ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 60.2.7.1— 2024/ ИСО 22166-201:2024

Роботы и робототехнические устройства

МОДУЛЬНЫЙ ПРИНЦИП ПОСТРОЕНИЯ СЕРВИСНЫХ РОБОТОВ

Часть 201

Общая информационная модель модулей

(ISO 22166-201:2024, Robotics — Modularity for service robots — Part 201: Common information model for modules, IDT)

Издание официальное

Москва
Российский институт стандартизации
2024

Предисловие

1 ПОДГОТОВЛЕН Федеральным государственным автономным научным учреждением «Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики» (ЦНИИ РТК) на основе собственного перевода на русский язык англоязычной версии стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 141 «Робототехника»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 12 ноября 2024 г. № 1656-ст

4 Настоящий стандарт идентичен международному стандарту ИСО 22166-201:2024 «Робототехника. Модульность сервисных роботов. Часть 201. Общая информационная модель модулей» (ISO 22166-201:2024 «Robotics — Modularity for service robots — Part 201: Common information model for modules», IDT).

Наименование настоящего стандарта изменено относительно наименования указанного международного стандарта для приведения в соответствие с ГОСТ Р 1.5—2012 (пункт 3.5) и для увязки с наименованиями, принятыми в существующем комплексе национальных стандартов Российской Федерации.

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты, сведения о которых приведены в дополнительном приложении ДА

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 ближайшем выпуске ежемесячного информационного указателя «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.rst.gov.ru)

Содержание

Введение

Требования стандартов комплекса ГОСТ Р 60 распространяются на роботы и робототехнические устройства. Целью стандартов является повышение интероперабельности роботов и их компонентов, а также снижение затрат на их разработку, производство и обслуживание за счет стандартизации и унификации процессов, интерфейсов, узлов и параметров.

Стандарты комплекса ГОСТ Р 60 представляют собой совокупность отдельно издаваемых стандартов. Стандарты данного комплекса относятся к одной из следующих тематических групп: «Общие положения, основные понятия, термины и определения», «Технические и эксплуатационные характеристики», «Безопасность», «Виды и методы испытаний», «Механические интерфейсы», «Электрические интерфейсы», «Коммуникационные интерфейсы», «Методы моделирования и программирования», «Методы построения траектории движения (навигация)», «Конструктивные элементы». Стандарты любой тематической группы могут относиться как ко всем роботам и робототехническим устройствам, так и к отдельным группам объектов стандартизации — промышленным роботам в целом, промышленным манипуляционным роботам, промышленным транспортным роботам, сервисным роботам в целом, сервисным манипуляционным роботам, сервисным мобильным роботам, а также к морским робототехническим комплексам.

Настоящий стандарт относится к тематической группе «Методы моделирования и программирования» и распространяется на все виды сервисных роботов.

Настоящий стандарт определяет общую информационную модель (ОИМ) модулей, входящих в состав сервисных роботов в соответствии с ИСО 22166-1. Используя данную ОИМ, модули можно легко объединять между собой и обеспечивать обмен данными между ними. Данная ОИМ разработана для повышения интероперабельности, возможности повторного использования и компонуемости модулей. ОИМ представляет собой метамодель, являющуюся базовой моделью для всех видов модулей, таких как аппаратные модули, модули с аппаратными и программными свойствами, программные модули и составные модули. Поэтому ОИМ предоставляет метахарактеристики, которыми может обладать модуль.

Изготовители и интеграторы модулей, разработчики и изготовители роботов, а также интеграторы робототехнических комплексов могут использовать ОИМ, чтобы упростить получение необходимых модулей, использовать разные модули в соответствии с требуемой функцией и бюджетом, а также разрабатывать новые (составные) модули на базе ОИМ. ОИМ может помочь в проектировании модулей, эксплуатации и техническом обслуживании сервисных роботов.

ОИМ содержит четыре вида информации:

- по идентификации модулей;
- выбору модулей;
- интеграции модулей;
- эксплуатации и техническому обслуживанию модулей.

ОИМ как метамодель состоит из нескольких субмоделей, к которым относятся «Идентификатор модуля», «Характеристики», «Входные и выходные переменные», «Состояние», «Сервисы», «Инфраструктура», «Уровень безопасности и защищенности», «Моделирование» и «Исполняемая форма».

Модель реализации ОИМ будет представлена в последующих стандартах серии ИСО 22166, которая применима к составным модулям и различным типам сервисных роботов.

Настоящий стандарт устанавливает требования и руководящие указания для информационной модели модулей сервисных роботов, соответствующей девяти принципам модульного построения, установленным в ИСО 22166-1.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Роботы и робототехнические устройства

МОДУЛЬНЫЙ ПРИНЦИП ПОСТРОЕНИЯ СЕРВИСНЫХ РОБОТОВ

Часть 201

Общая информационная модель модулей

Robots and robotic devices. Modular principle of service robots structure. Part 201. Common information model for modules

Дата введения — 2025—01—01

1 Область применения

Настоящий стандарт определяет требования и руководящие указания к общей информационной модели (ОИМ) модулей сервисных роботов для обеспечения интероперабельности, возможности повторного использования и компонуемости.

Настоящий стандарт определяет структуру ОИМ и детализирует ее использование по назначению, сущность атрибутов и подклассов.

Требования настоящего стандарта применимы к сервисным роботам.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты [для датированных ссылок применяют только указанное издание ссылочного стандарта, для недатированных — последнее издание (включая все изменения]:

ISO 22166-1:2021, Robotics — Modularity for service robots — Part 1: General requirements (Робототехника. Модульность сервисных роботов. Часть 1. Общие требования)

IETF RFC 4122, A Universally Unique Identifier (UUID) URN Namespace [Пространство имен URN универсального уникального идентификатора (UUID)]

IEEE/Open Group 1003.1-2017, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX™) Base Specifications, Issue 7 [Информационные технологии. Базовые спецификации интерфейса переносимой операционной системы (POSIX™), выпуск 7]

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями.

ИСО и МЭК поддерживают терминологические базы данных для использования в документах по стандартизации по следующим адресам:

- платформа онлайн-просмотра ИСО доступна по адресу http://www.iso.org/obp;
- Электропедия МЭК доступна по адресу http://www.electropedia.org.

3.1 **информационная модель;** ИМ (information model; IM): Абстракция и представление объектов в управляемой среде, их свойств, операций, а также способа, которым они связаны друг с другом.

[ИСО 22166-1:2021, пункт 3.1.11, модифицировано]

3.2 **общая информационная модель;** ОИМ (common information model; CIM): Информационная модель, которую модули чаще всего используют в сервисных роботах.

Издание официальное

3.3 **модуль** (module): Компонент (или сборка компонентов с заданными интерфейсами), сопровождаемый профилями характеристик и предназначенный для облегчения проектирования системы, интеграции, интероперабельности и повторного использования.

[ИСО 22166-1:2021, пункт 3.3.12]

3.4 **характеристика модуля** (module property): Атрибут или параметр модуля.

[ИСО 22166-1:2021, пункт 3.3.14]

3.5 **программный модуль** (software module): Модуль, реализация которого состоит исключительно из запрограммированных алгоритмов.

[ИСО 22166-1:2021, пункт 3.4.4]

3.6 **аппаратный модуль** (hardware module): Модуль, реализация которого состоит только из физических элементов, включая механические элементы, электронные схемы и любое программное обеспечение (например, встроенные микропрограммы), недоступное извне через коммуникационный интерфейс.

[ИСО 22166-1:2021, пункт 3.4.3]

3.7 **модуль с аппаратными и программными свойствами** (*аппаратно-программный модуль*) [module with hardware aspects and software aspects (*hardware-software module*)]: Модуль, реализация которого состоит из физических элементов, программного обеспечения и коммуникационного интерфейса, позволяющего обмениваться данными с другими модулями.

3.8 **менеджер модулей** (module manager): Реализует функции управления жизненным циклом экземпляров модулей, включая создание экземпляров модулей.

3.9 **экземпляр** (instance): Отдельный объект, созданный из программного модуля, или отдельный объект конкретного модуля с аппаратными свойствами.

Примечание 1 — В объектно-ориентированном программировании экземпляр представляет собой специфическую реализацию объекта.

Примечание 2 — В аппаратной реализации экземпляр является одним из одинаковых модулей, используемых в составном модуле.

3.10 **уровень эффективности защиты;** УЭЗ (performance level; PL): Дискретный уровень, используемый для определения способности элементов системы управления, связанных с обеспечением безопасности, осуществлять функцию безопасности в прогнозируемых условиях.

[ИСО 13849-1:2023, пункт 3.1.5]

3.11 **уровень полноты безопасности;** УПБ (safety integrity level; SIL): Дискретный уровень (принимающий одно из трех возможных значений), представляющий способность осуществлять функцию безопасности, при этом уровень 3 является высшим уровнем полноты безопасности, а уровень 1 — низшим.

[МЭК 62061:2021, пункт 3.11]

3.12 **исполняемая форма** (executable form): Форма программного кода, при которой программа или встроенная программа управляется и контролируется исключительно операционной средой модуля и не требует компиляции (например, отсутствие исходного кода, объектный код или код, компилируемый в нужное время).

Примечание 1 — Двумя основными видами исполняемой формы являются скомпилированные программы и скрипты.

Примечание 2 — Исполняемая форма может включать набор файлов и/или папок, которые составляют полный модуль, либо представлять собой единый файл.

[ИСО/МЭК 19790:2012, пункт 3.42]

3.13 **аппаратные свойства** (hardware aspects): Информация относительно параметров и функций, характеризующих модуль и его физические взаимосвязи, а также относительно допустимого диапазона физических параметров рабочей среды.

[ИСО 22166-1:2021, пункт 3.3.5]

3.14 **программные свойства** (software aspects): Информация о внешних программных характеристиках модуля и его интерфейса, а также о жизненном цикле выполнения функции данного модуля.

[ИСО 22166-1:2021, пункт 3.3.21]

4 Общая информационная модель модулей

4.1 Общие положения

Общая информационная модель модулей должна состоять из элементов, представленных в таблице 4.1. Имена характеристик и классов, используемых в информационной модели, должны соответствовать правилам присвоения имен, установленным в приложении А. В таблице 4.1 буквы «М» и «О» обозначают «обязательный (mandatory)» и «факультативный (optional)» соответственно. ОИМ, изображенная на рисунке 4.1, подробно описана в 4.3. Модели классов для ОИМ представлены согласно ИСО/МЭК 19505-1:2012. Информация, приведенная в таблице 4.1 и на рисунке 4.1, может быть использована на этапах проектирования, разработки, эксплуатации и технического обслуживания, а в приложении D представлено, на каком этапе используется информация, предоставленная информационной моделью.

