечны как в будущем, так и в прошлом.

```
template<TS T>
protected type PeriodicInterval<T> alias PIVL<T>
            specializes  SET<T> {
            T.diff       period;
            IVL<T>       phase;
            CS           alignment;
            BL           institutionSpecifiedTime;
            BL           contains(TS);
   literal  ST;
};
```

Тип данных PIVL полностью определен, если полностью определены его свойства period и phase. Интервал может быть определен только частично, если указаны либо только его ширина IVL.width, либо только одна из его границ.

Например, расписание «каждые 8 ч в течение 2 мин.» задает только период period и ширину IVL.width фазы phase, но не границу фазы phase. Напротив, расписание «каждые 8 ч, начиная с 4 ч» задает только период period и нижнюю границу IVL.low фазы phase, но не ее верхнюю границу IVL.high. Расписание «каждые 8 ч в течение 2 мин., начиная с 4 ч» полностью определено, поскольку заданы как период period, так и оба значения IVL.low и IVL.width для фазы phase.

Тип данных PIVL представляет собой параметризованное расширение типа данных, у которого параметр типа T ограничен типом данных TS и его расширениями. Поскольку тип данных PPD<TS>> представляет собой расширение типа данных TS, то он может использоваться для формирования значений типа PIVL<PPD<TS>>.

Нередко моменты времени и повторяющиеся расписания определены только приблизительно. Например, расписание «три раза в день в течение 10 мин.» обычно не означает, что свойство period в точности равно 8 ч, а интервал в точности 10 мин. На практике дистанция между событиями такого расписания может варьироваться от 3 до 12 ч, а ширина IVL.width интервала может быт меньше 5 мин. или больше 15 мин. Для указания степени свободы или «критичности ко времени» подобного расписания может использоваться тип данных PIVL<PPD<TS>>.

B.5.1.1 Свойство phase: IVL<T>

Определение: прототип повторяющегося интервала, задающий длительность каждого повторения и привязку последовательности компонентов типа данных PIVL к определенному моменту времени.

Свойство phase также обозначает привязку всей серии периодически повторяющихся интервалов к определенному моменту времени. Повторения, описываемые типом данных PIVL, не имеют ни начала, ни конца. Они бесконечны как в будущем, так и в прошлом. Значение компонента IVL.width фазы phase должно быть меньше значения свойства period или равно ему.

```
invariant (PIVL<T> x)
      where x.nonNull {
   x.phase.nonNull.implies(x.phase.width.lessOrEqual(x.period));
};
```

B.5.1.2 Свойство period: T.diff

Определение: длительность, указанная как обратная величина частоты повторения компонентов типа данных PIVL.

Свойство period имеет тип данных QTY с размерностью времени (T.diff). Для неопределенных значений типа PIVL свойство period представляет собой распределение вероятности прошедшего времени.

```
invariant(PIVL<T> x)
      where x.nonNull {
   x.period.nonNull;
};

invariant(PIVL x) {
   x.period.dataType.implies(PQ);
};
```

B.5.1.3 Свойство alignment: CS

Определение: указывает, должны ли повторения привязываться к календарным циклам, и если да, то каким образом (например, чтобы отличать «каждые 30 дней» от «5-го числа каждого месяца»). Непривязанное значение типа PIVL описывает повторения независимо от календаря. Привязанное значение типа PIVL синхронизируется с календарем.

Например, расписание «5-го числа каждого месяца» представляет собой значение типа PIVL, привязанное к календарю. Значение свойства period варьируется от 28 до 31 дней в зависимости от календарного месяца.

Напротив, расписание «каждые 30 дней» описывает не зависящее от календаря свойство period, которое каждый месяц приходится на разный день.

При привязке значения типа PIVL к календарю должен быть указан календарный цикл, к которому осуществляется привязка. Затем сплошной поток времени подразделяется на календарные циклы. Такое подразделение называется календарной «решеткой», генерируемой привязываемым календарным циклом. В каждом подразделе границы очередного интервала отстоят на одно и то же время от начального момента каждого подраздела. Другими словами, промежуток времени от следующей нижней линии решетки до начала интервала постоянен.

Например, в расписании «5-е число каждого месяца» привязка осуществляется к месяцу года (MY). Сплошной поток времени подразделяется на месяцы года. Промежуток времени от начала каждого месяца до начала очередного интервала составляет четыре дня (поскольку отсчет дней месяца (DM) начинается с 1). Таким образом, промежутки времени между последовательными интервалами будут немного меняться, так что начало интервала каждый раз будет приходиться на 5-е число.

B.5.1.4 Свойство institutionSpecifiedTime: BL

Определение: это свойство указывает, оставляется ли точное время события на усмотрение стороны, которая ведет расписание (например, чтобы отличать «каждые 8 ч» от «3 раза в день»).

Например, при расписании «три раза в день» среднее время между повторениями составляет 8 ч, но в случае, если свойство institutionSpecifiedTime имеет значение «true», точное время повторения должно следовать некоторому правилу, установленному лицом или организацией («стороной»), например, три раза в день могут означать повторения в 7:00, 12:00 и 19:00.

B.5.1.5 Литеральная форма

B.5.1.5.1 Общая литеральная форма

Общая литеральная форма периодических интервалов времени имеет следующий вид:

**(phase : IVL<T>( / (period : QTY ( [ @ (alignment( ] [ IST ]**

```
PIVL<T>.literal ST {
   PIVL<T>     : S2                    { $.equal($1); }
               | S2 "IST"              { $.phase.equal($1.phase);
                                         $.period.equal($1.period);
                                         $.institutionSpecified.equal(true); };
   PIVL<T> S2 : S1                     { $.equal($1); }
               | S1 "@" "(" ST ")"     { $.phase.equal($1.phase);
                                         $.period.equal($1.period);
                                         $.alignment.equal($4); };
   PIVL<T> S1 :
      IVL<T> "/" "(" QTY ")"           { $.phase.equal($1);
                                         $.period.equal($3); }
               | "/" "(" QTY ")"       { $.period.equal($2); };
};
```

Например, строка «[200004181100;200004181110]/(7 d)@DW» задает каждый вторник с 11:00 до 11:10. С другой стороны, строка «[200004181100;200004181110]/(1 mo)@DM» задает каждое 18-е число месяца с 11:00 до 11:10.

Коды календарных циклов, определенные для григорианского календаря, приведены в таблице B.36. В ней указаны однобуквенные и двухбуквенные коды. В качестве значений свойства alignment предпочтительны двухбуквенные коды.

B.5.1.5.2 Форма календарного шаблона

Эта форма используется для указания привязки повторений к календарю в более интуитивной форме «календарных шаблонов». Синтаксис календарного шаблона (полуформально) определяется следующим образом:

**(привязка( [ (цифры календарного цикла( [ .. (цифры календарного цикла ( ]] / (число : INT( [ IST ]**

Календарный шаблон представляет собой дату календаря, у которой опущены старшие значащие цифры (например, год и месяц). Для интерпретации цифр идентификатору периода приписывается префикс, идентифицирующий календарный цикл крайних левых цифр. Этот идентификатор календарного цикла «привязывает» правые цифры даты.

Коды календарных циклов, определенные для григорианского календаря, приведены в таблице B.36. В ней указаны однобуквенные и двухбуквенные коды. В качестве обозначения привязки календарного шаблона предпочтительны однобуквенные коды.

Например, строка «M0219» означает весь день 19 февраля каждого года. Свойство phase этого периодического интервала соответствует 19 февраля каждого года (например, «[19690219;19690220[»), свойство period имеет значение одного года, а свойство alignment означает месяц года (M). Привязанный календарный цикл совпадает с привязкой (в этом примере с месяцем года).

В календарном шаблоне могут быть также опущены правые цифры. Это означает интервал от наименьшего до наибольшего значения опущенных цифр. Например, строка «M0219» означает весь день 19 февраля; строка «M021918» — весь час 19 февраля от 18:00 до 19:00.

В отсутствие формального определения календарного шаблона используются следующие правила его разбора (на примере строки «M021918..21»):

1. Считать идентификатор привязываемого цикла (например, «M»).

2. Положить привязку равной этому календарному циклу (например, месяцу года).

3. Взять текущий момент времени и форматировать литерал точно до более высокого календарного цикла, следующего за привязываемым циклом (например, взять год «2000» и сконструировать строку «2000021918»); этот литерал служит «стеблем».

4. Считать сконструированный литерал (например, «2000021918») в значение типа TS и преобразовать его в интервал в соответствии со свойством IVL_TS.promotionTS (например, «[2000021918;2000021919[»); это «нижний интервал».

5. Если после разобранной части строки следует оператор оболочки «..», считать следующие цифры шаблона (например, «21»).

6. Выравнять вправо литерал «стебля» и только что считанные цифры шаблона:

```
"2000021918"
"         21";
```

7. Затем скопировать все цифры литерала стебля, отсутствующие в только что считанных цифрах шаблона (например получить «2000021921»).

8. Считать этот сконструированный литерал (например, «2000021918») в значение типа TS и преобразовать его в интервал в соответствии со свойством IVL_TS.promotionTS (например, «[2000021921;2000021922[»); это «верхний интервал».

9. Положить свойство phase равным выпуклой оболочке нижнего и верхнего интервала (например, «[2000021918;2000021922[»).

10. Если оператор оболочки отсутствует, то положить свойство phase равным нижнему интервалу.

B.5.1.5.3 Чередование

Календарный шаблон, после которого указана косая черта и целое число n, означающее, что данный шаблон применяется каждый n-й раз.

Например, строка «D19/2» задает 19-е число каждого второго месяца.

Выражение календарного шаблона применяется при первом появлении шаблона. В этот момент отсутствующие левые цифры шаблона дополняются, используя самую раннюю дату, соответствующую шаблону (и следуя предшествующему шаблону в комбинации спецификаций времени).

Например, строка «D19/2» задает 19-е число каждого второго месяца. Если эта строка применяется 14 марта 2000 г., то свойству phase присваивается значение «[20000319;20000320[/(2 mo)@DM» и двухмесячный цикл начинается 19 марта, затем 19 мая и т. д. Если эта строка применяется 20 марта, то цикл начнется 19 апреля, затем 19 июня и т. д.

Если после идентификатора календарного цикла не указаны цифры, то шаблон совпадает с любой датой. Целое число, указанное после косой черты, означает продолжительность цикла. В этих случаях свойство phase обладает только свойством ширины width, указывающим длительность привязанного календарного цикла (в данном примере — один день).

Например, строка «CD/2» означает каждый второй день, строка «H/8» означает каждый восьмой час с длительностью интервала один час.

B.5.1.5.4 Время, задаваемое стороной

После литерала типа данных PIVL или календарного шаблона могут быть указаны три буквы «IST» (Institution Specified Time — время, задаваемое стороной), обозначающие, что внутри большего календарного цикла (например, для цикла «час дня» большим циклом будет «день») повторяющиеся события планируются во время, задаваемое стороной. Этот суффикс используется при задании такого цикла, как «три раза в день», при котором между двумя последовательными событиями может пройти и четыре часа (между завтраком и обедом), и 10 ч (охватывающие ночь).