Примечание 1 — ОИМ для управления общей информацией в таблице 4.1 представлена на рисунке 4.4.

Примечание 2 — В настоящем стандарте представление ОИМ на языке XML использовано только в качестве примеров, чтобы помочь понять значение атрибутов в модели классов.

Таблица 4.1 — Общая информация по модулям и соответствующим групповым меткам (использованным на рисунке 4.4)

Позиция	Элемент	Общая информационная модель[b]	Информационные модели модулей[c]			Имя соответствующей группы/тега (Аббревиатура для каждой группы)
			Программно-аппаратный модуль	Аппаратный модуль	Программный модуль	
1	Имя модуля	M	M	M	M	GenInfo
2	Описание	O	O	O	O	
3	Изготовитель	M	M	M	M	
4	Примеры	O	O	O	O	
5	Версия информационной модели	M	M	M	M	IDnType
6	Идентификатор модуля	M	M	M	M	
7	Аппаратные свойства	O	M	M	—	
8	Программные свойства	O	M	—	M	
9	Характеристики модуля[a]	M	M	M	M	Properties
10	Входы	O	O	O	O	IOVariables
11	Выходы	O	O	O	O	
12	Состояние	O	O	O	M	—
13	Сервисы (возможности)	O	M	O	M	Services
14	Инфраструктура	O	O	M	M	Infra
15	Безопасность/защищенность	O	M	O	M	SafeSecure
16	Моделирование	O	O	O	O	Modelling
17	Исполняемая форма	O	O	O	M	ExecutableForm

[a] Являются обязательными только для тех модулей, на которые может быть оказано влияние (которые могут быть установлены) извне или, по крайней мере, для тех, которые оказывают ожидаемое воздействие на другие модули.

[b] Все элементы обязательны для ОИМ.

[c] Для информационных моделей, таких как программно-аппаратные модули, аппаратные модули или программные модули, некоторые типы элементов могут быть опущены в зависимости от их функциональностей. В частности, информационные модели аппаратных модулей и программных модулей элементы «Программные свойства» и «Аппаратные свойства» не включают соответственно.

Элемент, расположенный в позиции номер 1 таблицы 4.1, «Имя модуля» соответствует имени, представляющем модуль. Далее вместо выражения «элемент, расположенный в позиции номер N таблицы 4.1» используется сокращенное выражение «элемент N».

Элемент 2 «Описание» соответствует общему описанию модуля, т. е. что он собой представляет, что он делает и как он используется.

Элемент 3 «Изготовитель» соответствует контактной информации проектировщика, разработчика или изготовителя модуля.

Элемент 4 «Примеры» соответствует типичным вариантам применения модуля.

Элемент 5 «Версия информационной модели» соответствует номеру версии информационной модели, которую определяет настоящий стандарт.

Элемент 6 «Идентификатор модуля» должен соответствовать уникальному идентификатору модуля в системе, как описано в приложении В. Идентификатор модуля содержит информацию о типе модуля (аппаратно-программный, аппаратный или программный) и является ли он базовым или составным модулем.

Элементы 7 «Аппаратные свойства» и 8 «Программные свойства» относятся только к составным модулям. Если модуль состоит из двух или более аппаратно-программных модулей, программных модулей и/или аппаратных модулей, то идентификаторы модулей указаны в элементе «Аппаратные свойства», если они являются аппаратными или аппаратно-программными модулями, иначе (если они являются программными модулями) идентификаторы модулей указаны в элементе «Программные свойства».

Элемент 9 «Характеристики модуля» соответствует значениям, обычно используемым при инициализации модулей. Характеристики модулей разделяют на обязательные и факультативные, и это должно быть отмечено. Взаимосвязь между модулями представлена данным элементом, подробное содержание которого будет определено в последующих стандартах, таких как ИСО 22166-202.

Примечание 3 — Информация о взаимосвязи между модулями может быть предоставлена по-разному в зависимости от типа модуля.

Примечание 4 — Ограничения внешней среды также считаются характеристиками, примерами которых являются рабочая температура, рабочая влажность и максимально допустимый механический удар. Параметры, связанные с поведением модуля, могут быть указаны в характеристиках. Например, каждый коэффициент, используемый в алгоритме ПИД (пропорциональный, интегральный, дифференциальный) регулятора, используется один раз при инициализации и изменяется и используется несколько раз в процессе выполнения соответствующих программных модулей.

Элементы 10 и 11 «Входы» и «Выходы» соответствуют именам переменных при передаче данных в модуль и/или из модуля.

Пример 1 — Входами и выходами программного модуля управления сервоприводами являются показания датчика положения и значения управляющих воздействий на двигатель соответственно.

Примечание 5 — Характеристики являются разновидностями входных значений с точки зрения модулей, но их отличие состоит в том, что входные значения относятся к внешней среде модуля, а характеристики связаны с параметрами самого модуля. Например, входами программного модуля управления сервоприводами являются показания датчика положения, а характеристиками являются коэффициенты P, I и D, соответствующие пропорциональной, интегральной и дифференциальной составляющим.

Элемент 12 «Состояние» соответствует состоянию модуля в процессе работы.

Примечание 6 — Состояние не используют на этапах проектирования и разработки.

Элемент 13 «Сервисы (возможности)» соответствует интерфейсам, которые модуль предоставляет и использует для сервисов робота.

Примечание 7 — Сервис означает выполнение одной или нескольких функций модуля для других модулей с помощью заранее определенного интерфейса.

Пример 2 — Примеры формата функции для программных свойств приведены в таблице 4.2. Типы данных, такие как int16 и uint8, для данного примера определены в приложении С, таблица С.5.

Таблица 4.2 — Примеры формата функции для программных модулей

Имя	Аргументы	Возвращаемое значение	Описание
initialize	Integer init_ival1, Real init_rval2	Integer	Инициализация с использованием двух аргументов Возвращаемое значение: (0: успех, отрицательное значение: тип ошибки)
	Integer init_ival1, Real init_rval2, Integer init_ival3	Integer	Инициализация с использованием трех аргументов Возвращаемое значение: (0: успех, отрицательное значение: тип ошибки)

Пример 3 — Пример формата функции для электрических/электронных свойств приведен в таблице 4.3, в которой аргументы «connType», «keying» и «busProtocol» означают тип соединителя, исполнение соединителя и тип протокола, который использует модуль. Данная функция используется для проверки того, что равноправный модуль использует надлежащие электрические свойства модуля. Значениями типа соединителя могут быть USB-A, RJ45, DB-9 и т. д. Значениями исполнения являются «штекер» или «гнездо». Значениями аргумента «busProtocol» могут быть USB, Ethernet, EtherCAT, RS232 и т. д.

Таблица 4.3 — Пример формата функции для проверки электрической/электронной применимости

Имя	Аргументы	Возвращаемое значение	Описание
checkElecConnectivity	String connType, String keying, String busProtocol	Boolean	Проверка электрической/электронной применимости с использованием трех аргументов Возвращаемое значение: (True: успех, False: ошибка)

Пример 4 — Пример формата функции для механических свойств приведен в таблице 4.4. Как и для электрических/электронных свойств, данная функция используется для проверки того, что равноправный модуль использует надлежащие механические свойства модуля. Однако в отличие от предыдущего примера формат функции для механических свойств может быть более сложным из-за большого разнообразия их использования на практике. Поэтому в данном примере использованы только две упрощенные категории — шарнир и звено.

Таблица 4.4 — Пример формата функции для проверки механической применимости

Имя	Аргументы	Возвращаемое значение	Описание
linkConnectivity	Real origin, Real mass, Real Inertia[], String shape, Real size, Real axis[], String connection, Real collision[]	Boolean	Проверка применимости механического звена с использованием 8 аргументов Возвращаемое значение: (True: успех, False: ошибка)
jointConnectivity	Real origin[], Real axis[], Real limits, Real damping, Real friction	Boolean	Проверка применимости механического шарнира с использованием 5 аргументов Возвращаемое значение: (True: успех, False: ошибка)

Элемент 14 «Инфраструктура» соответствует перечню аппаратного и/или программного обеспечения, которое модули обычно используют или с которыми соединяются.

Пример 5 — Примерами инфраструктуры являются тип энергоснабжения, тип межплатформенного программного обеспечения, тип шины данных и тип базы данных.

Элемент 15 «Безопасность/защищенность» соответствует связанному с безопасностью уровню эффективности защиты и информации о защищенности, обеспечиваемой модулем. В элементе «Безопасность/защищенность» обычного сервисного робота используется связанный с безопасностью уровень эффективности защиты, определенный в ИСО 13849-1, и указывается уровень, связанный с защищенностью, значение которого находится в диапазоне от 0 до 4. Уровень защищенности 0 означает, что в модуле вообще не предусмотрены меры по обеспечению защищенности. Уровни защищенности от 1 до 4 определены в МЭК 62443-4-2:2019. Однако для специфических типов роботов, таких как медицинские

роботы и роботы для оказания физической помощи, следует использовать другие стандарты, связанные с безопасностью и защищенностью. Уровень эффективности защиты обычно определяют для каждой отдельной функции безопасности. Если у модуля есть несколько функций безопасности, то такой модуль должен обеспечить уровень эффективности защиты, используя комбинацию уровней эффективности защиты всех функций безопасности модуля, либо с помощью верификации и валидации общей связанной с безопасностью функции модуля. Более подробные сведения приведены в 4.3.8.

Элемент 16 «Моделирование» соответствует разным типам моделей, предназначенных для имитационного моделирования и разработки.

Элемент 17 «Исполняемая форма» соответствует программным кодам, исполнение которых обеспечивает достижение или поддержку назначения модуля.

Классы, представленные в настоящем стандарте, могут быть описаны с помощью табличного формата, приведенного в приложении Е.

4.2 Взаимосвязь между ОИМ и конкретными ИМ

Общую информационную модель модулей необходимо использовать в информационных моделях всех типов модулей, к которым относятся аппаратно-программные модули, программные модули и аппаратные модули. Взаимосвязи между информационными моделями модулей представлены на рисунке 4.1. Аппаратно-программные модули обладают как аппаратными, так и программными свойствами. Пример взаимосвязей между информационными моделями аппаратных модулей, программных модулей и аппаратно-программных модулей представлен на рисунке 4.2.

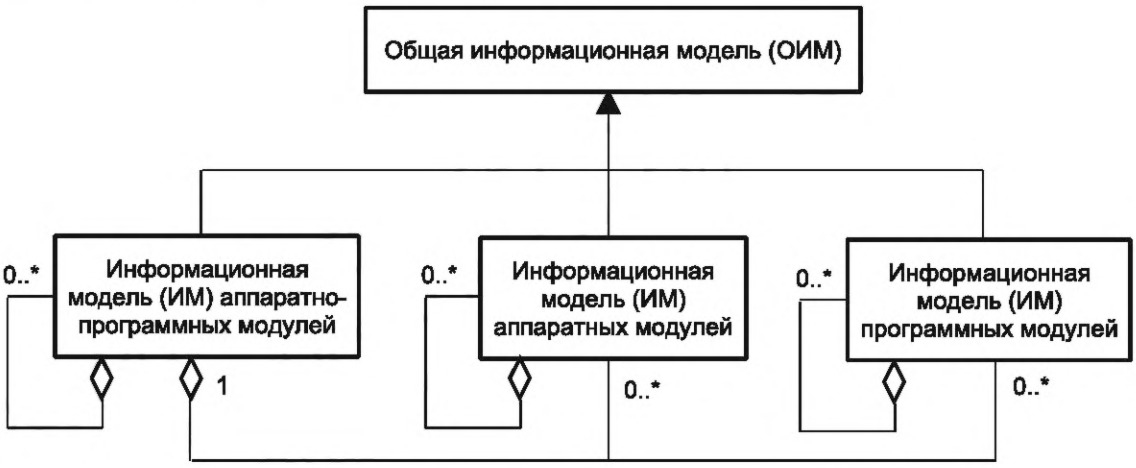

Рисунок 4.1 — Взаимосвязи между информационными моделями модулей

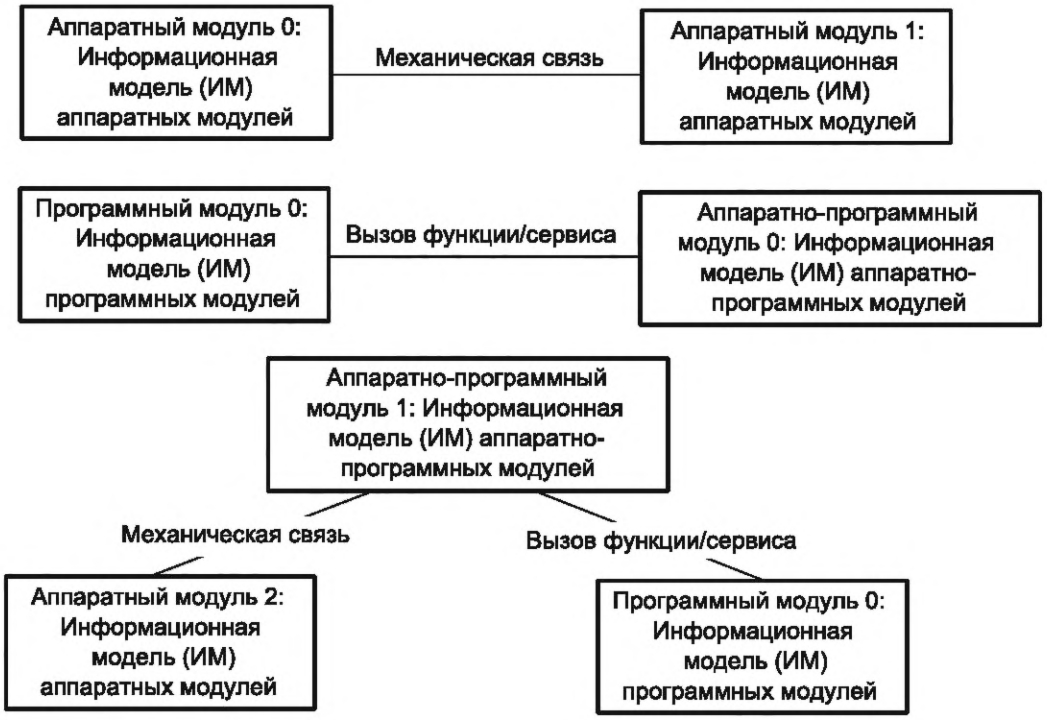

Рисунок 4.2 — Пример взаимосвязей между информационными моделями модулей

4.3 Класс для общей информационной модели

4.3.1 Общие положения

Общая информационная модель должна быть сформирована из 9 внутренних классов, представленных на рисунке 4.3, где 8 классов взяты из таблицы 4.1, а дополнительный класс Status представляет информацию о состоянии модуля. Данный класс в основном используется во время работы модуля. Четыре элемента группы GenInfo из таблицы 4.1, которыми являются Имя модуля, Описание, Изготовитель и Примеры, становятся атрибутами класса Common Information Model (CIM).

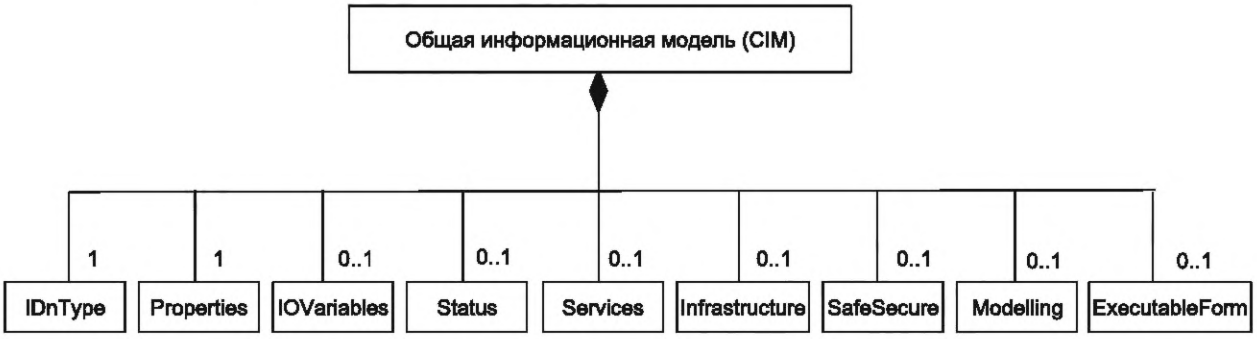

Рисунок 4.3 — Классы общей информационной модели

Класс CIM (Общая информационная модель) должен иметь атрибуты, представленные на рисунке 4.4 и в таблице 4.5, которые основаны на ИСО 22166-1:2021, разделы 4—7. Значения атрибутов ModuleName, Description, Manufacture и Examples определены в приложении А. Классы IDnType, Properties, IOVariables, Status, Services, Infrastructure, SafeSecure, Modelling и ExecutableForm определены в 4.3.2—4.3.10.

Примечание 1 — Атрибуты могут быть объявлены принадлежащими к одному из следующих типов: конфиденциальный (-), защищенный (*) или публичный (+). Сначала указывают имя атрибута, а затем определяют его тип данных. Символом, разделяющим имя атрибута и его тип данных, является двоеточие (:). Если атрибуты объявлены публичными, то нет необходимости определять функции доступа к ним.

Примечание 2 — Четыре элемента: имя модуля, описание, изготовитель и примеры — являются атрибутами класса CIM.

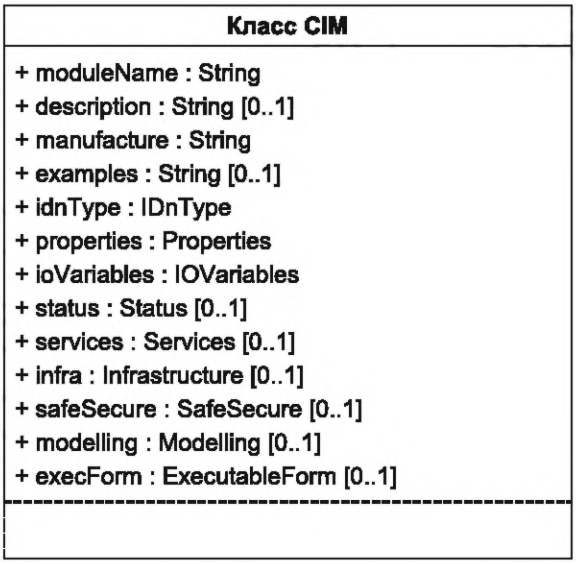

Рисунок 4.4 — Диаграмма классов класса CIM

Таблица 4.5 — Класс CIM

Описание: Корневой класс для CIM (Общая информационная модель)				
Выведен из: Не применимо				
Атрибуты:				
moduleName	String	M	1	Имя, представляющее модуль
description	String	O	1	Общее представление модуля, что он собой представляет, что он делает и как его использовать
manufacturer	String	M	1	Контактная информация о проектировщике, разработчике или изготовителе модуля
examples	String	O	1	Типичные случаи применения модуля
idnType	IDnType	M	1	ID модуля включает уникальный идентификатор, ID экземпляра, собственные программные/аппаратные свойства в виде их ID. Более подробно см. 4.3.2
properties	Properties	M	1	Список конфигурируемых параметров модуля. Более подробно см. 4.3.3
ioVariables	IOVariables	O	1	Информация о входных и выходных переменных модуля. Более подробно см. 4.3.4
status	Status	O	1	Информация о состоянии и работоспособности модуля. Более подробно см. 4.3.5
services	Services	O	1	Интерфейсы, которые модуль предоставляет и использует для сервисов робота. Более подробно см. 4.3.6
infra	Infrastructure	O	1	Информация об инфраструктуре модуля включает сведения об электропитании, шине данных, базе данных и защите от проникновения. Более подробно см. 4.3.7

Окончание таблицы 4.5

safeSecure	SafeSecure	O	1	Уровни безопасности и защищенности, которые обеспечивает модуль. Более подробно см. 4.3.8
modelling	Modelling	O	1	Информация, связанная с моделированием модуля. Более подробно см. 4.3.9
execForm	ExecutableForm	O	1	Информация, связанная с программами, выполняемыми для достижения или поддержки назначения модуля. Более подробно см. 4.3.10

4.3.2 Класс для идентификатора модуля

Информация об идентификаторе модуля должна быть определена в классе IDnType, который должен иметь атрибуты, представленные на рисунке 4.5 и в таблице 4.6. Значения атрибута moduleID и других атрибутов из таблицы 4.6 определены в приложениях B и C. Если в составном модуле присутствуют модули одного и того же типа, то такие модули должны быть идентифицированы индивидуально. Это обеспечивает элемент Instance ID (или IID). Для программных модулей IID присваивается динамически менеджером модулей. Для модулей с аппаратными свойствами IID присваивается статически изготовителем модулей. Если IID используется, то он должен быть уникальным. Версия информационной модели является номером версии информационной модели, использованной при определении модуля. Версия информационной модели изменяется при изменении атрибутов классов, представленных в настоящем стандарте.