Таблица B.44 — Примеры литеральных выражений типа данных PIVL

| Общая литеральная форма | Форма календарного шаблона | Описание |
|---|---|---|
| [198709:198710[/(1 a)@MY | M09 | Весь месяц сентябрь каждого года (следует обратить внимание, что указание 1987 г. в общей литеральной форме не имеет особого значения, поскольку периодические интервалы повторяются каждый год как в прошлом, так и в будущем) |

*Продолжение таблицы В.44*

| Общая литеральная форма | Форма календарного шаблона | Описание |
|---|---|---|
| [19870915;19870916[/(1 a)@DM | M0915 | Весь день 15 сентября каждого года |
| [1987091516;1987091517[/(1 a)@DM | M091516 | Весь час с 16 ч 15 сентября каждого года |
| [198709151630;198709151710]/(1 a)@DM | M09151630..1710 | Интервал 16:30—17:10, повторяющийся каждый год 15 сентября |
| [1987091516;[/(1 a)@DM | | Интервал, начинающийся с 16 ч, конец интервала явно не указан. Повторяется каждый год 15 сентября |
| [198709151630;198709151631[/(1 a)@DM | M09151630 | Интервал в целую минуту, начиная с 16:30. Повторяется каждый год 15 сентября |
| [1987091516;1987091517[/(1mo)@DM | D1516..17 | 15-й день каждого месяца с 16 до 17 ч |
| [1987091516;1987091517[/(1 mo) | | 15 сентября 1987 г. с 16 до 17 ч, и затем через каждые 730,5 ч (в этом примере мало практического смысла, он приведен для сравнения непривязанной формы с привязанной, указанной в предыдущей строке) |
| [1987091516;1987091517[/(1 mo)@HD | | 15 сентября 1987 г. с 16 до 17 ч и затем через каждые 30,4375 дней, но привязанный к часу дня |
| [1 mo]/(2 mo)@MY | M/2 | Каждый второй месяц года; не определено, будет ли это последовательность (январь, март...) или (февраль, апрель...) |
| [198701;197502[/(2 mo)@MY | M01..12/2 | Каждый второй месяц года, январь, март... |
| [198702;197503[/(2 mo)@MY | M02..12/2 | Каждый второй месяц года, февраль, апрель... |
| [19870401;19870930[/(1 a)@DM | M04..09 | С 1 апреля по 30 сентября (включительно) |
| 19870401-0930/(1 a)@DM | M0401..0930 | С 1 апреля по 30 сентября (в общей литеральной форме фаза phase задана через дефис) |
| [20001202;20001203[/(1 wk)@DW | J6 | Каждую субботу |
| [20001202;20001203[/(2 wk)@DW | J6/2 | Каждую вторую субботу |
| [20001202;20001203[/(3 wk)@DW | J6/3 | Каждую третью субботу |
| [1 d]/(2 d)@DW | J/2 | Каждый второй день недели, не определено, будет ли это последовательность (понедельник, среда, пятница...) или (вторник, четверг, суббота...) |
| [20001204;20001205[/(2 d)@DW | J2..6/2 | Каждый второй день недели (вторник, четверг, суббота, вторник, четверг, суббота...) |
| [20001204;20001205[/(2 d) | D/2 | Каждый второй день (вторник, четверг, суббота, понедельник, среда, пятница, воскресенье, вторник...) |
| [19870601;19870606[/(1 wk)@DW | J1..5 | Каждую неделю с понедельника по пятницу |
| [19870601;19870608[/(2 wk) | W/2 | Каждую вторую неделю (непрерывно) |

*Окончание таблицы В.44*

| Общая литеральная форма | Форма календарного шаблона | Описание |
|---|---|---|
| [19870101;19870105[/(2 wk)@WY | WY/2 | Каждую вторую неделю года (характерный пример влияния привязки к календарю: продолжительность фазы составляет только 4 дня и тем не менее представляет целую неделю в календарной привязке «неделя года») |
| [19870406;19870413[/(1 a)@WY | WY15 | 15-я календарная неделя каждого года |
| [19870105;19870112[/(1 mo)@WM | WM2 | Вторая неделя каждого месяца |
| [19870508;19870509[/(1 a)@DY | DY128 | 128-й день каждого года |
| [10 min]/(2 d) | | Каждый второй день в течение 10 мин. (известна только ширина повторяющегося интервала) |
| [1 h]/(8 h) | H/8 | Каждый восьмой час (в течение 60 мин.) |
| [1 h]/(8 h) IST | H/8 IST | Три раза в день во время, заданное стороной (каждый раз в течение 60 мин.) |
| /(8 h) IST | | Три раза в день во время, заданное стороной. О повторяющемся интервале ничего не известно, то есть литерал описывает только период (частоту), а фаза остается неопределенной |

В.5.1.5 Периодические интервалы как множества

Существенным свойством множества является то, что оно содержит элементы. Для значений типов PIVL, не привязанных к календарю, свойство contains (содержит) определяется следующим образом: элемент t типа данных TS содержится во множестве типа PIVL в том и только том случае, если существует целое число i, для которого сумма t с произведением period на i является элементом фазы phase.

```
invariant (PIVL<TS> x, TS t)
      where x.nonNull.and(x.alignment.isNull) {
   x.contains(t).equal(exists(INT i) {
      x.phase.contains(t.plus(x.period.times(i)));
      });
};
```

Для значений типов PIVL, привязанных к календарю, свойство contains определяется, используя свойство календарного цикла sum(t, n), которое добавляет n таких календарных циклов к моменту времени t.

```
invariant (PIVL<TS> x, TS t, CalendarCycle c)
      where x.nonNull.and(c.equal(x.alignment)) {
   x.contains(t).equal(exists(INT i) {
      x.phase.contains(c.sum(t, i));
      });
};
```

**В.5.2 Тип данных периодического интервала времени с привязкой к событию EIVL (специализация типа данных SET)**

Определение: задает периодический интервал времени, повторения которого основаны на событиях повседневной деятельности или других важных событиях, которые связаны с временем, но не имеют точно заданного времени.

Например, расписание «через час после завтрака» указывает, что интервал начинается через час после завершения завтрака. Предполагается, что завтрак происходит до обеда, но при этом никакое конкретное время завтрака не определено.

```
template<TS T>
protected type EventRelatedPeriodicInterval<T> alias EIVL<T>
```

```
            specializes  SET<T>{
            CS           event;
            IVL<PQ>      offset;
            IVL<T>       occurrenceAt(TS eventTime);
            BL           contains(TS);
  literal   ST;
};
```

B.5.2.1 Свойство event: CS

Определение: код повседневной (периодической) деятельности, на основе которой определяется периодический интервал, связанный с событием.

В домен значений этого атрибута могут включаться такие события, для которых верно все нижеперечисленное:

- событие обычно происходит на регулярной основе;
- событие означает деятельность в течение некоторого времени;
- событие не полностью определяется временем.

Таблица B.45 — Домен значений свойства event

| Код | Имя | Определение |
|---|---|---|
| AC | AC | Перед едой (лат. ante cibus) |
| ACD | ACT | Перед обедом (лат. ante cibus diurnus) |
| ACM | ACM | Перед завтраком (лат. ante cibus matutinus) |
| ACV | ACV | Перед ужином (лат. ante cibus vespertinus) |
| HS | HS | Часы сна |
| IC | IC | Между приемами пищи (лат. inter cibus) |
| ICD | ICD | Между обедом и ужином |
| ICM | ICM | Между завтраком и обедом |
| ICV | ICV | Между ужином и часами сна |
| PC | PC | После еды (post cibus) |
| PCD | PCD | После обеда (post cibus diurnus) |
| PCM | PCM | После завтрака (post cibus matutinus) |
| PCV | PCV | После ужина (post cibus vespertinus) |

B.5.2.2 Свойство offset: IVL<PQ>

Определение: интервал прошедшего времени (длительность, а не абсолютные моменты времени), задающий смещения начала, ширины и конца значения типа EIVL относительно времени фактического совершения такого события.

Например, для расписания «за один час до завтрака в течение 10 минут» значение IVL.low имеет свойство offset, равное 1 час, а значение IVL.width имеет свойство offset, равное 10 минут.

B.5.2.3 Литеральная форма

Литеральная форма типа данных EIVL начинается с кода события, за которым следует необязательный интервал разности во времени.

```
EIVL<TS>.literal ST {
   EIVL<TS> : event          { $.event.equal($1); }
            | event offset   { $.event.equal($1);
                               $.offset.equal($2); };
   CS event : ST             { $.code.equal($1);
                               $.codeSystem.equal(2.16.840.1.113883.5.1019); }
   IVL<PQ> offset
            : "+" IVL<PQ>    { $.equal($2); }
            | "-" IVL<PQ>    { $.low.equal($2.high.negate);
                               $.high.equal($2.low.negate);
                               $.width.equal($2.width);
                               $.lowClosed($2.highClosed);
```

```
                                $.highClosed($2.lowClosed); };
};
```

Например, расписание «через один час после еды» имеет литерал «PC+[1h;1h]». Расписание «за один час перед сном в течение 10 минут» имеет литерал «HS-[50min;1h]».

B.5.2.4 Разрешение связи с событием

Тип данных EIVL описывает множество моментов времени, поэтому можно проверить, является ли конкретный момент или интервал времени элементом этого множества. Вопрос, содержит ли значение типа EIVL данный интервал времени, разрешается с помощью отношения χ time, обозначаемого как EVENT(event, time). Свойство occurrenceAt(t) задает повторение интервала, имеющее место, если событие происходит в момент времени t.

```
invariant(EIVL<T> x, T eventTime, IVL<T> v)
      where v.equal(x.occurrenceAt(eventTime)) {
   v.low.equal(eventTime.plus(x.offset.low));
   v.high.equal(eventTime.plus(x.offset.high));
   v.lowClosed.equal(x.offset.lowClosed);
   v.highClosed.equal(x.offset.highClosed);
};
```

Таким образом, значение типа EIVL содержит момент времени t типа TS, если существует время события e с повторением интервала v, содержащим t.

```
invariant(EIVL<T> x, T y) {
   x.contains(y).equal(exists(T e, IVL<T> v)
      where EVENT(x.event, y).and(v.resolvedAt(y)) {
         v.contains(y);
         });
};
```

### B.5.3 Тип данных общей спецификации времени GTS (специализация типа данных SET<TS>)

Определение: множество моментов времени, указывающее времена событий и действий, а также шаблоны циклов, которые могут применяться к некоторым видам информации, например, к телефонным номерам (утро, вечер), к адресам (для так называемых «перелетных птиц», которые живут зимой на юге, а летом на севере), а также к часам работы.

Тип GTS имеет следующие особенности:

- он является общим типом множества моментов времени SET<TS>. Поэтому тип данных GTS может использоваться для определения того, попадает ли данное значение типа TS в расписание, описанное значением типа GTS;
- тип GTS представляет собой сочетание нескольких интервалов типа PIVL. Эта особенность позволяет определить, как именно простые и комплексные шаблоны повторений могут быть описаны с помощью типа данных GTS;
- тип данных GTS может служить генератором списков типа LIST<IVL<TS>>. Эта особенность позволяет использовать тип данных GTS для генерирования всех повторяющихся интервалов события или действия либо всех периодов действительности факта,
- тип данных GTS может использоваться как синтаксис календарных выражений. Эта особенность характерна для литеральной формы этого типа данных.

В любом случае тип данных GTS определяется как множество моментов времени SET<TS>. С помощью свойств SET.union, SET.intersect и SET.difference из простых множеств типа SET<TS> можно сконструировать более сложные. В конечном счете строительными блоками всех значений типа GTS являются типы данных IVL, PIVL и EIVL. Конструкция значения типа GTS может быть определена в литеральной форме. Для генерации комбинации более простых множеств моментов времени из данного значения типа GTS не определена никакая специальная структура типа данных. Хотя любая реализация и могла бы содержать такое структурированное представление, оно не является необходимым, чтобы передавать значения типа GTS в литеральной форме[1)].

```
type GeneralTimingSpecification alias GTS specializes SET<TS> {
            IVL<TS>   hull;
            IVL<TS>   nextTo(TS x)
```

[1)] Тип данных GTS служит примером типа данных, который имеет только алгебраическое определение, а не описывается структурой данных, с помощью которой можно было бы реализовать поведение такого типа данных. Алгебраическое определение выглядит чрезвычайно простым, поэтому можно подозревать его неполноту. Поскольку в настоящее время для представления значений типа GTS используется только литеральная форма, то все определение структуры данных оставлено на будущее.

```
            IVL<TS>   nextAfter(TS x)
            GTS       periodicHull(GTS x);
            BL        interleaves(GTS x);
  demotion  LIST<IVL<TS>>;
  literal   ST;
};
```

B.5.3.1 Выпуклая оболочка

Выпуклая оболочка представляет собой наименьший интервал, содержащий все интервалы расписания. Как указано в описании свойства SET.hull, для всех полностью упорядоченных множеств определена выпуклая оболочка. Поскольку тип данных GTS является специализацией типа данных SET<TS>, значения которого полностью упорядочены, то для всех значений типа GTS определена выпуклая оболочка.

Выпуклая оболочка значения типа GTS может быть менее формально названа «внешним связным интервалом». Таким образом, выпуклая оболочка значения типа GTS описывает абсолютные начало и конец повторяющегося расписания. Для бесконечных повторений (например, описываемых типом данных PIVL) выпуклая оболочка имеет бесконечные границы.

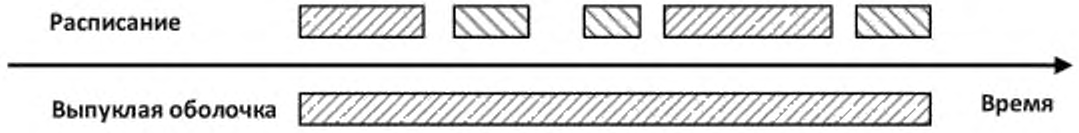


Рисунок B.16 — Выпуклая оболочка расписания

B.5.3.2 Тип данных GTS как последовательность интервалов событий

Значение типа GTS является генератором интервалов, в течение которых происходят события или осуществляются действия либо действительно некоторое состояние.

Свойство nextTo отображает каждый момент времени t на наибольшее непрерывное подмножество («интервал события») v значения S типа GTS, где v — ближайший интервал к моменту t, начинающийся позже момента t или содержащий момент t.

```
invariant(GTS S, TS t, IVL<TS> v) {
   v.equal(S.nextTo(t)).equal(
      S.contains(v).and(
         forall(IVL<TS> u) where x.contains(u) {
            u.contains(v).implies(u.equal(v));
            };
         .and(v.contains(t).or(forall(TS i) where t.lessOrEqual(i)
               .and(i.lessThan(v.low)) {
            S.contains(i).not; })));
};
```

Свойство nextAfter отображает каждый момент времени t на наибольшее непрерывное подмножество («интервал события») v значения S типа GTS, которое представляет собой ближайший интервал к моменту t, начинающийся позже момента t.

```
invariant(GTS S, TS t) {
   S.contains(t).not.implies(S.nextAfter(t).equal(S.nextTo(t)));
   S.contains(t).implies(S.nextAfter(t).equal(
               S.except(nextTo(t)).nextTo(t)));
};

invariant(GTS S, TS t) {
   S.contains(t).not.implies(S.nextAfter(t).equal(S.nextTo(t)));
   S.contains(t).implies(S.nextAfter(t).equal(
               S.except(nextTo(t)).nextTo(t)));
};
```

Значение типа GTS может быть преобразовано в список типа LIST<IVL<TS>>.

```
invariant(GTS x)
      where x.isEmpty {
```

```
   ((LIST<IVL<TS>>)x).isEmpty; };
invariant(GTS x, IVL<TS> first)
      where x.notEmpty.and(x.hull.low.nonNull)
                      .and(first.equal(x.nextTo(x.hull.low)) {
   ((LIST<IVL<TS>>)x).head.equal(first);
   ((LIST<IVL<TS>>)x).tail.equal(
      (LIST<IVL<TS>>)x.except(first));
};
```

B.5.3.3 Чередующиеся расписания и периодические оболочки

Два значения A и B типа GTS чередуются, если интервалы их событий чередуются на оси времени. Это понятие иллюстрируется рисунком B.17.

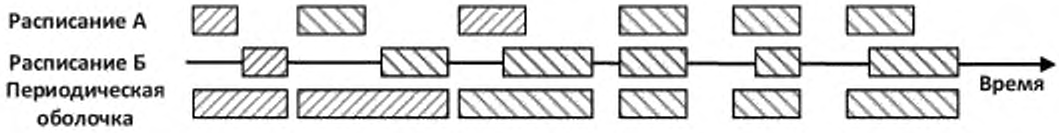


Рисунок B.17 — Чередующиеся расписания и периодическая оболочка

Значения A и B типа GTS считаются чередующимися, если интервалы событий обоих групп могут быть попарно упорядочены. При этом соответствующие пары интервалов $a \subseteq A$ и $b \subseteq B$ должны удовлетворять следующему условию: интервал a начинается до начала интервала b (или в то же самое время), и интервал b завершается после завершения интервала a (или в то же самое время).

Отношение чередования имеет место, если два расписания имеют одинаковую среднюю частоту и при этом второе расписание никогда не «перекрывает» первое расписание. Другими словами, никакой интервал события второго расписания не может начаться раньше соответствующего ему интервала события первого расписания.

Для двух чередующихся значений типа GTS можно определить периодическую оболочку, представляющую собой совокупность выпуклых оболочек соответствующих пар интервалов событий.

Периодическая оболочка важна для конструирования двух расписаний с помощью сочетания значений типа GTS. Например, для конструирования ежегодного периодического интервала от Дня памяти (последний понедельник мая) до Дня труда (первый понедельник сентября) можно сконструировать расписание M для Дня памяти и расписание L для Дня труда, а затем скомбинировать эти два расписания, используя периодическую оболочку расписаний M и L.

```
invariant(GTS A, B)
      where x.nonNull.and(y.nonNull) {
   A.interleaves(B).equal(
      forall(IVL<TS> a, b, c; TS t)
         where a.equal(A.nextTo(t))
                .and(b.equal(B.nextTo(a.low)))
                .and(c.equal(A.nextTo(b.high))) {
            b.equal(B.nextTo(a.high));
            a.low.lessOrEqual(b.low);
            c.equal(A.nextTo(b.high));
            c.equal(a).or(c.equal(A.nextAfter(a.high)));
            });
};
```

Периодическая оболочка значений A и B типа GTS, где B чередуется с A, определяется как попарная выпуклая оболочка соответствующих интервалов появлений A и B.

```
invariant(GTS A, B, C)
      where A.interleaves(B) {
   A.periodicHull(B).equal(C).equal(
      forall(IVL<TS> a, b; TS t)
         where a.equal(A.nextTo(t))
                .and(b.equal(B.nextTo(a.low))) {
            C.contains(c).equal(c.equal(a.hull(b)));
            });
};
```

Отношение чередования является рефлексивным, не симметричным и не транзитивным. Операция периодической оболочки не коммутативна и не ассоциативна[1].

B.5.3.4 Литеральная форма

Литерал типа данных GTS позволяет указывать комбинации значений типов данных IVL<TS> и PIVL, используя операции объединения и пересечения множеств[2].

Объединения описываются в форме списка значений, разделенных точками с запятыми. Пересечения задаются в форме списка значений, разделенных пробельными символами. Операция пересечения имеет более высокий приоритет по отношению к операции объединения. Разность множеств задается с помощью символа обратной косой черты, она имеет промежуточный приоритет, то есть слабее пересечения, но сильнее объединения. При необходимости для переопределения порядка применения операций могут использоваться скобки.

```
GTS.literal ST {
   GTS symbol : union                    { $.equal($1); }
              | exclusion                { $.equal($1); };
   SET<TS> union
              : symbol ";" intersection  { $.equal($1.union($3)); }
              | intersection             { $.equal($1); };
   SET<TS> exclusion
              : symbol "\" intersection  { $.equal($1.except($3)); }
              | intersection             { $.equal($1); };
   SET<TS> intersection
              : hull intersection        { $.equal($1.intersection($2)); }
              | hull                     { $.equal($1); };
   SET<TS> hull
              : hull ".." factor         { $.equal($1.periodicHull($3)); }
              | factor                   { $.equal($1); };
   SET<TS> factor
              : IVL<TS>                  { $.equal($1); }
              | PIVL<TS>                 { $.equal($1); }
              | EIVL<TS>                 { $.equal($1); }
              | "(" GTS ")"              { $.equal($1); };
};
```

В таблице B.46 приведены практические примеры комплексных литералов типа данных GTS. Более простые примеры приведены в описании литеральных форм типов данных IVL, PIVL и EIVL.

Таблица B.46 — Примеры литеральных выражений типа данных GTS

| Литеральное выражение | Значение |
|---|---|
| M09 D15 H16 N30 S34.12 | 15 сентября, 16:30:34.12 как пересечение нескольких периодических интервалов времени (календарных шаблонов) |
| M0915163034.12 | 15 сентября, 16:30:34.12 как один простой периодический интервал времени (календарный шаблон) |
| M01; M03; M07 | Январь, март и июль (объединение трех периодических интервалов времени) |
| M04..09 M/2 | Каждый второй месяц с апреля по сентябрь (апрель, июнь, август) |
| J1; J2; J4 | Понедельник, вторник, четверг |

[1] Свойство чередования может показаться чрезмерно ограниченным. Однако эти ограничения вполне разумны в тех случаях, когда определяются свойства чередования и периодической оболочки. Для безопасного и предсказуемого комбинирования двух расписаний необходимо знать, который из операндов задает начальные моменты времени, а какой — конечные моменты периодической оболочки интервалов событий.

[2] Эта спецификация литерала опять-таки выглядит удивительно простой, поэтому можно подозревать ее неполной. Однако литерал типа данных GTS основан на литералах типов данных TS, IVL, PIVL и EIVL, а также подразумевает литералы расширений типа данных TS, в частности параметризованного типа данных PPD<TS>. Спецификация самого литерала GTS должна только связывать между собой другие литеральные формы, что само по себе представляется достаточно простой задачей.

*Окончание таблицы В.46*

| Литеральное выражение | Значение |
|---|---|
| W/2 J2 | Каждый второй вторник (пересечение каждой второй недели и каждого вторника) |
| 1999 WY15 | 15-я календарная неделя 1999 г. (код периода старшей календарной единицы не обязателен) |
| WM2 J6 | Суббота второй недели месяца |
| M05 WM2 J6 | Суббота второй недели мая |
| M05 DM08..14 J7 | День матери (второе воскресенье мая) |
| J1..5 H0800..1600 | С 8:00 до 16:00 с понедельника по пятницу |
| J1..4 H0800..1600; J5 H0800..1200 | С 8:00 до 16:00 с понедельника по четверг и с 8:00 до полудня в пятницу |
| [10 d] H/8 | Три раза в день в течение 10 дней (каждый раз в течение 60-минутного интервала) |
| H0800..1600 \J3 | С 8:00 до 16:00 каждый день, кроме среды |
| (M0825..31 J1)..M0831 | Последняя календарная неделя августа |
| JHNUSMEM..JHNUSLBR | Сезон отпусков в США с Дня памяти до Дня труда |

В.5.3.4.1 Символьное сокращение выражений типа данных GTS

В таблице В.47 приведены символьные сокращения значений типа GTS, которые могут использоваться в литералах вместо их эквивалентов. Сокращения определены для общих периодов дня (AM — утро, PM — пополудни), для периодов недели (рабочий день, выходной) и для праздников. Вычисления дат некоторых праздников, прежде всего Пасхи, представляют определенную сложность, превышающую возможности их представления в форме литерала типа данных GTS. Предполагается, что даты таких праздников берутся из некоторой таблицы или генерируются некоторым модулем вне области применения настоящей спецификации.

Эти сокращения являются именованными значениями типа GTS и могут, в свою очередь, использоваться в литерале типа GTS. Например, можно использовать строку «JHCHRXME H08..12» для указания, что рабочие часы в канун Рождества ограничены интервалом с 8 до 12 ч. А строка «JHNUSMEM..JHNUSLBR» может использоваться для обозначения типичного плавательного сезона на Среднем Западе, продолжающегося от Дня памяти до Дня труда.