Примечание 1 — Под модулем с аппаратными свойствами понимается аппаратный модуль или модуль с аппаратными и программными свойствами.

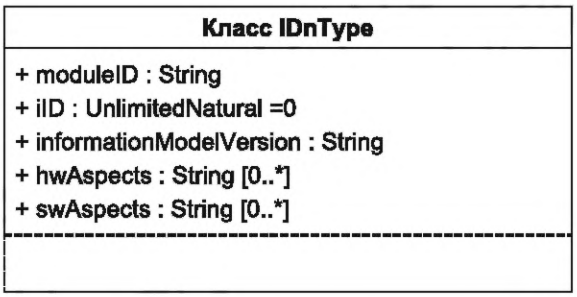

Рисунок 4.5 — Диаграмма классов для класса IDnType

Таблица 4.6 — Класс IDnType

Описание: Информация об элементе Module ID должна быть определена в классе IDnType				
Выведен из: Не применимо				
Атрибуты:				
moduleID	String	M	1	Формат атрибута moduleID определен в приложении B
iID	UnlimitedNatural	M	1	Instance ID, значением по умолчанию является 0
informationModelVersion	String	M	1	Номер версии информационной модели
hwAspects	String	O	N	Список идентификаторов модулей для аппаратных модулей и модулей с аппаратными и программными свойствами
swAspects	String	O	N	Список идентификаторов модулей для программных модулей

Атрибуты hwAspects и swAspects представляют списки идентификаторов модулей с аппаратными свойствами и с программными свойствами соответственно.

При создании экземпляра данного класса должен быть задан атрибут instanceID с использованием атрибута moduleID. Если какой-либо модуль добавляется к составному модулю, то данный составной модуль должен проверить, не произойдет ли дублирование атрибута moduleID в данном составном модуле. Если дублирование имеет место (т. е. такой же модуль уже существует в составном модуле), то менеджер модулей или изготовитель составного модуля должен присвоить данному модулю новый атрибут instanceID.

Примечание 2 — Элемент Module ID задает изготовитель модуля.

Примечание 3 — Атрибут moduleID, включая IID, используется для доступа к модулю. Значением IID по умолчанию является «0».

Примечание 4 — Значение атрибута informationModelVersion равно «1.0», если используется информационная модель, определенная в настоящем стандарте.

4.3.3 Класс для характеристик

Информация о характеристиках модуля должна быть определена в классе Properties, который должен представлять характеристики модуля в соответствии с рисунком 4.6 и таблицами 4.7—4.9, содержащими информацию, необходимую для исполнения модуля, и информацию об операционной среде или состоянии модуля. На рисунке 4.6 приведена диаграмма классов для класса Properties и представлена взаимосвязь между классами, представленными в таблицах 4.7—4.9. Конкретные значения атрибутов, таких как имя и тип данных, должны быть определены с использованием значений, указанных для элемента Property в приложении С.

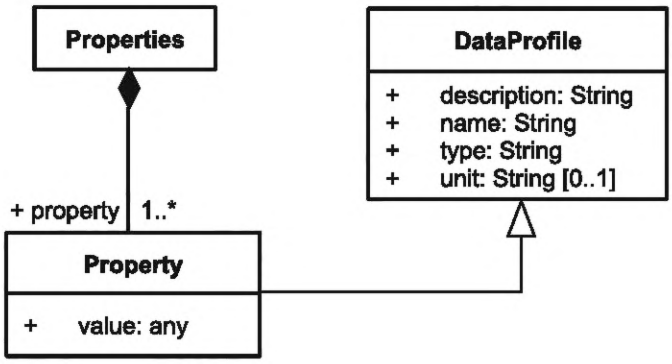

Рисунок 4.6 — Диаграмма классов для класса Properties

Таблица 4.7 — Класс DataProfile

Описание: Класс для профилей данных				
Выведен из: Не применимо				
Атрибуты:				
description	String	M	1	Описание свойства
name	String	M	1	Имя свойства
type	String	M	1	Тип данных свойства
unit	String	O	1	Единицы измерения свойства. Если их нет, то значением является ‘null’

Таблица 4.8 — Класс Property

Описание: Класс для свойства				
Выведен из: DataProfile				
Атрибуты:				
value	any	M	1	Значение свойства

Таблица 4.9 — Класс Properties

Описание: Определяет информацию о свойствах модуля				
Выведен из: Не применимо				
Атрибуты:				
property	Property	O	N	Список атрибутов, полученных согласно классу Property

Приведенные далее примеры на языке XML должны быть преобразованы, как представлено на рисунке 4.6:

```
<!—example for hardware-software module -->
<Property name="maxRatedCurrent" value=15 type="float32" unit="ampere" description =
"maximum of rated current for motor" />
<Property name="angleResolution" value=1 type="float32" unit="degree" description =
"Angle Resolution of Lidar Sensor" />
<Property name="minOpRange" value=0.05 type="float32" unit="meter" description =
"minimum operation range of Lidar Sensor" />
<Property name="maxOpRange" value=10 type="float32" unit="meter" description =
"maximum operation range of Lidar Sensor" />
<Property name="commModule" value="Ethernet microUSB" type="array of string"
unit="null" description = "communication protocol type of Lidar Sensor" />
<!—example for software module -->
<Property name="Weight" value=150 type="float32" unit="gram" description =
"weight of Lidar Sensor" />  <-- example for hardware-software module -->
<Property name="maxRadius" value=2 type="float32" unit="meter" description =
"maximum radius for Obstacle Avoidance" />
<Property name="minRadius" value=0.5 type="float32" unit="meter" description =
"minimum radius for Obstacle Avoidance" />
<!-- example for hardware module -->
<Property name="origin" type="array of float32" unit="cm", "radian" description =
"pose of the inertial reference frame, relative to the link reference frame including
xyz and rpy" value=[0 0 0 0 0 0]/>
<Property name="mass" type="float32" unit="gram" description = "mass of a link" value =15 />
< Property name="inertia" type="array of float32" unit="g-cm2" description =
"inertial properties of the link, 3x3 matrix" value =[1 0 0; 0 1 0; 0 0 1] />
<Property name="shape" type="string" unit="" description = "shape of the visual
object" value =cylinder />
<Property name="size" type="3x1 array of float32" unit="cm" description = "three
side lengths of the box" value =[100; 200; 100] />
```

4.3.4 Класс для входных и выходных переменных

Информация о входных и выходных переменных модуля должна быть определена в классе IOVariables, представленном на рисунке 4.7 и в таблицах 4.10—4.15. Класс IOVariables должен определять используемые в модуле переменные, которые содержат информацию, необходимую для выполнения модуля. Эти переменные должны быть определены с помощью класса Variable, представленного в таблице 4.12. Класс Variable наследует от класса VariableProfile, представленного в таблице 4.11, который в свою очередь наследует от класса DataProfile, представленного в таблице 4.7. Для хранения дополнительной информации в качестве контейнера может быть использован класс NVList, представленный в таблице 4.14. NVList содержит объекты класса NameValue, определенного в таблице 4.15. Значения IN, OUT и INOUT определены в таблице 4.10 как перечислимые типы данных. Конкретные значения атрибутов, таких как имя и тип данных, должны быть определены с использованием значений, указанных для элемента IOVariables в приложении С. На рисунке 4.7 представлена взаимосвязь между классами для класса IOVariables.

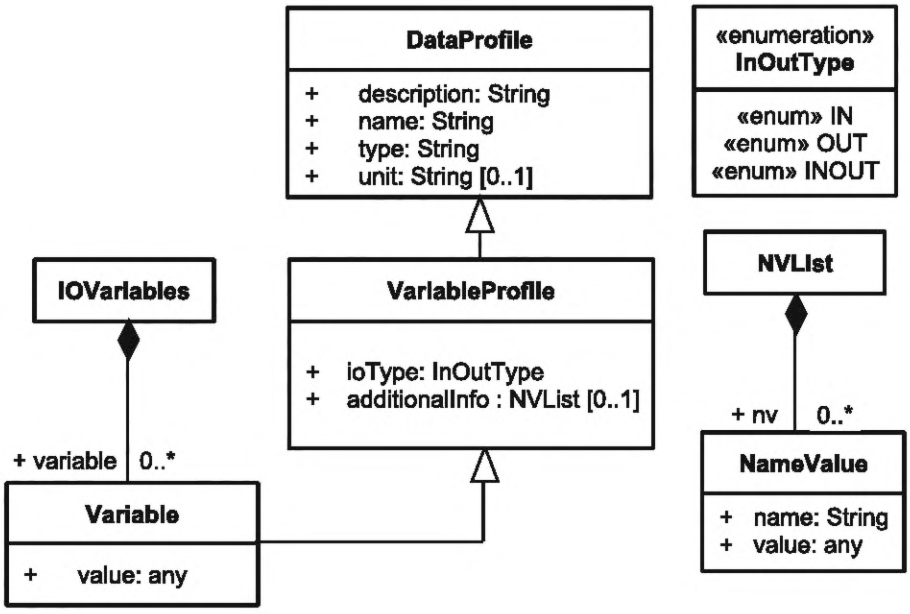

Рисунок 4.7 — Взаимосвязь между классами для класса IOVariables

Таблица 4.10 — Перечисление InOutType

Описание: Типы входных и выходных переменных	
Атрибуты:	
IN	Используют для переменных, в которые внешние модули заносят данные
OUT	Используют для переменных, из которых внешние модули считывают данные
INOUT	Используют для переменных, из которых внешние модули считывают данные и в которые заносят данные

Таблица 4.11 — Класс VariableProfile

Описание: Класс для профиля переменных				
Выведен из: DataPrifile (см. таблицу 4.7)				
Атрибуты:				
ioType	InOutTipe	M	1	Один из {IN, OUT, INOUT}. См. таблицу 4.10
additionalInfo	NVList	O	1	Класс NVList служит в качестве контейнера для дополнительной информации по разным классам. См. таблицу 4.14

Таблица 4.12 — Класс Variable

Описание: Класс для переменной				
Выведен из: VariablePrifile (см. таблицу 4.11)				
Атрибуты:				
value	any	M	1	Значение переменной

Таблица 4.13 — Класс IOVariables

Описание: Класс для входных и выходных переменных модуля				
Выведен из: Не применимо				
Атрибуты:				
variable	Variable	O	N	Список атрибутов, созданных в соответствии с тегом Variable

Таблица 4.14 — Класс NVList

Описание: Список NameValue для дополнительной информации по разным классам				
Выведен из: Не применимо				
Атрибуты:				
nv	NameValue	M	N	Список элементов Name Value. См. таблицу 4.15