Таблица В.47 — Домен сокращений выражений типа данных GTS

| Код | Определение | Формальное определение |
|---|---|---|
| AM | Каждое утро во время, заданное стороной | H00..11 IST |
| PM | Каждая середина дня во время, заданное стороной | H12..23 IST |
| BID | Два раза в день во время, заданное стороной | /(12 h) IST |
| TID | Три раза в день во время, заданное стороной | /(8 h) IST |
| QID | Четыре раза в день во время, заданное стороной | /(6 h) IST |
| JB | Обычные рабочие дни (с понедельника по пятницу, исключая праздники) | J1..5 \JH |
| JE | Обычные выходные дни (суббота и воскресенье, не считая праздников) | J6..7 |
| JH | Праздники | |
| GTSAbbreviationHolidaysChristianRoman | Сокращения для римских христианских праздников | |

*Окончание таблицы В.47*

| Код | Определение | Формальное определение |
|---|---|---|
| JHCHRXME | | M1224 |
| JHCHRXMS | | M1225 |
| JHCHRNEW | | M0101 |
| JHCHREAS | | |
| JHCHRGFR | | |
| JHCHRPEN | | |
| JHNUS | Национальные праздники США (праздничные дни, установленные для федеральных служащих Федеральным законом США 5 U.S.C. 6103) | |
| JHNUSMLK | День Мартина Лютера Кинга (третий понедельник января) | M0115..21 J1 |
| JHNUSPRE | День рождения Вашингтона (День Президента), третий понедельник февраля | M0215..21 J1 |
| JHNUSMEM | День памяти (последний понедельник мая) | M0525..31 J1 |
| JHNUSMEM5 | Пятница недели перед Днем памяти | M0522..28 J5 |
| JHNUSMEM6 | Суббота недели перед Днем памяти | M0523..29 J6 |
| JHNUSIND | День независимости (4 июля) | M0704 |
| JHNUSIND5 | Пятница, непосредственно предшествующая 4 июля [статья (b) Федерального закона 5 U.S.C. 6103] | M0703 J5 |
| JHNUSIND1 | Понедельник, непосредственно следующий за 4 июля [статья (b) Федерального закона 5 U.S.C. 6103] | M0705 J1 |
| JHNUSLBR | День труда (первый понедельник сентября) | M0901..07 J1 |
| JHNUSCLM | День Колумба (второй понедельник октября) | M1008..14 J1 |
| JHNUSVET | День ветеранов (11 ноября) | M1111 |
| JHNUSTKS | День благодарения (четвертый четверг ноября) | M1122..28 J4 |
| JHNUSTKS5 | Пятница после Дня благодарения | M1123..29 J5 |

Примечания

1 Эта таблица не полна. Она не включает нехристианские религиозные праздники (установленные григорианской (западной) традицией) или национальные праздники, установленные за пределами США. Эти ограничения могут быть устранены с помощью постепенного пополнения таблицы.

2 Праздники имеют местную специфику. Какие именно религиозные праздники отмечаются под сокращением JH, зависит от местной и других традиций. Для обеспечения глобальной интероперабельности вместо именованных сокращений праздников безопаснее использовать специально сконструированные литералы GTS. Однако некоторые праздники, зависящие от фаз луны (например, Пасха), или декретированные праздники не могут быть простым образом представлены литералом типа данных GTS.

### B.6 Справочные типы данных

Эти типы данных в настоящее время отнесены к числу справочных, пока не будут разрешены известные вопросы, связанные с их конструированием.

#### B.6.1 Тип данных параметрического распределения вероятности PPD (специализация типа данных T)

Определение: параметризованное расширение типа данных, описывающее неопределенность количественных значений с помощью функции распределения и ее параметров. Помимо специфичных параметров распределения всегда определяются среднее значение (математическое ожидание) и стандартное отклонение, что обеспечивает минимальный уровень интероперабельности в случае, когда приложения-получатели не могут обеспечить обработку определенного распределения вероятности.

Таблица В.48 — Сводка свойств типа данных PPD<T>

| Имя | Тип | Описание |
|---|---|---|
| standardDeviation | QTY | Основная мера вариабельности/неопределенности значения (квадратный корень из суммы квадратов всех разностей значений данных и математического ожидания). Свойство standardDeviation (стандартное отклонение) используется для нормализации данных в целях вычисления функции распределения. По свойству standardDeviation приложения, которые не могут обеспечить обработку функции распределения, могут получить определенную информацию об уровне доверия к данным |
| distributionType | CE | Код, указывающий тип распределения вероятности. Возможные значения показаны в таблице В.49. Пустое значение кода (с причиной пустоты «unknown» — неизвестно) означает, что функция распределения вероятности неизвестна. В этом случае значение свойства standardDeviation означает эмпирическую догадку |

```
template<QTY T>
type ParametricProbabilityDistribution<T> alias PPD<T> specializes T {
   QTY      standardDeviation;
   CS       distributionType;
   IVL<T>   confidenceInterval(REAL p);
   REAL     probability(IVL<T> x);
   PPD<T>   times(REAL x);
};
```

Например, наиболее распространенным вступительным экзаменом в высшие учебные заведения США является SAT, состоящий из двух разделов: знание языка и математика. Каждая из них имеет минимальную оценку 400 (ни одного правильного ответа на вопросы) и максимальную 800. В 1998 г., согласно данным организации College Board, в этом тесте приняли участие 1 172 779 абитуриентов. Средняя оценка по математическому разделу составила 512, а стандартное отклонение — 112. Нормальное распределение со значениями параметров (512, 112) очень хорошо согласуется с реальным распределением оценок, полученных участниками тестирования. В большинстве случаев нет необходимости указывать свыше миллиона результатов, когда вместо них можно использовать только 2 параметра!

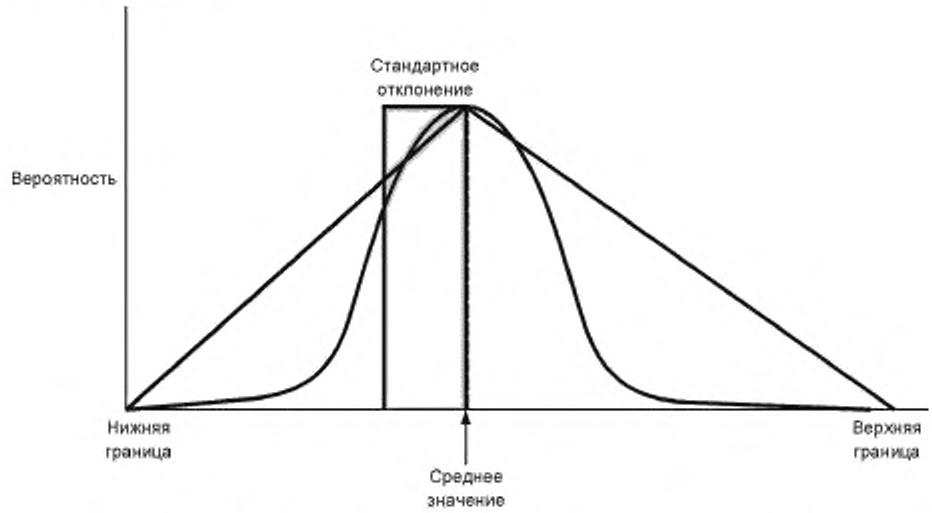


Рисунок В.18 — Пример параметрического распределения вероятности

Следует учесть, что нормальное распределение является только одним из нескольких распределений, используемых в стандартах HL7.

Так как тип данных PPD является специализацией типа T, то простое значение T является средним значением (математическим ожиданием) распределения вероятности. Если приложения не могут обеспечить обработку

распределения вероятности, то они могут воспользоваться простым значением T и пренебречь его неопределенностью. Это простое значение T также используется для стандартизации данных при вычислении распределения.

Распределения вероятности определены для целых или вещественных чисел и нормализованы по отношению к некоторой точке отсчета (обычно нуль) и эталонной единице (например, стандартное отклонение standardDeviation = 1). Когда в настоящей спецификации в качестве базовых типов используются другие величины, то для масштабирования распределения вероятности используются среднее значение и стандартное отклонение standardDeviation. Например, если величина типа PPD<PQ> представляет собой длину со средним значением 20 футов и стандартным отклонением standardDeviation, равным 2 дюймам, то нормализованная функция распределения $f(x)$, отображающая вещественное число $x$ на плотность вероятности, должна быть преобразована в функцию $f'(x')$, отображающую длину $x'$ на плотность вероятности по формуле $f'(x') = f((x' - \mu) / \sigma)$.

По возможности тип данных PPD соответствует требованиям, изложенным в документе «ISO Guide to the Expression of Uncertainty in Measurement» (GUM) (руководство по представлению неопределенности измерений) и отраженным в публикации организации NIST «1297 Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results». Тип данных PPD описывает только то, как представляется неопределенность, а не как она вычисляется. Понятие «стандартной неопределенности», введенное в документе ISO GUM, соответствует свойству standardDeviation.

B.6.1.1 Свойство standardDeviation: QTY

Основная мера вариабельности/неопределенности значения (квадратный корень из суммы квадратов всех разностей значений данных и математического ожидания). Свойство standardDeviation используется для нормализации данных в целях вычисления функции распределения. По свойству standardDeviation приложения, которые не могут обеспечить обработку функции распределения, могут получить определенную информацию об уровне доверия к данным.

Свойство standardDeviation является специализацией типа данных QTY (произведенного от типа данных T.diffType), представляющего разности значений типа T. Если в качестве T выступают типы данных REAL или INT, то свойство T.diffType также имеет значение REAL или INT соответственно. Но если тип данных T равен TS, то тип данных T.diffType является типом данных физической величины PQ с размерностью времени.

```
invariant(PPD x) {
   x.standardDeviation.dataType.implies(T.diffType);
};
```

B.6.1.2 Свойство distributionType: CE

Определение: код, указывающий тип распределения вероятности. Возможные значения показаны в таблице B.49. Пустое значение кода (с причиной пустоты unknown — неизвестно) означает, что функция распределения вероятности неизвестна. В этом случае значение свойства standardDeviation означает эмпирическую догадку.

Определенные распределения вероятности перечислены в таблице B.49. Многие типы распределений определены в терминах специальных параметров (например, $\alpha$ и $\beta$ в $\gamma$-распределении, число степеней свободы в t-распределении и т. д.). Однако при этом для всех типов распределений определены среднее значение и стандартное отклонение.

Таблица B.49 — Домен значений свойства distributionType

| Код | Имя | Определение |
|---|---|---|
| (NULL) | | Пустое значение кода указывает, что среднее значение оценивалось без существенного учета распределения вероятности. В этом случае значение стандартного отклонения не имеет строго обоснованной интерпретации, но можно исходить из того, что принято для нормального распределения, а именно, что интервал «среднее значение ± одно стандартное отклонение» имеет доверительный уровень, примерно равный двум третям |
| U | Равномерное | Равномерное распределение присваивает одинаковую плотность вероятности значениям, принадлежащим определенному интервалу, и нулевую плотность всем остальным значениям. Ширина интервала составляет $2\sigma\sqrt{3}$. Таким образом, при равномерном распределении присваивается плотность вероятности $f(x) = (2\sigma\sqrt{3})^{-1}$ значениям в интервале $\mu - \sigma\sqrt{3} \geq x \leq \mu + \sigma\sqrt{3}$ и $f(x) = 0$ остальным значениям |
| N | Нормальное (Гауссово) | Это хорошо известное распределение вероятности с плотностью в форме колокола. В силу центральной предельной теоремы теории вероятностей нормальное распределение имеет неограниченные случайные значения, являющиеся результатом многих случайных процессов. Нормальное распределение является достаточно точным даже для значений, ограниченных с одной стороны (например, больших 0), если в терминах стандартных отклонений среднее значение находится «достаточно далеко» от границы шкалы |