Таблица 4.15 — Класс NameValue

Описание: Список NameValue для дополнительной информации по разным классам				
Выведен из: Не применимо				
Атрибуты:				
name	String	M	1	Имя дополнительной информации
value	any	M	1	Значение дополнительной информации

Приведенные далее примеры на языке XML должны быть преобразованы, как представлено на рисунке 4.7:

```
 <IOVariables>
     <Input name="controlValue" type="float32" unit="ampere" description="current
control to motor" /> <!-- IOType=IN -->
 <Output name="encoder" value= 0 type ="uint32" unit = "none" description = "absolute
encoder's value" /> <!-- IOType=OUT -->
     <Inout name="state" type="uint8" value= 0 unit = "none" description = "status
of motor"/>  <!-- IOType=INOUT -->
 </IOVariables>
```

Из этих примеров генерируются следующие переменные и типы данных:

```
+ controlValue : float32 // or public float32 controlValue; current control command
to motor, input
+ encoder=0 : uint32 // or public uint32 encoder=0; value of absolute encoder, output
+ state=0 : uint8 // or public uint8 state=0; reset or read status of motor, input
and output
```

4.3.5 Класс для состояния

Информация о состоянии и работоспособности модуля должна быть определена в классе Status, представленном на рисунке 4.8 и в таблице 4.17. Класс Status представляет текущее состояние модуля, которое определяет состояние работоспособности модуля, если существует информационная модель аппаратного обеспечения, состояние жизненного цикла выполнения модуля, если существует информационная модель программного обеспечения, а также тип ошибки, которая может возникнуть в процессе работы модуля.

Следует использовать номера, присвоенные ошибкам в IEEE/Open Group 1003.1-2017. Для каждого программного модуля могут быть определены дополнительные номера ошибок, но которые не должны конфликтовать с номерами ошибок в POSIX.1003.1.

Примечание 1 — Примерами значений атрибута HealthCond являются: GH (полная работоспособность), H1 (состояние 1), H2 (состояние 2), Fa (неисправность).

Примечание 2 — Примерами ошибок, указываемых в атрибуте ErrorType, являются: «Operation not permitted (Работа не разрешена)», «Authentication Error (Ошибка авторизации)», «Bad Parameter (Неправильный параметр)», «Unsupported Service (Неподдерживаемый сервис)», «out of Range (Вне диапазона) и «Precondition not met (Заданные условия не соблюдены)». Значения атрибута ErrorType будут точно определены в ИСО 22166-202 и других стандартах.

Примечание 3 — Значение атрибута ExeStatus определено в ИСО 22166-1 и должно быть одним из следующих: CREATED (Создан), IDLE (Ожидает), EXECUTING (Выполняется), DESTRUCTED (Удален) и ERROR (Ошибка). Эти значения определены в таблице 4.16 как перечислимые типы данных.

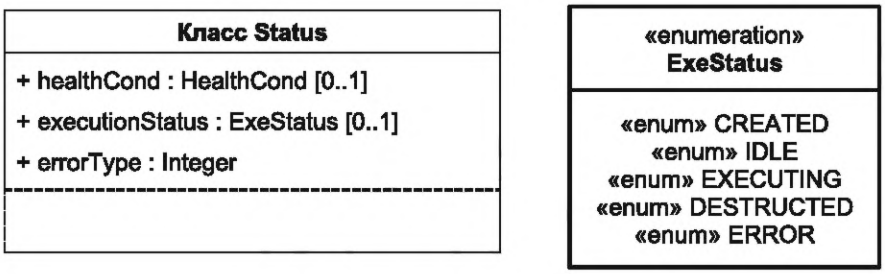

Рисунок 4.8 — Диаграмма классов для класса Status

Таблица 4.16 — Перечисление ExeStatus

Описание: Определяет типы состояния выполнения	
Атрибуты:	
CREATED	Состояние «создан»
IDLE	Состояние «ожидает»
EXECUTING	Состояние «выполняется»
DESTRUCTED	Состояние «удален»
ERROR	Состояние «ошибка»

Таблица 4.17 — Класс Status

Описание: Класс для состояния модуля				
Выведен из: Не применимо				
Атрибуты:				
healthCond	HealthCond	O	1	У некоторых модулей атрибут HealthCond не имеет значений. Данный перечислимый тип данных имеет следующие значения: {GH, H1, H2, Fa}, где GH, H1, H2 и Fa означают «Полная работоспособность», «Тип состояния 1», «Тип состояния 2» и «Сбой» соответственно
executionStatus	ExeStatus	O	1	У некоторых модулей атрибут ExeStatus может не иметь значений (см. таблицу 4.16)
errorType	Integer	M	1	Зависит от типа модуля. См. примечание 2

4.3.6 Класс для сервисов

Информация о сервисах модуля должна быть определена в классе Services, который представляет методы (или функции), реализуемые данным модулем, как показано на рисунке 4.9 и в таблице 4.18. Атрибуты и методы, представленные в таблице 4.18, должны быть получены с помощью CIMServicePackage, представленного ниже на языке IDL, либо определены из классов, определенных в таблицах 4.19—4.24. На рисунке 4.9 представлена взаимосвязь между классами для класса Services, которые представлены в таблицах 4.18—4.24.

Примечание — В данном пункте сервисы определены с использованием ИСО/МЭК 19516:2020.

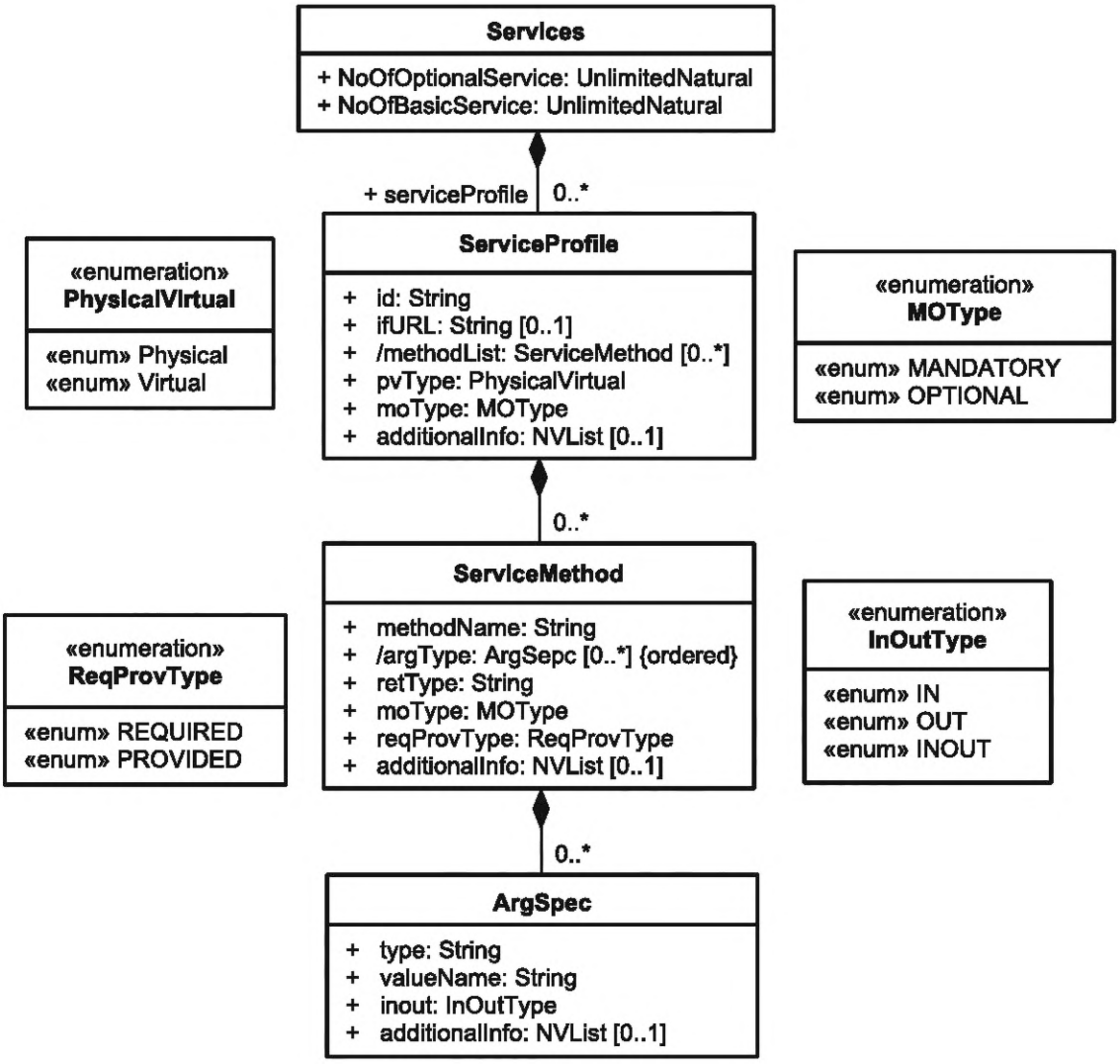

Рисунок 4.9 — Взаимосвязь между классами для класса Services

Таблица 4.18 — Класс Services

Описание: Класс для сервисов модуля				
Выведен из: Не применимо				
Атрибуты:				
NoOfBasicService	UnlimitedNatural	M	1	Число предоставляемых базовых сервисов
NoOfOptionalService	UnlimitedNatural	M	1	Число предоставляемых факультативных сервисов
serviceProfile	ServiceProfile	O	N	Прототип для базовых (обязательных) сервисов, serviceProfile может быть представлен списком. См. таблицу 4.20

Таблица 4.19 — Перечисление PhysicalVirtual

Описание: Определяет тип интерфейса	
Атрибуты:	
Physical	Физический интерфейс, например механический или электрический интерфейс
Virtual	Виртуальный интерфейс, например программный интерфейс

Таблица 4.20 — Класс ServiceProfile

Описание: Класс для профиля сервисов				
Выведен из: Не применимо				
Атрибуты:				
id	String	M	1	Имя профиля сервиса. Один или несколько профилей сервисов могут быть представлены
ifURL	String	O	1	URL/путь файла, определенного в IDL (см. CIMService Package) для сервиса
methodList	ServiceMethod	O	N	Список методов для сервиса, определенных на языке XML. См. таблицу 4.23
pvType	PhysicalVirtual	M	1	PhysicalVirtual: перечислимый тип данных {Physical, Virtual} (см. таблицу 4.19). Physical (Mechanical/Elec) Method или Virtual (Software) Method
moType	MOType	M	1	MOType: перечислимый тип данных {MANDATORY, OPTIONAL}. См. таблицу 4.21
additionalInfo	NVList	O	1	Аргументы с их значениями по умолчанию при инициализации для сервиса. См. таблицу 4.14
Должен быть представлен только один из двух типов аргументов ifURL и methodList. Если он не представлен, то значением является NULL.				

Таблица 4.21 — Перечисление MOType

Описание: Определяет тип сервиса — обязательный или факультативный	
Атрибуты:	
MANDATORY	Обязательный
OPTIONAL	Факультативный

Таблица 4.22 — Перечисление ReqProvType

Описание: Определяет тип сервиса — требуемый или предоставляемый	
Атрибуты:	
REQUIRED	Требуемый сервис
PROVIDED	Предоставляемый сервис

Таблица 4.23 — Класс ServiceMethod

Описание: Класс для метода сервиса				
Выведен из: Не применимо				
Атрибуты:				
methodName	String	M	1	Имя метода сервиса
argType	ArgSpec	O	N	Упорядоченный список аргументов, используемых в методе. См. таблицу 4.24
retType	String	M	1	Тип данных возвращаемого значения. См. таблицу С.5
moType	MOType	M	1	Обязательный или факультативный метод. См. таблицу 4.21
reqProvType	ReqProvType	M	1	Предоставляемый или требуемый метод. См. таблицу 4.22
additionalInfo	NVList	O	1	Список NameValue для дополнительной информации. См. таблицу 4.14

Таблица 4.24 — Класс ArgSpec

Описание: Класс для определения аргумента				
Выведен из: Не применимо				
Атрибуты:				
type	String	O	N	Тип данных аргумента, используемого в методе. См. таблицу С.5
valueName	String	M	1	Имя аргумента, используемого в методе
inout	InOutType	M	1	См. таблицу 4.10
additionalInfo	NVList	O	1	Список NameValue для дополнительной информации. См. таблицу 4.14

```
CIMServicePackage written in IDL:

Module CIMServicePackage {
interface CIMService; // abstract declaration
struct NameValue
{
   string name;
   any value; // "any" means any data type
  };
  // Mandatory/Optional enum type
  enum MOType
  {
        MANDATORY,
        OPTIONAL
  };
  struct ArgSpec {
      string type;              // argument type
      string valueName;         // argument name
      InOut inout;              // enumeration {IN, OUT, INOUT}
      NVList additionalInfo;    // additional information
   } ;
   typedef sequence<ArgSpec> ArgSpecList;    //sequence means a list or vector of
                                             // given data
   struct ServiceMethod
   {
      string methodName;
      ArgSpecList argType;
      string retType;
      MOType moType;     // enumeration {"MANDATORY", "OPTIONAL"}
      ReqProvType reqProvType; // Provided or required method.
      NVList additionalInfo;   // additional information
   };
   typedef sequence<NameValue> NVList;
   typedef sequence<ServiceMethod> MethodList;
   enumeration PhysicalVirtual
   {
        PHYSICAL,
        VIRTUAL
   };
   // ServiceProfile
   struct ServiceProfile
   {
```

```
    string id;
    PhysicalVirtual pvType; // Enumeration PHYSICAL, VIRTUAL
    MethodList methodList;  // mandatory/optional method defined in CIMService
    MOType moType;   // MOType: {MANDATORY, OPTIONAL}.
    NVList additionalInfo;  //additional information
  };
  interface CIMService
  {
    // define here for prototypes of methods for CIM Service
    // format: <type_spec | void> <method_identifier> ( <parameter decls> )
    // <parameter decls> ::= <para_dcl>
    // <param_dcl> ::= <"in"|"out"|"inout"> <type_spec> <parameter_name>
    // example:
    //   uint8 initialize(in int16 val1, in float64 val2);
    //   boolean checkElecConnectivity(string connType, string keying,
    //                                 string busProtocol)
    //   boolean linkConnectivity(array origin, float mass, array inertia,
    //     string shape, array size,array axis,string connection,array collision)
  };
};
```

4.3.7 Класс для инфраструктуры

Информация об инфраструктуре модуля должна быть определена в классе Infrastructure, который представляет типы инфраструктур, поддерживаемых данным модулем, как показано на рисунке 4.10 и в таблице 4.27. В данном пункте определены следующие инфраструктурные модули: Electric Power, Data Bus, Database и Ingress Protection.

Примечание 1 — Если дополнительный инфраструктурный модуль существует, то он может быть определен в связанной с ним информационной модели.

Классы DataBus и Power, использованные в классе Infrastructure, представлены в таблицах 4.25 и 4.26 соответственно.

Примечание 2 — Значения аргументов typePhyMac и typeApp в таблице 4.25 и аргумента dbType в таблице 4.27 определены в ИСО 22166-202.

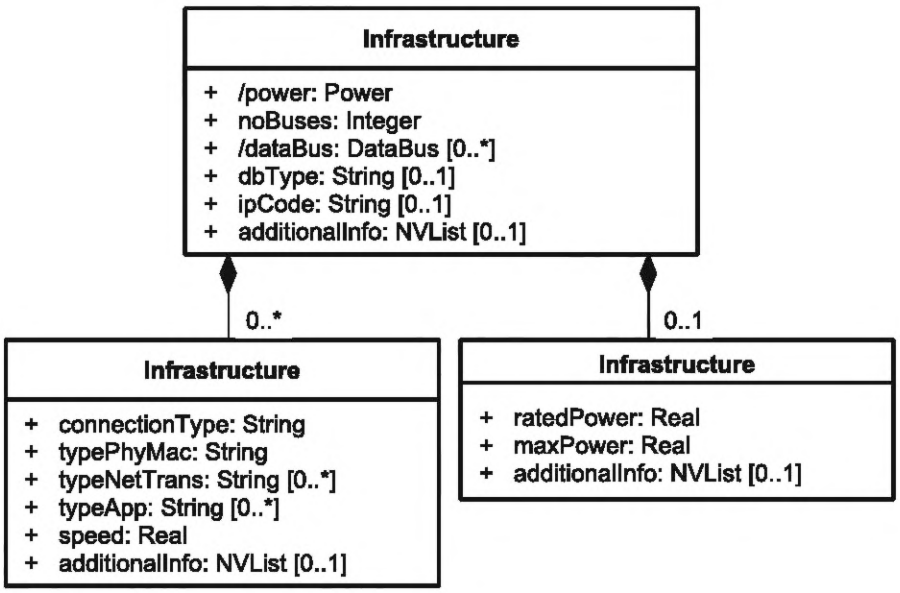

Рисунок 4.10 — Взаимосвязь между классами для класса Infrastructure

Таблица 4.25 — Класс DataBus

Описание: Класс для шины данных				
Выведен из: Не применимо				
Атрибуты:				
ConnectionType	String	M	1	Тип соединения: USB-A, USB-C, USD-C, RG45, DB9, DB15, DB25
typePhyMac	String	O	N	Физический/MAC тип протокола: USB, CAN2.0, EtherCAT, Ethernet, RS485
typeNetTrans	String	M	1	Сетевой протокол и транспортный протокол, IP, TCP, CANopen
typeApp	String	M	1	Протокол уровня сеанса/представления/приложения: список протоколов
speed	Real	M	1	Скорость передачи: Кбит/с
additionalInfo	NVList	O	1	Список NameValue для дополнительной информации. См. таблицу 4.14

Таблица 4.26 — Класс Power

Описание: Класс для мощности				
Выведен из: Не применимо				
Атрибуты:				
ratedPower	Real	M	1	Номинальная мощность в ваттах
maxPower	Real	M	1	Максимальная мощность в ваттах
additionalInfo	NVList	O	1	Список NameValue для дополнительной информации. См. таблицу 4.14

Таблица 4.27 — Класс Infrastructure

Описание: Класс для инфраструктуры				
Выведен из: Не применимо				
Атрибуты:				
power	Power	O	1	См. таблицу 4.26
noBuses	Integer	O	1	Количество шин данных, использованных в модуле
dataBus	DataBus	O	N	Шины данных (или коммуникационные протоколы), которые предоставляет модуль. Если значение атрибута noBuses больше 0, то данное поле является обязательным. См. таблицу 4.25
dbType	String	O	1	Типы системы управления базой данных, использованной в модуле (например, Oracle, SQL и т. д.)
ipCode	String	O	1	Код IP. Степень защиты по МЭК 60529:2013; две цифры
additionalInfo	NVList	O	1	Список NameValue для дополнительной информации. См. таблицу 4.14

4.3.8 Класс для безопасности и защищенности

Класс SafeSecure должен представлять уровень безопасности и уровень защищенности, предоставляемые модулем, как показано на рисунке 4.11 и в таблице 4.38. На рисунке 4.11 приведена взаимосвязь между классами для класса SafeSecure, представленными в таблицах 4.28—4.38.

Меры по обеспечению конкретной функции безопасности должны быть определены с помощью типа данной функции безопасности, а также уровня эффективности защиты (отсутствие или a—e) или уровня полноты безопасности (0, 1—3) для данного типа функции безопасности. У модуля могут

отсутствовать функции безопасности или их может быть даже несколько. Функция безопасности должна быть определена с помощью класса SafetyFunction, представленного в таблице 4.34. Разные типы функций безопасности представлены в таблице 4.28. Уровень безопасности, обеспечиваемый функциями безопасности, должен быть определен с помощью УЭЗ или УПБ. Уровень защищенности должен быть определен для функций защищенности.

Меры по обеспечению конкретной киберзащищенности должны быть определены с помощью установления типа защищенности и уровня защищенности (0—4) для данного типа. Разные типы мер по обеспечению защищенности представлены в таблице 4.29. У модуля могут отсутствовать меры по обеспечению защищенности или их может быть даже несколько. Мера по конкретной киберзащищенности должна быть определена с помощью класса CyberSecurity, представленного в таблице 4.37. Если модуль реализует две или более меры по обеспечению защищенности, то такой модуль должен иметь общий уровень защищенности модуля, определяемый комбинацией уровней защищенности предоставляемых мер обеспечения защищенности либо посредством верификации и валидации всех мер обеспечения защищенности модуля.

Уровень безопасности модуля предоставляется только в качестве информации при оценке безопасности системы. Даже если уровень безопасности для функции безопасности модуля задан, составной модуль должен быть независимо оценен в отношении своей функции безопасности в соответствии с методами оценивания безопасности на системном уровне.

Примечание 1 — Определение уровня безопасности для функции безопасности основано на ИСО 13482:2014, а определение уровня защищенности для функции защищенности основано на МЭК 62443-4-2:2019.