*Окончание таблицы В.49*

| Код | Имя | Определение |
|---|---|---|
| LN | Логарифмически-нормальное | Логарифмически-нормальное распределение используется для преобразования экспоненциально растущей случайной переменной в нормально распределенную случайную переменную U = log *X*. Для логарифмически-нормального распределения могут быть заданы среднее значение $\mu$ и стандартное отклонение $\sigma$. Эти значения являются параметрами исходного распределения, а не преобразованными параметрами нормального распределения, которые традиционно имеют те же самые буквенные обозначения. Эти логарифмически-нормальные параметры $\mu_{log}$ и $\sigma_{log}$ связаны со средним значением $\mu$ и стандартным отклонением $\sigma$ значений, данных по формулам $\sigma_{log}^2 = \log(\sigma^2/\mu^2 + 1)$ и $\mu_{log} = \log \mu - \sigma_{log}^2/2$ |
| G | γ (гамма) | Гамма-распределение (γ) используется для данных с наклонным графиком и правой границей, при которых максимум кривой распределения находится возле начала координат. Оно имеет два параметра $\alpha$ и $\beta$. Связи со средним значением $\mu$ и отклонением $\sigma^2$ задаются формулами $\mu = \alpha\,\beta$ и $\sigma^2 = \alpha\,\beta^2$ |
| E | Экспоненциальное | Используется для данных, описывающих вымирание. Экспоненциальное распределение является частным случаем гамма-распределения, у которого $\alpha = 1$, следовательно, для вычисления среднего значения $\mu$ и отклонения $\sigma^2$ справедливы формулы $\mu = \beta$ и $\sigma^2 = \beta^2$ |
| X2 | χ | Используется для описания суммы случайных переменных, возникающих при оценке (а не измерении) отклонения от образца. Распределение $\chi^2$ имеет единственный параметр υ, так называемое число степеней свободы (равное числу независимых слагаемых). Это распределение является частным случаем гамма-распределения, у которого $\alpha = \upsilon/2$ и $\beta = 2$, следовательно, для вычисления среднего значения $\mu$ и отклонения $\sigma^2$ справедливы формулы $\mu = \upsilon$ и $\sigma^2 = 2\upsilon$ |
| T | t (распределение Стьюдента) | Используется для описания отношения нормальной случайной переменной к квадратному корню от случайной переменной, имеющей распределение вероятности $\chi^2$. Распределение Стьюдента имеет один параметр υ (число степеней свободы). Его связь со средним значением $\mu$ и отклонением $\sigma^2$ определяется формулами $\mu = 0$ и $\sigma^2 = \upsilon/(\upsilon - 2)$ |
| F | F | Используется для описания отношения двух случайных переменных, имеющих распределение вероятности $\chi^2$. F-распределение имеет два параметра $\upsilon_1$ и $\upsilon_2$, которые являются числами степеней свободы соответственно числителя и знаменателя. Связь этих параметров со средним значением $\mu$ и отклонением $\sigma^2$ определяется формулами $\mu = \upsilon_2/(\upsilon_2 - 2)$ и $\sigma^2 = (2\,\upsilon_2(\upsilon_2 + \upsilon_1 - 2))/(\upsilon_1(\upsilon_2 - 2)^2(\upsilon_2 - 4))$ |
| B | β (beta) | Бета-распределение используется для данных, ограниченных с обоих сторон, график которых может быть, а может и не быть наклонным (что встречается при оценке вероятности). Для задания кривой используются два параметра $\alpha$ и $\beta$. Связь этих параметров со средним значением $\mu$ и отклонением $\sigma^2$ определяется формулами $\mu = \alpha/(\alpha + \beta)$ и $(\sigma^2 = \alpha\,\beta/((\alpha + \beta)^2(\alpha + \beta + 1))$ |

Обработка трех типов распределений, а именно неизвестное (NULL), равномерное и нормальное, должна обеспечиваться каждой системой, объявившей о поддержке типа данных PPD. Все остальные типы распределений необязательны. Если система, интерпретирующая представление значения типа PPD, обнаруживает тип распределения, который она не может распознать, то она должна отобразить его на тип распределение «неизвестное» (NULL).

В.6.1.3 Литеральная форма

Общий синтаксис литерала значения типа PPD задается следующим образом:

```
PPD<T>.literal ST {
   PPD<T>  : T "(" type QTY ")"  { ((T)$).equal($1);
                                   $.distributionType.equal($3);
                                   $.standardDeviation.equal($4); };
   CV type : ST                  { $.value.equal($1);
                                   $.codeSystem.equal(2.16.840.1.113883.5.1020); };
};
```

***Примеры***

***1 Литерал значения типа PPD<REAL> с нормальным распределением, средним значением 1.23 и стандартным отклонением 0.005 задается строкой «1.23(N0.005)».***

***2 Примером литерала значения типа PPD<PQ> служит строка «1.23 m (5 mm)», описывающая неизвестное распределение со средним значением 1.23 метра и стандартным отклонением 5 миллиметров.***

***3 Примером литерала значения типа PPD<TS> служит строка «2000041113(U4 h)», описывающая равномерное распределение со средним значением 13:00 1 апреля 2000 года и стандартным отклонением 4 часа.***

**B.6.2 Тип данных распределения вероятности вещественных значений PPD<REAL> (специализация типа данных PPD)**

```
type ParametricProbabilityDistribution<REAL> alias PPD<REAL>;
```

Параметрическое распределение вероятности вещественных значений полностью определяется специализируемым типом данных. Однако для литеральных представлений и преобразований распределения вероятности вещественных значений имеются некоторые специальные возможности, обсуждаемые в настоящем подразделе.

B.6.2.1 Преобразование вещественного значения (тип данных REAL) в неопределенное вещественное значение (тип данных PPD<REAL>)

При преобразовании вещественного значения (типа REAL) в значение типа PPD<REAL> стандартное отклонение PPD.standardDeviation вычисляется из порядка величины вещественного значения и из значения свойства REAL.precision (число значащих цифр). Пусть x — вещественное значение типа REAL, у которого свойство REAL.precision имеет значение n. Порядок величины e значения x можно определить по формуле e = log10 |x|, где e округляется до следующего целого числа, ближайшего к нулю (особый случай: если значение x равно нулю, то e равно нулю). Тогда значение наименьшей значащей цифры I равно $10^{e-n}$ и стандартное отклонение $\sigma$ (то есть значение свойства PPD.standardDeviation) определяется формулой $\sigma = I / 2$.

Таблица B.50 — Примеры стандартных отклонений, вычисленных по точности n и порядку величины e

| Представление | x | e | n | e - n + 1 | I | σ |
|---|---|---|---|---|---|---|
| 0 | 0 | (0) | 1 | 0 | 1 | 0.5 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0.5 |
| 2 | 2 | 0 | 1 | 0 | 1 | 0.5 |
| 9 | 9 | 0 | 1 | 0 | 1 | 0.5 |
| 10 | 10 | 1 | 2 | 0 | 1 | 0.5 |
| 100 | 100 | 2 | 3 | 0 | 1 | 0.5 |
| 1e+1 | 10 | 1 | 1 | 1 | 10 | 5 |
| 1e+2 | 100 | 2 | 1 | 2 | 100 | 50 |
| 10e+1 | 100 | 2 | 2 | 1 | 10 | 5 |
| 1.1 | 1.1 | 0 | 2 | –1 | 0.1 | 0.05 |
| 10.1 | 10.1 | 1 | 3 | –1 | 0.1 | 0.05 |
| 1.1e+2 | 110 | 2 | 2 | 1 | 10 | 5 |
| 1.1e-2 | 0.011 | –2 | 2 | –3 | 0.001 | 0.0005 |
| 1.1e-4 | 0.00011 | –4 | 2 | –5 | 0.00001 | 0.000005 |
| 10.1e-4 | 0.00101 | –3 | 3 | –5 | 0.00001 | 0.000005 |
| 0.1e-1 | 0.01 | –2 | 1 | –2 | 0.01 | 0.005 |
| 0.01e-1 | 0.001 | –3 | 1 | –3 | 0.001 | 0.0005 |
| 0.01e-2 | 0.0001 | –4 | 1 | –4 | 0.0001 | 0.00005 |
| 0.00 | 0 | (0) | 3 | –2 | 0.01 | 0.005 |

B.6.2.2 Краткая литеральная форма

Помимо общей литеральной формы типа PPD<T> для типа данных PPD<REAL> определена краткая литеральная форма. Она определена таким образом, что значение свойства стандартного отклонения PPD. standardDeviation может быть выражено в терминах наименьшей значащей цифры мантиссы. Этот литерал определен как расширения литерала типа данных REAL:

```
PPD<REAL>.literal ST {
   PPD<REAL> mantissa
           : REAL.mantissa "(" type QTY ")"  { ((T)$).equal($1);
                                               $.distributionType.equal($3);
                                               $.standardDeviation.equal($4); }
           | REAL.mantissa                   { $.equal($1);
                                               $.distributionType.equal($3);
                                               $.standardDeviation.equal(
                                              $1.leastSignificantDigit.times(0.5)); };
   CS type : ST                              { $.value.equal($1);
                                            $.system.equal(2.16.840.1.113883.5.1019);
};
};
```

***Пример — Строка «1.23e-3 (U5e-6)» общей литеральной формы обозначает равномерное распределение (PPD.distributionType = «U») с центром 1.23 × $10^{-3}$ и стандартным отклонением PPD. standardDeviation, равным 5 × $10^{-6}$. Краткая литеральная форма этого же значения имеет вид «1.230(U5) e-3».***

**B.6.3 Тип данных распределения вероятности физических величин PPD<PQ> (специализация типа данных PPD)**

Тип данных PPD<PQ> строится из типа данных PPD. Но поскольку свойство единиц измерения PQ.unit может быть выведено из границ интервала, то типу данных PPD<PQ> можно придать дополнительную семантику и определить для него отдельную литеральную форму. Для этого тип данных PPD<PQ> надо рассматривать как распределение вероятности вещественных значений с одной единицей измерения.

```
type ParametricProbabilityDistribution<PQ> alias PPD<PQ> {
   PPD<REAL> value;
   CS  unit;
};
```

Единица применяется как к среднему значению, так и к стандартному отклонению PPD.standardDeviation.

```
invariant(PPD<PQ> x)
      where x.nonNull {
   x.value.nonNull;
   ((REAL)x.value).equal(((PQ)x).value);
   x.unit.equal(((PQ)x).unit);
   x.value.standardDeviation.equal(x.standardDeviation.value);
   x.standardDeviation.unit.equal(x.unit);
};
```

B.6.3.1 Краткая литеральная форма

Краткая литеральная форма типа данных PPD<PQ> определяется на основе краткой литеральной формы типа данных PPD<REAL>, в которую подставляется значение физической величины (тип данных REAL). Этот литерал определяется как расширение литерала типа данных PQ.

```
PPD<PQ>.literal ST {
   PPD<PQ> : PPD<REAL> " " unit  { $.value.equal($1);
                                   $.unit.equal($3); }
};
```

***Примеры — Строка «1.23e-3 m (N5e-6 m)» общей литеральной формы описывает нормальное распределение длины с центром 1.23 × $10^{-3}$ м и стандартным отклонением PPD.standardDeviation, рав-***

***ным 5 × $10^{-6}$ м. Краткая литеральная форма этого же значения имеет вид «1.230(N5)e-3 m». Допустима также форма «1.23e-3(N0.005e-3) m»; она представляет собой краткую литеральную форму типа данных PPD<PQ>, скомбинированную с общей литеральной формой типа данных PPD<REAL>.***

**B.6.4 Тип данных распределения вероятности моментов времени PPD<TS> (специализация типа данных PPD)**

```
type ParametricProbabilityDistribution<TS> alias PPD<TS>;
```

Тип данных PPD<TS> полностью определяется специализируемым типом данных. Свойство стандартного отклонения PPD.standardDeviation имеет тип данных, равный значению свойства TS.diffType, а именно длительность (тип данных PQ с размерностью времени).

B.6.4.1 Преобразование значения типа TS в значение типа PPD<TS>

При преобразовании значения типа TS в значение типа PPD<TS> стандартное отклонение PPD.standardDeviation вычисляется из порядка значения типа TS и такой точности (число значащих цифр), при которой значение PPD.standardDeviations охватывает максимальный диапазон времени, описываемый не указанными цифрами. Например, в строке «20000609» не указанные цифры относятся к часам и более мелким единицам. В совокупности все эти цифры охватывают длительность 24 ч, поэтому стандартное отклонение PPD.standardDeviation равно 12 ч (от 20000609000000.0000... до 20000609999999.9999... (= 20000610).

Это правило отличается от указанного для типа данных REAL тем, что диапазон неопределенности лежит выше заданного момента времени. Это сделано, опираясь на суждение здравого смысла, согласно которому 9 июня охватывает все сутки 9 июня с центром в полудне, а не в полночи.

**Приложение С**
**(справочное)**

## Словарные домены HL7

Версия: 810-20090108
Ответственная рабочая группа: Vocabulary Work Group, Health Level Seven, Inc.
Сгенерировано: RoseTree 4.2.29, 2009-02-05T14:12:56
Дата последней публикации: 20090209 19:42

### С.1 Введение

**Примечание для голосования в ИСО**

Настоящий выпуск содержания словаря HL7 предназначен для обеспечения интерпретации проекта международного стандарта prEN ISO 27953 — Individual Case Safety Report (ICSR) (Индивидуальный отчет по безопасности лекарственного средства). Он отражает текущее состояние словаря для документа ICSR.