Примечание 2 — Предоставляемые изготовителем значения атрибутов в таблицах 4.34 и 4.37 являются неизменяемыми.

Примечание 3 — Уровень эффективности защиты при отсутствии функции безопасности обозначается как «none» или «n».

Примечание 4 — Функции, связанные с безопасностью, и/или функции, связанные с защищенностью, представлены в классе Services, который может быть использован для управления безопасностью/защищенностью.

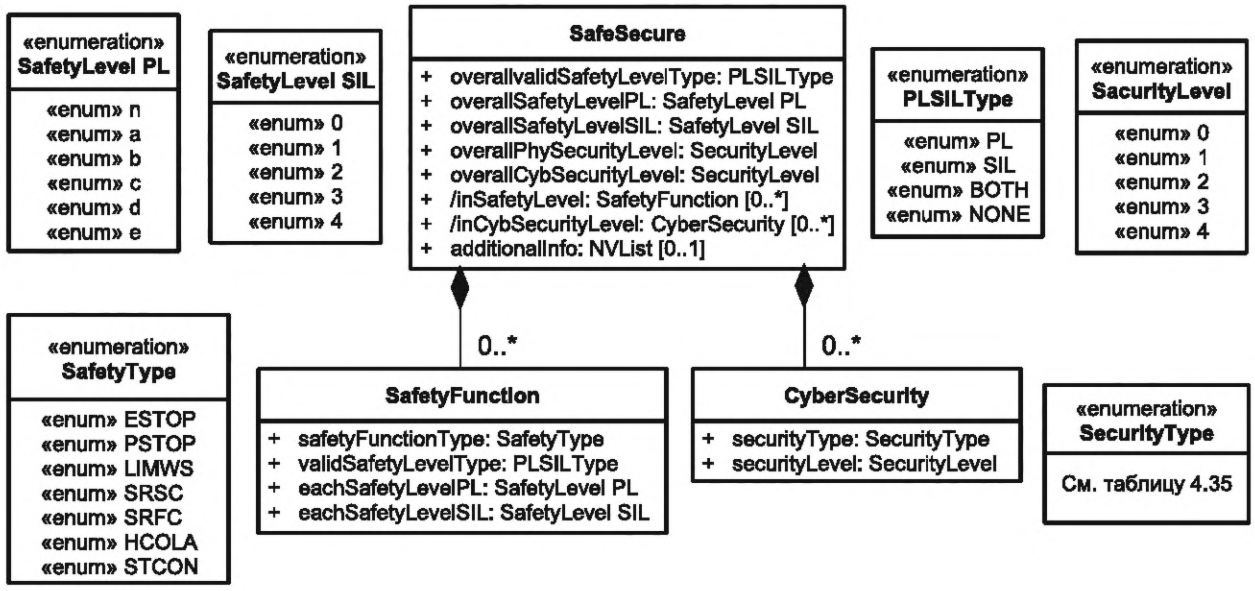

Рисунок 4.11 — Взаимосвязь между классами для класса SafeSecure

Таблица 4.28 — Типы функции безопасности и их теги

Функция безопасности (ИСО 13482:2014)	Тег
Аварийная остановка	ESTOP
Защитная остановка	PSTOP
Ограничения на рабочее пространство (включая обход запрещенных зон)	LIMWS

Окончание таблицы 4.28

Связанное с безопасностью управление скоростью	SRSC
Связанное с безопасностью управление усилием	SRFC
Избегание опасных столкновений	HCOLA
Управление устойчивостью (включая защиту от опрокидывания)	STCON

Примечание 5 — Функции безопасности, не представленные в таблице 4.28, могут быть добавлены при необходимости.

Таблица 4.29 — Типы меры по обеспечению киберзащищенности и их теги

Тип меры по обеспечению защищенности (МЭК 62443-4-2:2019)	**Тег**
Идентификация и аутентификация пользователя	HU_IA
Идентификация и аутентификация программного процесса и устройства	SD_IA
Управление клиентами	ACNT_MGT
Управление идентификаторами	ID_MGT
Управление аутентификаторами	AUTH_MGT
Управление беспроводным доступом	WIRELEE_MGT
Прочность аутентификации на основе паролей	PW_AUTH
Сертификация инфраструктуры публичных ключей	PK_CERT
Устойчивость аутентификации на основе публичных ключей	STR_PK_AUTH
Неудачные попытки логина	LOGIN_NO
Доступ через ненадежные сети	ACC_UNTRUST_NET
Применение авторизации	AUTHORIZE
Контроль использования беспроводной связи	WIRELESS_USE
Блокировка сессии	SESS_LOCK
Дистанционное завершение сессии	SESS_TERM
Согласованный контроль сессий	SECC_CNTR
Поддающиеся проверке события	AUDT_EVT
Метки времени	TIMESTM
Невозможность отказа от авторства	NON_REP
Достоверность коммуникаций	COMM_INTG
Защита от вредоносных программ	PROT_MALI_CODE
Верификация функциональности защищенности	SECUR_VERIFY
Программная и информационная целостность	SW_INTGT
Валидация входов	INPUT_VALD
Детерминированные выходы	DET_OUT
Обработка ошибок	ERR_HNDL
Безошибочность сессий	SESS_INTGT
Конфиденциальность информации	INFO_CONFI

Окончание таблицы 4.29

Сохраняемость информации	INFO_PERS
Применение криптографии	CRYTO
Ограниченный поток данных	RSTIC_FLOW
Защита от отказа в обслуживании	DoS
Управление ресурсами	RESOU_MGT
Исправление и восстановление системы контроля	CNTR_RECOV_RECON

Таблица 4.30 — Перечисление SafetyType

Описание: Тип безопасности модуля. См. таблицу 4.28	
Атрибуты:	
ESTOP	Аварийная остановка
PSTOP	Защитная остановка
LIMWS	Ограничения на рабочее пространство (включая обход запрещенных зон)
SRSC	Связанное с безопасностью управление скоростью
SRFC	Связанное с безопасностью управление усилием
HCOLA	Избегание опасных столкновений
STCON	Управление устойчивостью (включая защиту от опрокидывания)

Таблица 4.31 — Перечисление SafetyLevelPL

Описание: Представление уровня эффективности защиты	
Атрибуты:	
n	None. Нет уровня эффективности защиты
a	PL a
b	PL b
c	PL c
d	PL d
e	PL e

Таблица 4.32 — Перечисление SafetyLevelSIL

Описание: Уровень полноты безопасности	
Атрибуты:	
0	None. Нет уровня полноты безопасности
1	SIL 1
2	SIL 2
3	SIL 3
4	SIL 4

Таблица 4.33 — Перечисление PLSILType

Описание: Тип представления безопасности	
Атрибуты:	
PL	Уровень безопасности, представленный в типе УПБ

Окончание таблицы 4.33

SIL	Уровень безопасности, представленный в типе УЭЗ
BOTH	Уровень безопасности, представленный в типах УПБ и УЭЗ
NONE	Уровень безопасности, не представленный, ни в типе УПБ ни в типе УЭЗ

Таблица 4.34 — Класс SafetyFunction

Описание: Функция безопасности модуля				
Выведен из: Не применимо				
Атрибуты:				
safetyFunctionType	SafetyType	M	1	Тип безопасности. См. таблицу 4.30
validSafetyLevelType	PLSILType	M	1	Предоставление типа уровня безопасности. См. таблицу 4.33
eachSafetyLevelPL	SafetyLevelPL	M	1	См. таблицу 4.31. n: Нет функции безопасности
eachSafetyLevelSIL	SafetyLevelSIL	M	1	См. таблицу 4.32. 0: Нет функции безопасности

Таблица 4.35 — Перечисление SecurityType

Описание: Тип защищенности модуля. См. таблицу 4.29	
Атрибуты:	
HU_IA	Идентификация и аутентификация пользователя
SD_IA	Идентификация и аутентификация программного процесса и устройства
ACNT_MGT	Управление клиентами
ID_MGT	Управление идентификаторами
AUTH_MGT	Управление аутентификаторами
WIRELEE_MGT	Управление беспроводным доступом
PW_AUTH	Прочность аутентификации на основе паролей
PK_CERT	Сертификация инфраструктуры публичных ключей
STR_PK_AUTH	Устойчивость аутентификации на основе публичных ключей
LOGIN_NO	Неудачные попытки логина
ACC_UNTRUST_NET	Доступ через ненадежные сети
AUTHORIZE	Применение авторизации
WIRELESS_USE	Контроль использования беспроводной связи
SESS_LOCK	Блокировка сессии
SESS_TERM	Дистанционное завершение сессии
SECC_CNTR	Согласованный контроль сессий
AUDT_EVT	Поддающиеся проверке события
TIMESTM	Метки времени
NON_REP	Невозможность отказа от авторства
COMM_INTG	Достоверность коммуникаций
PROT_MALI_CODE	Защита от вредоносных программ

Окончание таблицы 4.35

SECUR_VERIFY	Верификация функциональности защищенности
SW_INTGT	Программная и информационная целостность
INPUT_VALD	Валидация входов
DET_OUT	Детерминированные выходы
ERR_HNDL	Обработка ошибок
SESS_INTGT	Безошибочность сессий
INFO_CONFI	Конфиденциальность информации
INFO_PERS	Сохраняемость информации
CRYTO	Применение криптографии
RSTIC_FLOW	Ограниченный поток данных
DoS	Защита от отказа в обслуживании
RESOU_MGT	Управление ресурсами
CNTR_RECOV_RECON	Исправление и восстановление системы контроля

Таблица 4.36 — Перечисление SecurityLevel

Описание: Представление уровня защищенности		
Атрибуты:		
	Киберзащищенность	Физическая защищенность
0	None. Нет уровня защищенности	None. Нет уровня защищенности
1	Значения, определенные в МЭК 62443-4-2:2019	LatchSensor
2		LockwithKey
3		LockwithActuator
4		Не определен

Таблица 4.37 — Класс CyberSecurity

Описание: Класс киберзащищенности				
Выведен из: Не применимо				
Атрибуты:				
securityType	SecurityType	M	1	См. таблицы 4.35 и 4.29
eachSecurityLevel	SecurityLevel	M	1	См. таблицу 4.36

Таблица 4.38 — Класс SafeSecure

Описание: Класс для безопасности и защищенности модуля. Если модуль обеспечивает единственную функцию безопасности, то ее значение равно значению атрибута eachSafetyLevel функции безопасности. По крайней мере одно из этих двух значений должно быть предоставлено				
Выведен из: Не применимо				
Атрибуты:				
overallValidSafetyLevelType	PLSILType	M	1	Форма представления уровня общей безопасности (см. таблицу 4.33)
overallSafetyLevelPL	SafetyLevelPL	M	1	Установить PL, если атрибут overallValidSafetyLevelType равен PL или BOTH (см. таблицу 4.31)

Окончание таблицы 4.38

overallSafetyLevelSIL	SafetyLevelSIL	M	1	Установить SIL, если атрибут overallValidSafetyLevelType равен SIL или BOTH (см. таблицу 4.32)
overallPhySecurityLevel	SecurityLevel	M	1	Общая физическая защищенность модуля. См. таблицу 4.36
overallCybSecurityLevel	SecurityLevel	M	1	Общая киберзащищенность модуля. См. таблицу 4.37
inSafetyLevel	SafetyFunction	O	N	УЭЗ отдельной функции безопасности. См. таблицу 4.34
inCybSecurityLevel	CyberSecurity	O	N	Отдельная функция защищенности. См. таблицу 4.37
additionalInfo	NVList	O	1	Список NameValue для дополнительной информации. См. таблицу 4.14

4.3.9 Класс для моделирования

Класс Modelling должен представлять поддерживаемую модулем информацию, относящуюся к моделированию, как показано на рисунке 4.12 и в таблицах 4.39 и 4.40. На рисунке 4.12 представлена взаимосвязь между классами для класса Modelling.