**С.1.1 Обзор**

При коммуникации и хранении данных в современном здравоохранении активно используется кодированная информация. В стандартах HL7 соответствующие справочники и классификаторы объединены общим понятием словаря. В этих стандартах определено несколько типов объектов, которые воплощают разные характеристики словаря. В то время как другие элементы стандартов HL7 в основном касаются структуры, в словарях основной акцент делается на содержании.

В соответствии с философией стандартов Версии 3, состоящей в последовательном ограничении абстрактной информационной модели, наименее ограниченной категорией словаря является словарный домен. [Пересмотреть словарь терминов и его связи]: словарный домен HL7 представляет собой именованную категорию сходных понятий (семантический тип), каждое из которых связано с одним или несколькими кодированными элементами. Словарные домены существуют в силу потребности ограничить назначение кодированного элемента, откладывая его привязку к конкретной кодированной терминологии на поздние этапы процесса разработки или реализации стандарта. Таким образом, словарные домены не зависят от какого-либо конкретного словаря терминов или системы кодирования.

Иерархическая категоризации словарных доменов позволяет далее ограничивать широту семантической категории, охватываемой словарным доменом. Такие ограниченные домены называются «поддоменами». Их применение позволяет далее специализировать (ограничить) значения, включенные в домен.

Список значений, включенных в домен понятий или в поддомен, называется набором значений (value set). Набор значений состоит из одного или нескольких кодированных понятий. Коды этих понятий могут передаваться в значениях кодированного типа данных, передаваемых в сообщениях стандарта HL7. Версии 3. Ассоциирование набора значений с данным словарным доменом понятий или поддоменом называется «связыванием». В разных обстоятельствах с одним и тем же словарным доменом могут быть связаны разные наборы значений.

Код понятия уникален только в определенном контексте. Например, в одном контексте код «М» может означать мужской пол (male), а в другом служить признаком состояния в браке (married). Контекст, в котором определено понятие, называется системой кодирования. В домене медицинских терминов уже существует большое число систем кодирования, например, МКБ-9, SNOMED-CT и т. д. По возможности в стандартах HL7 используются существующие коды понятий, а не конструируются новые.

В тот момент, когда разработчики сообщений и те, кто применяет эти сообщения на практике, примут решение об использовании конкретной терминологии для представления смыслового содержания кодированного элемента, набор значений становится связанным с кодированным элементом.

Словарные таблицы, определенные в стандарте HL7 для кодированных атрибутов классов, содержатся в словарном хранилище HL7, из которого могут извлекаться различные представления его содержания, позволяющие получить списки словарных значений, предназначенные для использования в Эталонной информационной модели HL7 RIM (Reference Information Model). Эти представления имеют табличный формат. К ним относятся словарные домены HL7, системы кодирования (как те, что ведутся комитетом HL7, так и внешние системы, используемые в стандартах HL7), наборы значений, определенные в стандартах HL7, а также таблицы перекрестных ссылок между словарными доменами и кодированными атрибутами.

В повествовательной части содержания модели RIM каждому кодированному атрибуту присвоены имя словарного домена и соответствующий квалификатор расширяемости. Эта спецификация указана в первой строке описания атрибута и имеет следующий формат:

«Словарный домен: MyConceptDomain (CWE)»

В тексте модели RIM имя словарного домена является гиперссылкой на соответствующую таблицу с описанием этого домена.

**С.1.2 Содержание**

Ниже представлено описание словаря, разработанного и привязанного к моделям. Из хранилища словаря HL7 извлечены три основные группы данных. Каждая группа состоит из указателя, элементы которого связаны с конкретными записями словаря. Каждая из этих записей представлена в табличной форме, специфичной для соответствующей словарной единицы. Каждый из приведенных ниже разделов содержит подраздел руководства, помещенный под указателем, детально описывающий графы таблиц, а также ключевые слова и соглашения по представлению информации, используемые для таблицы данной группы. Эти три раздела указателей описывают следующие группы данных:

- словарные домены HL7. Содержание этого раздела организовано в алфавитном порядке по имени домена. Указано также, имеются ли специализации домена в виде одного или нескольких поддоменов;
- вспомогательные системы кодирования HL7. Содержание этого раздела организовано в алфавитном порядке по имени системы кодирования. Указано также, является ли система кодирования «структурной»;
- наборы значений HL7. Существует большое число наборов значений, определенных в словаре HL7, и организация этого раздела отражает попытку улучшить представление простого алфавитного списка тысяч имен, многие из которых не являются простыми мнемоническими. Большинство наборов значений произведено исключительно или в основном от одной системы кодирования, поэтому указатель содержания раздела содержит список соответствующих систем кодирования, организованный в алфавитном порядке по имени системы кодирования. Если набор значений включает в себя коды из нескольких систем кодирования, то получить к нему доступ можно из записи указателя с именем любой из этих систем.

В каждом из этих указателей обеспечивается переход по гиперссылке на таблицу данных, содержащую информацию о соответствующем элементе индекса.

**С.2 Словарные домены HL7**

**С.2.1 Указатель словарных доменов**

Таблица С.1 содержит ручной «указатель» словарных доменов, определенных в стандартах HL7. В ней перечислены только домены верхнего уровня. Многие словарные домены являются специализациями этих доменов и могут быть найдены по гиперссылке от их предшественника. Все словарные домены, имеющие специализации, выделены ниже в таблице полужирным шрифтом. Более специфичные списки находятся в отдельном файле. Для каждого словарного домена в этом списке указаны:

- его текстовое определение;
- его специализации и генерализации, если таковые имеются;
- атрибуты модели RIM, для которых этот домен служит ограничением;
- наборы значений, основанные на этом домене.

Таблица С.1 — Указатель словарных доменов

| | | |
|---|---|---|
| AcknowledgementCondition | ElementName | ParticipationFunction |
| AcknowledgementDetailCode | EmployeeJob | ParticipationMode |
| AcknowledgementDetailType | **EmployeeJobClass** | ParticipationSignature |
| AcknowledgementType | EncounterAcuity | ParticipationType |
| ActClass | EncounterDischargeDisposition | PatientImportance |
| **ActCode** | EncounterReferralSource | PaymentTerms |
| ActMood | EncounterSpecialCourtesy | PersonDisabilityType |
| ActPriority | EntityClass | ProcedureMethod |
| ActReason | **EntityCode** | ProcessingID |
| ActRelationshipCheckpoint | EntityDeterminer | ProcessingMode |
| ActRelationshipJoin | EntityHandling | QueryPriority |
| ActRelationshipSplit | EntityRisk | QueryRequestLimit |
| **ActRelationshipSubset** | EntityStatus | QueryResponse |
| ActRelationshipType | Ethnicity | QueryStatusCode |

*Окончание таблицы С.1*

| | | |
|---|---|---|
| ActSite | ExposureMode | Race |
| ActStatus | GenderStatus | RelationalName |
| ActUncertainty | HL7StandardVersionCode | RelationalOperator |
| AdministrativeGender | HumanLanguage | RelationshipConjunction |
| AttentionKeyword | InvoiceElementModifier | ReligiousAffiliation |
| AttentionLineValue | JobTitleName | ResponseLevel |
| BatchName | LanguageAbilityMode | ResponseModality |
| CaseDetectionMethod | LanguageAbilityProficiency | ResponseMode |
| CaseDiseaseImported | ListOwnershipLevel | RoleClass |
| CaseTransmissionMode | LivingArrangement | RoleCode |
| CommunicationFunctionType | LocalRemoteControlState | RoleLinkType |
| Confidentiality | ManagedParticipationStatus | RoleStatus |
| ContainerCap | ManufacturerModelName | RouteOfAdministration |
| ContainerSeparator | MaritalStatus | SQLConjunction |
| ContentProcessingMode | MaterialForm | Sequencing |
| ContextControl | MessageWaitingPriority | SoftwareName |
| Currency | ModifyIndicator | SpecialArrangement |
| DeviceAlertLevel | ObservationInterpretation | SubstitutionCondition |
| DocumentCompletion | ObservationMethod | TargetAwareness |
| DocumentStorage | ObservationValue | |
| EducationLevel | OrganizationIndustryClass | |

**С.2.2 Руководство по таблицам словарных доменов**

По каждой ссылке в приведенном выше индексе можно перейти к таблице, описывающей соответствующий домен понятий. Для каждого домена понятий сгенерирована одна отдельная таблица. В таблице домена понятий одна строка повторяет имя домена, указанное в указателе, и для каждого поддомена, если такие существуют, приведены отдельные строки. В тех случаях, когда для дальнейшего ограничения поддомена определен дополнительный поддомен, эта практика продолжается для более глубоких уровней вложенности. Графы таблицы домена содержат информацию о значениях, входящих в домен; в приведенном ниже описании эти графы пронумерованы. Заголовки граф показаны в скобках.

1. Уровень (Lvl)

В этой графе указан уровень поддомена, описываемого данной строкой. В первой строке указан уровень поддомена «0», означающий, что эта строка соответствует родительскому домену, указанному в индексе доменов понятий. Уровнем «1» обозначен поддомен родительского поддомена, уровнем «2» — поддомен этого поддомена и т. д.

2. Имя и связь (Concept Domain Name Value Set Binding)

В этой графе указано имя домена понятия и его контексты связей, если таковые имеются. Имя набрано обычным текстом, а имя набора значений — курсивом. В скобках непосредственно за именем набора значений указаны контексты связи, если таковые имеются. Если для набора значений домена контекст не классифицирован, то используется обозначение (nos); оно служит указанием, что связь все еще обсуждается в уполномоченной рабочей группе и пока еще не включена в общую совокупность связей. Имя набора значений представляет собой гиперссылку на строку набора значений, включенную в таблицу наборов значений. Следует обратить внимание, что для некоторых записей объектный идентификатор (ОИД) набора значений выделен курсивом и заключен в квадратные скобки; это показывает, что для этого набора значения еще не определены. Обратите также внимание, что в большинстве из этих случаев имя набора значений совпадает с именем домена понятия.

3. Связь с доменом атрибута (атрибут модели RIM)

В этой графе указан атрибут модели RIM, тип данных и квалификатор расширяемости, связанные с доменом понятий. Имя атрибута модели RIM представляет собой гиперссылку на описание данного атрибута в модели RIM, приведенное в настоящей спецификации. Первый элемент в скобках означает тип данных, которым в каждом случае является тип данных «CD» или одна из его специализаций, предназначенные для описания кодированных данных в Версии 3. Второй элемент в скобках означает квалификатор расширяемости («CNE» или «CWE»). Эти два элемента разделены символом косой черты («/»). Следует обратить внимание, что в большинстве случаев атрибут указан только для домена понятия уровня «0»; большинство поддоменов получены в процессе ограничения модели и связаны с атрибутами, которые являются ограничениями атрибута модели RIM. Также следует учесть, что в большинстве случаев домен понятий связан только с одним атрибутом модели RIM.

4. Документация (Definition/Description)

Эта графа содержит все документирование, относящееся к данному домену понятий. Оно включает в себя определение домена, а также обсуждение или описание соответствующего класса понятий. Следует обратить внимание, что в некоторых случаях указано только «Требуется добавить описание», что понятно без разъяснений. В других случаях ячейка таблицы в этой графе пуста, что означает отсутствие документации для данного домена понятий. Описание может содержать ограничения, примеры и другие компоненты документации, полезные для понимания значения и использования данного домена понятий.

### С.3 Системы кодирования, поддерживаемые в стандартах HL7

#### С.3.1 Указатель систем кодирования

Таблица С.2 содержит ручной «указатель» систем кодирования, поддерживаемых в стандартах HL7. Ее строки упорядочены по именам систем кодирования. В ней показаны все системы кодирования, идентифицированные в словарном хранилище HL7. Некоторые из этих систем разработаны и сопровождаются комитетом HL7, для них приводится полное содержание. Их имена выделены в таблице полужирным шрифтом. Несколько внешних систем кодирования, содержание которых распространяется комитетом HL7 по соглашениям с их авторами, также полностью приведены. Каждая из этих внешних систем кодирования представлена в отдельном html-файле. Доступ к ней можно получить по гиперссылке с ее именем. Оставшиеся (внешние) системы кодирования перечислены в отдельном документе, их содержание не приводится. Определения различных систем кодирования могут содержать ссылки на любую из этих систем кодирования, независимо от того, находится ли она в ведении комитета HL7 или иной организации (см. следующий подраздел).

Таблица С.2 — Индекс систем кодирования

| | | |
|---|---|---|
| **Acknowledgement Detail Type** | **EntityClass** | **QueryResponse** |
| **AcknowledgementCondition** | **EntityCode** | **QueryStatusCode** |
| **AcknowledgementType** | **EntityDeterminer** | **RelationalOperator** |
| **ActClass** | **EntityStatus** | **RelationshipConjunction** |
| **ActCode** | **HL7StandardVersionCode** | **ResponseLevel** |
| **ActMood** | ISO 3166 Part 1 Country Codes | **ResponseModality** |
| **ActReason** | **ManagedParticipationStatus** | **ResponseMode** |
| **ActRelationshipCheckpoint** | **ModifyIndicator** | **RoleClass** |
| **ActRelationshipJoin** | **NullFlavor** | **RoleCode** |
| **ActRelationshipSplit** | **ObservationValue** | **RoleLinkType** |
| **ActRelationshipSubset** | **OrderableDrugForm** | **RoleStatus** |
| **ActRelationshipType** | **ParticipationType** | **Sequencing** |
| **ActStatus** | **ProcessingID** | **TransmissionRelationshipTypeCode** |
| **CommunicationFunctionType** | **ProcessingMode** | |
| **ContextControl** | **QueryPriority** | |

**С.3.2 Руководство по таблицам систем кодирования**

Каждая из гиперссылок в приведенном выше указателе указывает на таблицу с содержанием систем кодирования, архивированным в словарном хранилище HL7. Строки указателя упорядочены по именам систем кодирования. Содержание систем кодирования, имена которых выделены полужирным шрифтом, берется из словарного хранилища HL7. Это означает, что эти системы кодирования либо ведутся и сопровождаются комитетом HL7 с использованием процессов гармонизации и утверждения, либо зеркально отражены в словарном хранилище HL7 для удобства использования. К другим системам кодирования относятся те, которые ведутся внешними организациями, и часть их содержания используется здесь для различных связей в моделях и доменах понятий. Содержание таких систем не отражено в словарном хранилище HL7; для получения копии содержания следует обратиться к издателям соответствующей терминологии.

Каждая таблица с содержанием системы кодирования содержит четыре графы; имена граф и детали их содержания описаны ниже; имена граф указаны в скобках. Каждая строка таблицы описывает один код понятия (кодированный термин).

1. Уровень иерархии и возможность выбора (Lvl-Typ)

Первая часть данных в этой графе представляет собой целое число, указывающее глубину иерархии кода, описанного в данной строке. За ним следуют дефис и одна из следующих букв верхнего регистра:

- L (Leaf — лист); термин, не имеющий потомков в иерархии специализации, который может быть выбран в инструментальных средствах HL7 и поэтому рассматривается как лист;
- A (Abstract — абстрактный); термин, имеющий потомков в иерархии специализации, но который сам по себе не может быть выбран в инструментальных средствах HL7 и поэтому рассматривается как абстрактный;
- S (Specializable — специализируемый); термин, имеющий потомков в иерархии специализации, который может быть выбран в инструментальных средствах HL7 и поэтому рассматривается как специализируемый.

2. Код/понятие и ссылка на главный код набора значений (Concept Code Head Code-defined Value Set)

Эта графа содержит строковое значение кода понятия. В тех случаях, когда этот код используется как «главный код» набора значений, определенного как «главный код и все его потомки», то имя этого набора значений указано в той же ячейке, что и код термина, только выделено курсивом и помещено под кодом. Имя набора значений является гиперссылкой на соответствующую таблицу с описанием набора значений. Для удобства восприятия иерархии на каждом уровне вложенности коду предшествует точка (символ с кодом 0x2E).

3. Имя кодированного понятия, предназначенное для вывода на экран (Print Name)

Это строка, ассоциированная с термином кодированного понятия и используемая как основное обозначение понятия на английском языке.

4. Определение кодированного понятия, свойства и отношения (Coded Concept Definition, Properties and Relationships)

В этой графе указаны определение кодированного понятия и все присвоенные ему свойства и связи. Свойства представляют собой дополнительные значения, которые описывают или классифицируют понятие. Отношения определяют семантические связи этого понятия с другими понятиями. Наиболее распространенными отношения понятий являются «Specializes» (является специализацией) и «Generalizes» (является обобщением), определяющие иерархии подтипов в системе кодирования. Следует учесть, что многие понятия не имеют таких определений.

**С.4 Наборы значений HL7**

**С.4.1 Указатель наборов значений**

Таблица С.3 содержит ручной «указатель» наборов значений, определенных в стандартах HL7. Поскольку общее число таких наборов значений превышает 1800, и при этом они не выстроены в иерархию и не имеют иного естественного порядка, то они произвольно сгруппированы по системам кодирования, на которых они основаны (некоторые наборы значений основаны на нескольких системах кодирования и поэтому появятся в списках столько раз, на скольких системах кодирования они основаны).

Группировки систем кодирования отражают также способ распределения определений наборов значений по html-файлам. Первая таблица этой группы представляет собой список тех систем кодирования, на основе которых определен хотя бы один набор значений. Из нее можно перейти в документ, перечисляющий конкретные наборы значений, основанные на данной системе кодирования.

Таблица С.3 — Системы кодирования, на основе которых определены наборы значений

| | | |
|---|---|---|
| Acknowledgement Detail Type | EntityClass | QueryResponse |
| AcknowledgementCondition | EntityCode | QueryStatusCode |
| AcknowledgementType | EntityDeterminer | RelationalOperator |
| ActClass | EntityStatus | RelationshipConjunction |
| ActCode | HL7StandardVersionCode | ResponseLevel |

*Окончание таблицы С.3*

| | | |
|---|---|---|
| ActMood | ISO 3166 Part 1 Country Codes | ResponseModality |
| ActReason | ManagedParticipationStatus | ResponseMode |
| ActRelationshipCheckpoint | ModifyIndicator | RoleClass |
| ActRelationshipJoin | NullFlavor | RoleCode |
| ActRelationshipSplit | ObservationValue | RoleLinkType |
| ActRelationshipSubset | OrderableDrugForm | RoleStatus |
| ActRelationshipType | ParticipationType | Sequencing |
| ActStatus | ProcessingID | TransmissionRelationshipTypeCode |
| CommunicationFunctionType | ProcessingMode | |
| ContextControl | QueryPriority | |

**С.4.2 Руководство по таблицам наборов значений**

Для просмотра наборов значений предусмотрено навигационное средство в виде таблицы С.3, в которой перечислены системы кодирования, содержание которых используется в наборах значений. Имена этих систем представляют собой гиперссылки, по которым можно перейти к вспомогательным файлам, содержащим все определения наборов значений, основанных на конкретной системе кодирования. Для системы кодирования, по которой определен единственный набор значений, страница, открываемая по гиперссылке, содержит информационную таблицу, определяющую этот набор значений. Если система кодирования используется в нескольких наборах значений, то в верхней части страницы, открываемой по гиперссылке, приводится таблица с упорядоченным по алфавиту списком имен наборов значений, использующих эту систему кодирования. Эти имена представляют собой гиперссылки, по которым можно перейти к таблице с определением соответствующего набора значений.

Блок информации, описывающий набор значений, начинается с имени набора значений, за которым в квадратных скобках указан его ОИД. Затем приводится краткая сводка систем кодирования, значения которых включены в данный набор значений. Эта сводка пронумерована; каждому имени дополнительной системы кодирования, часть содержания которой включена в набор значений, присвоен порядковый номер. За этим номером следует имя системы кодирования, являющееся гиперссылкой на таблицу с описанием системы кодирования, переход к которой обеспечен также из предшествующего раздела. После имени системы кодирования в квадратных скобках указан ее ОИД.

Затем перечисляются контекстные связи (если таковые имеются). Раздел контекстных связей начинается строкой «Context Bindings to concept domain(s):» (контекстные связи с доменами понятий), под которой следует перечисление связей (по одной связи в строке). Каждое описание связи содержит имя домена понятия (являющееся гиперссылкой на соответствующую таблицу с описанием домена понятия), за которым следуют указание «in Realm:» (в среде) и имя той среды, для которой определена эта связь. В случае, если среда не классифицирована, то есть решение о связи все еще рассматривается уполномоченной рабочей группой, вместо имени среды указывается слово «undetermined» (неопределенная).

Ниже указан состав описания набора значений, представляемого его авторами (некоторые наборы значений не имеют описаний, которые в будущем должны быть добавлены).

Вслед за описанием набора значений приводится таблица, содержащая определение его содержания. У этой таблицы пять граф, показывающих информацию о каждом утверждении, являющемся компонентом определения набора значений и занимающем отдельную строку. Определение содержания представлено в виде группировок, которые должны сделать его более понятным. Это определение состоит из списка «блоков содержания», каждый из которых имеет один и тот же тип формата. Блок содержания может также содержать дополнительный блок содержания с тем же типом формата, который, в свою очередь, может содержать другой блок содержания; таким образом, определение является рекурсивным, как и модель набора значений, которой оно должно соответствовать. Правильно определенный набор значений не имеет циклов в своей рекурсивной структуре.

Таблица, содержащая определение набора значений, аккуратно отображает эту рекурсивную структуру, но при этом, естественно, подвержена ограничениям, присущим представлению рекурсивных структур в виде линейной таблицы. Для некоторой компенсации этих ограничений используются специальные символы, отступы и гиперссылки. Ниже описаны ключевые слова, специальные символы, определения и другая информация, помещаемая в каждой из этих граф. Номера идентифицируют положение графы в таблице, а заголовки граф указаны в скобках.

a) Определение глубины вложения (Lvl)

Это целое число, указывающее глубину вложения утверждения, описывающего компонент набора значений. Оно помогает визуализировать такие конструкции, как вложенные наборы значений, включаемые транзитивные замыкания, операторы в конкретном блоке содержания и т. д. Нумерация глубины вложения всегда начинается с нуля, и соответствующая строка можно рассматриваться как «корень» определения набора значений. Эти номера можно использовать для идентификации каждого определения «блока содержания». Но эти номера обозначают уровень вложения рекурсивной структуры и ссылок; конкретный блок содержания распространяется на несколько номеров, поэтому его уровень вложения следует рассматривать как «больше nn», а не равным конкретному номеру.

b) Тип содержания (Content Type)

В этой графе указан тип утверждения, описывающего компонент определения набора значений. В ней могут быть указаны ключевые фразы, принадлежащие определенному набору и указывающие, каким образом содержание, объявленное в остальных столбцах, включается в набор значений. Некоторые из этих фраз являются флагами в утверждении, а некоторые — ссылками на множество включаемых кодов. Следует обратить внимание, что каждому элементу этой графы предшествует одна или несколько точек («.»), показывающих глубину вложения утверждения, по одной точке на каждый уровень. В этой графе могут быть указаны следующие ключевые фразы:

- content (содержание): это ключевой элемент набора значений; для тех наборов, что основаны на нескольких системах кодирования, в этом элементе указана первая из систем, участвующих в определении набора значений. Он всегда занимает первую строку таблицы и имеет уровень глубины «0» (нуль). Только одна строка таблицы может быть помечена таким ключом. Следующие строки представляют собой определение первого раздела содержания. Первая строка с таким ключом служит заголовком первого раздела содержания. Следует учесть, что определение набора значений рекурсивно, поэтому блоки содержания могут входить в состав других блоков. Поэтому блок содержания, помеченный этим ключевым словом, является определением всего набора значений, все остальные определения «вложены» в него. Каждый конкретный экземпляр содержания может содержать дочерние элементы только одного из следующих пяти типов, и только один из этих типов, а именно codeBasedContent, допускает повторы. (Имейте в виду, что этот ограниченный список пяти подтипов распространяется также на элементы типа unionWithContent. Типы intersectionWithContent и excludedContent обсуждаются ниже при описании типа combinedContent.);