Информация по моделированию должна содержать информацию, предоставляемую модулем для моделирования, состоящую из информации для геометрического моделирования и информации для динамического или поведенческого моделирования с физическими и/или логическими характеристиками. Примерами первой являются трехмерные модели и файлы в универсальном формате описания роботов (URDF). Примерами второй являются запускающие и/или управляющие программы для модуля. Данная информация должна быть представлена в соответствии с приложением С. Изготовители модулей могут предоставить несколько файлов для моделирования. При отсутствии URDF-файлов или файлов с трехмерными моделями соответствующий элемент должен быть помечен как «N/A» (не применимо).

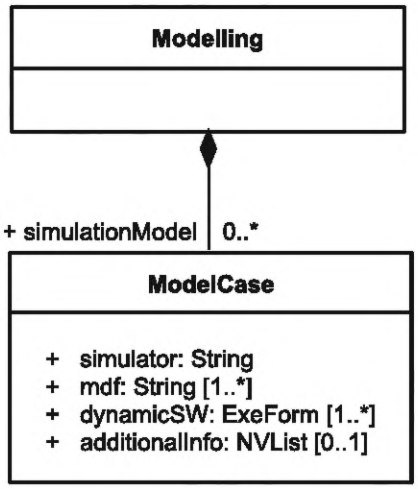

Рисунок 4.12 — Взаимосвязь между классами для класса Modelling

Таблица 4.39 — Класс ModelCase

Описание: Используется для указания путей к файлам, содержащим модель для моделирования модуля				
Выведен из: Не применимо				
Атрибуты:				
simulator	String	M	1	Список программ моделирования, поддерживаемых модулем

Окончание таблицы 4.39

mdf	String	M	N	Путь к файлу описания модели или путь к папке, содержащей файлы описания модели, либо «N/A». Файл описания модели может содержать трехмерную модель
dynamicSW	ExeForm	M	N	Динамические программы и/или управляющие программы для моделирования, связанного с модулем
additionalInfo	NVList	O	1	Список NameValue для дополнительной информации. См. таблицу 4.14

Таблица 4.40 — Класс Modelling

Описание: Используется при наличии нескольких симуляторов, но трехмерная модель может быть использована только на одном симуляторе				
Выведен из: Не применимо				
Атрибуты:				
simulationModel	ModelCase	O	N	Обеспечивает информацию об имитационных моделях, используемых в модуле. См. таблицу 4.39

4.3.10 Класс для исполняемой формы

Класс ExecutableForm должен представлять информацию, связанную с программой, исполняемой для достижения или поддержки назначения модуля, как показано на рисунке 4.13 и в таблице 4.41. На рисунке 4.13 представлена взаимосвязь между классами для класса ExecutableForm, который представлен в таблицах 4.41 и 4.42. Класс ExecutableForm представляет пути к файлам, которые необходимы при исполнении модуля или когда другие модули исполняют программу из данного модуля. Все эти файлы подразделяются на два типа: файлы, которые должны быть исполнены непосредственно, представлены классом ExeForm в таблице 4.42 и библиотечные файлы, которые загружаются в оперативную память. Путь к файлам должен быть определен в формате URL, который представляется только профилем. Это означает, что пользователь может модифицировать URL, изменяя содержимое профиля.

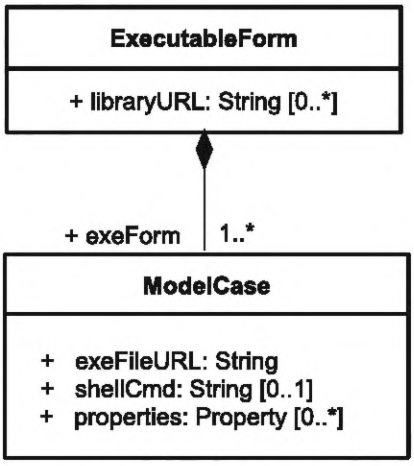

Рисунок 4.13 — Взаимосвязь между классами для класса ExecutableForm

Таблица 4.41 — Класс ExecutableForm

Описание: Представляет информацию, связанную с программой, исполняемой для достижения или поддержки назначения модуля. Должны быть указаны свойства, такие как используемая операционная система, которые определены в ИСО 22166-202				
Выведен из: Не применимо				
Атрибуты:				
exeForm	ExeForm	M	N	См. таблицу 4.42

Окончание таблицы 4.41

libraryURL	String	O	N	Двоичный код или исходный код, использованный в модуле. Для исходного кода требуется интерпретатор. Если библиотека не нужна, то значением данного атрибута должно быть NULL

Таблица 4.42 — Класс ExeForm

Описание: Представляет информацию, связанную с программой, которая должна быть исполнена для модуля				
Выведен из: Не применимо				
Атрибуты:				
exeFileURL	String	M	1	Путь к исполняемому файлу, который должен быть выполнен
shellCmd	String	O	1	Командная строка для выполнения «exeFileURL», заданная как информация о данном модуле. Данная командная строка необходима, если программа «exeFileURL» использует какие-либо входные данные из командной строки. Если никакие данные не требуются, то значением данного аргумента должно быть NULL
properties	Property	O	N	См. таблицу 4.8. Свойства и их значения, необходимые при выполнении программы «exeFileURL». Если никакие свойства не требуются, то значением данного аргумента должно быть NULL

Класс ExeForm представляет информацию, необходимую при выполнении исполняемой формы модуля, которая включает URL для исполняемой формы, программную оболочку и свойства. Программная оболочка содержит информацию, используемую в качестве входных аргументов при выполнении связанной с ней исполняемой формы. Свойства содержат информацию, необходимую при выполнении исполняемой формы, к которой они относятся.

Приложение А
(обязательное)

Правила присвоения имен

А.1 Общие положения

Правила присвоения имен выведены на основе руководств и принципов, описанных в ИСО 11179-5:2015. Данные правила были приспособлены для конкретного применения в информационной модели. Они разработаны для обеспечения единообразного и адекватного именования как базовых, так и агрегированных информационных объектов.

Базовый информационный объект используется для представления информации и построения информационных моделей, а его примерами являются класс объекта, атрибут класса объекта, элемент тега и элемент свойства. Он определен как Name (Имя элемента), являющееся именем объекта, полученным с помощью правил присвоения имен.

Агрегированный информационный объект определен как имя, состоящее из имен базовых информационных объектов.

А.2 Правила присвоения имен

Правило 1: Имя элемента должно быть уникальным в логической группировке данных, к которой принадлежит данный элемент.

Правило 2: Имя класса допускается начинать с прописной буквы. Имя члена класса рекомендуется начинать со строчной буквы.

Правило 3: Имя класса не должно содержать последовательных дублирующих слов.

Правило 4: Рекомендуется, чтобы имя элемента содержало менее 10 символов при максимальной длине менее 256 символов. При необходимости сокращение имени элемента может быть определено и использовано.

Правило 5: Имя элемента должно разделяться точкой (.).

Правило 6: В имени элемента могут быть использованы буквенно-цифровые символы, если они не определены особо.

Правило 7: Имя элемента должно быть представлено в единственном числе, если только его смысл не предполагает множественного числа (например, goods).

Правило 8: Имя элемента может состоять из двух или более слов (например, aggregate.information).

Некоторые примеры применения правил присвоения имен представлены на рисунке А.1.

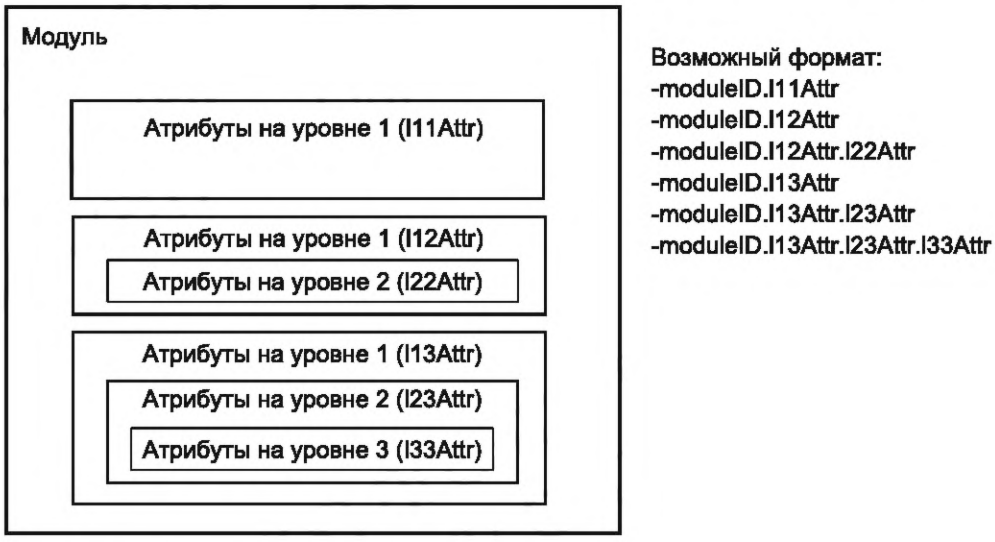

Рисунок А.1 — Взаимосвязь модуля и атрибутов на каждом уровне

Приложение В
(обязательное)

Правило назначения идентификатора модуля

Идентификатор модуля должен состоять из 6 элементов, которые представлены на рисунке В.1, а их типы представлены в таблице В.1.