1) codeBasedContent (содержание, основанное на коде): этот подтип указывает, что в набор значений должен быть включен конкретный код из системы кодирования. Идентификация системы кодирования, из которой берется этот код, указана в последней строке из числа тех, что имеют номер уровня на единицу меньше уровня текущей строки (указывающей «текущее» множество содержания системы кодирования); определения системы кодирования наследуются всеми строками блока содержания. Этот конкретный код указан в графе, озаглавленной «Primary Reference» (основная ссылка). Блок содержания может включать в себя произвольное число строк с ключевым словом codeBasedContent, предназначенных для явного включения отдельного кода в блок содержания. В этих строках указана точная идентификация кода. Хотя в каждой из этих строк указан один код из одной системы кодирования, указание в графе «Qualifiers/Identifiers» (квалификаторы/идентификаторы) свойства TransitiveClosure (транзитивное замыкание) может привести к заданию для одной строки коллекции кодов, являющейся поддеревом в иерархии кодов системы кодирования;

2) propertyBasedContent (содержание, основанное на свойстве): этот подтип служит заголовком для произвольного числа строк, каждая из которых представляет собой спецификацию включения или исключения, основанную на свойствах понятий или на свойствах кода, определенных для системы кодирования в словарном хранилище. Для каждой из этих строк в данной графе Content Type будет указано объявление элемента; такими объявлениями служат:

- includeWithConceptProperty (включить по свойству понятия): в набор значений включается множество кодов, получающееся с помощью выборки по значению свойства понятия. Идентификация этого свойства указана в графе «Primary Reference». Условие выборки указано в графе «Qualifiers/Identifiers»;
- excludeWithConceptProperty (исключить по свойству понятия): из набора значений исключается множество кодов, получающееся с помощью выборки по значению свойства понятия. Идентификация этого свойства указана в графе «Primary Reference». Условие выборки указано в графе «Qualifiers/Identifiers»;
- includeWithCodeProperty (включить по свойству кода): в набор значений включается множество кодов, получающееся с помощью выборки по значению свойства кода. Идентификация этого свойства указана в графе «Primary Reference». Условие выборки указано в графе «Qualifiers/Identifiers»;
- excludeWithCodeProperty (исключить по свойству кода): из набора значений исключается множество кодов, получающееся с помощью выборки по значению свойства кода. Идентификация этого свойства указана в графе «Primary Reference». Условие выборки указано в графе «Qualifiers/Identifiers».

Имейте в виду, что точный синтаксис условий выборки и аргументов, используемых в таких определениях, будет предложен позже (в настоящее время в словарном хранилище HL7 нет ни одного набора значений, в котором использовался бы такой подтип утверждения);

3) codeFilterContent (содержание на основе фильтрации кодов): этот подтип идентифицирует строку, в которой указано выражение, задающее коллекцию кодов. Это выражение указано в графе «PrimaryReference». В блоке

содержания может присутствовать только одна такая строка. Определение фильтра указано в этой же строке и представляет собой выражение в той форме, которая поддерживается системой кодирования. Оно показано в графе «Primary Reference», а за ним в скобках указан язык выражения. Если выражение, сохраненное в базе данных, не является допустимым для одного из обозначенных языков, то оно будет показано как «комментарий» на одном из языков и ему будет предшествовать фраза «invalid expression» (недопустимое выражение). В этом случае будет также зарегистрирован открытый вопрос по отношению к родительскому типу содержания;

4) valueSetReference (ссылка на набор значений): этот подтип указывает, что данное определение ссылается на другой набор значений. Имя включаемого набора значений указано в графе «Primary Reference», ОИД этого набора значений, заключенный в квадратные скобки, приводится в графе «Qualifiers/Identifiers». Имя включаемого набора значения представляет собой гиперссылку, по которой можно перейти к таблице с его описанием, находящейся в том же файле, что и данный набор значений. Имейте в виду, что иногда имя включаемого набора значений не является гиперссылкой; это означает, что включаемый набор имеет пустое содержание (висячая ссылка, неполное определение или иной вид ошибки). В этих случаях для включаемого набора значений нет и таблицы с определением содержания;

5) combinedContent (комбинированное содержание): этот подтип означает, что следующие за ним строки являются определениями дополнительного содержания, включаемого в набор значений. Каждая из строк, следующая за строкой этого подтипа, должна содержать утверждение одного из нескольких различных типов, которое должно быть применено для получения списка кодированных понятий, включаемых в набор значений. Каждая следующая строка с номером уровня, превышающим номер уровня строки с этим ключевым словом, содержит отдельные утверждения, определяющие содержание, являющееся частью данного блока содержания. Оно должно быть скомбинировано с кодами из заданной системы кодирования. В блоке содержания может быть не более одной строки такого подтипа; она служит заголовком блока содержания.

Когда в набор значений должен быть включен дополнительный блок содержания, начинающийся с ключевого слова «combinedContent», то во второй графе могут быть указаны дополнительные ключевые слова, указывающие, каким образом конкретный блок содержания может сочетаться с набором значений. Строка, непосредственно следующая за строкой с ключевым словом «combinedContent», должна содержать одно из следующих ключевых слов, указывающих, каким образом специфицируемое в ней содержание будет сочетаться с набором значений. Следует учесть, что в блоке содержания, озаглавленном строкой с ключевым словом «combinedContent», каждой из следующих строк, определяющих содержание и имеющих подтипы «codeBasedContent», «valueSetReference», «propertyBasedContent», «codeFilterContent» или описывающих вложенный блок «combinedContent», должна предшествовать строка, которая в первой графе содержит одно из трех следующих ключевых слов, указывающих, каким образом содержание, определенное следующей строкой, должно сочетаться для получения множества кодов, образующих результирующий набор значений:

- unionWithContent (объединение с содержанием): это ключевое слово означает, что содержание, определяемое следующей строкой, должно быть скомбинировано с предыдущим содержанием путем объединения. Термин «объединение» используется в теоретико-множественном смысле, поскольку набор значений обычно можно трактовать как «множество» в математике. Система кодирования, из которой берется содержание, обязательно должна быть указана в графе «Code System» одной из этих строк;
- intersectionWithContent (пересечение с содержанием): это ключевое слово означает, что содержание, определяемое следующей строкой, должно быть скомбинировано с содержанием, определяемым предыдущими строками, путем пересечения;
- excludeContent (исключить из содержания): это ключевое слово используется, когда надо исключить часть содержания из определяемого набора значений. Чаще всего это имеет место, когда включается «дерево» кодов (например, с помощью ключевого слова «codeBasedContent» и транзитивного замыкания), но вершину этого дерева включать не надо. Это ключевое слово можно также использовать в ситуации, когда произвольный список кодов должен быть исключен из коллекции кодов, которое было включено ранее с помощью операции объединения.

c) Система кодирования (Code System)

В этой графе указано имя системы кодирования, из которой должны быть взяты коды для блока содержания. Он заполняется в каждой строке с ключевым словом «content» или «unionWithContent» и должен быть пустым для других строк. Имя системы кодирования является гиперссылкой, по которой можно перейти к табличному описанию системы кодирования, находящемуся в соответствующем файле (см. предыдущий подраздел).

d) Специфичное вложенное содержание (Primary Reference)

В этой графе указан точный элемент, который должен использоваться для определения содержания набора значений. Для строк с ключевым словом «codeBasedContent» в этой графе указано значение кода из активной системы кодирования. Если оно содержит имя кода из системы кодирования, являющейся текущей для данного блока содержания, то это имя является гиперссылкой на конкретную строку с определением этого кода в табличном описании этой системы кодирования. Для строк с ключевым словом «valueSetReference» эта графа содержит имя набора значения, являющееся гиперссылкой на соответствующую строку в данном разделе таблиц с наборами значений. Для других типов строк эта графа может содержать имя свойства понятия или свойства кода. Для строк, не имеющих ссылок на словарный объект (например, строк с ключевым словом «combinedContent») или ссыла-

ющихся только на систему кодирования (указываемую в графе «Code System»), эта графа должна оставаться пустой.

е) Дополнительные данные (Qualifiers/Identifiers)

Эта графа содержит дополнительную информацию, используемую для определения набора значений. Обычно в ней указаны выражения и аргументы, используемые в сочетании с определением содержания, указанным в графе «Content Type», но в этой графе могут использоваться специфичные ключевые слова и фразы, представляющие определенные флаги управления, выставляемые в определении набора значений, а также некоторые другие типы определяющих выражений, которые могут быть добавлены к определению набора значений его авторами. Эта графа может содержать ОИД, заключенный в квадратные скобки и обычно присутствующий только в строках с ключевым словом «valueSetReference», хотя в большинстве случаев эта графа не имеет структурированного содержания. Одной из специфичных ключевых фраз является:

1) Relationship: Generalizes With: TransitiveClosure (отношение: генерализует транзитивное замыкание)

Она указывает, что для сбора совокупности кодов, задаваемой данной строкой, должен быть выполнен транзитивный обход иерархии, начиная с кода, указанного в графе «Primary Reference». Эта ключевая фраза используется только для строк с ключевым словом «codeBasedContent», она включается в ту же строку, что и заданный кодированный термин, и означает, что набор значений включает в себя все понятия, являющиеся потомками этого кодированного термина.

### С.5 Запрещенные элементы словаря HL7

#### С.5.1 Список запрещенных элементов

Процесс ведения содержания словаря HL7 предусматривает две стадии изъятия (в действительности удаления) таких элементов, как домены понятий, системы кодирования, понятия, включенные в систему кодирования, и наборы значений.

Если известно или предполагается, что элемент использовался для определения формальной спецификации HL7, например типа данных, документа или сообщения, то первой стадией его удаления является «запрещение» его дальнейшего использования для составления новых спецификаций. Тем самым пользователям запрещенного содержания дается предупреждение о необходимости изменения их спецификаций, позволяющего в будущем не зависеть от этого содержания.

На второй стадии запрещенное содержание фактически изымается или удаляется из определенного содержания словаря.

Чтобы дать точное представление о действующем содержании словаря, рабочие группы Vocabulary Work Group и Publishing Work Group договорились об исключении показа запрещенного материала (за исключением запрещенных понятий) в основном содержании словаря. Кроме того, чтобы это исключение было наглядным, они договорились предоставлять таблицу всех запрещенных доменов понятий, систем кодирования и наборов значений с указанием версии словаря, в которой начал действовать запрет. Эта таблица приведена ниже:

| Запрещенные словарные элементы | |
|---|---|
| Имя элемента | Выпуск, в котором запрещен впервые |
| Запрещенные наборы значений | |
| EntityDeterminerDescribedQuantified (2.16.840.1.113883.1.11.20053) | 589-20081114 |
| HL7TriggerEventCode (2.16.840.1.113883.1.11.19427) | 623-20081218 |

**Приложение ДА**
**(справочное)**

## Сведения о соответствии ссылочных международных стандартов национальным стандартам

Таблица ДА.1

| Обозначение ссылочного международного стандарта | Степень соответствия | Обозначение и наименование соответствующего национального стандарта |
|---|---|---|
| ISO/HL7 21731:2006 | IDT | ГОСТ Р ИСО/HL7 21731—2013 «Информатизация здоровья. HL7, Версия 3. Эталонная информационная модель. Выпуск 1» |
| HL7 2008 | — | * |

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:
– IDT — идентичный стандарт.

## Библиография

[1] Object Management Group IDL Syntax and Semantics chapter — OMG 2002 (http://www.omg.org/cgi-bin/doc?formal/02-06-39)
[2] Web Services Description Language (WSDL) 1.1 — W3C 2001 (http://www.w3.org/TR/wsdl)

УДК 004:61:006.354 ОКС 35.240.80 П85 ОКСТУ 4002

Ключевые слова: здравоохранение, информатизация здоровья, электронная передача данных

Корректор *Е.Р. Ароян*
Компьютерная верстка *Ю.В. Поповой*

Сдано в набор 01.12.2016. Подписано в печать 27.12.2016. Формат 60 × 84 $^{1}/_{8}$. Гарнитура Ариал.
Усл. печ. л. 50,7.

---

Набрано в ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
www.jurisizdat.ru y-book@mail.ru

Издано во ФГУП «СТАНДАРТИНФОРМ», 123995, Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru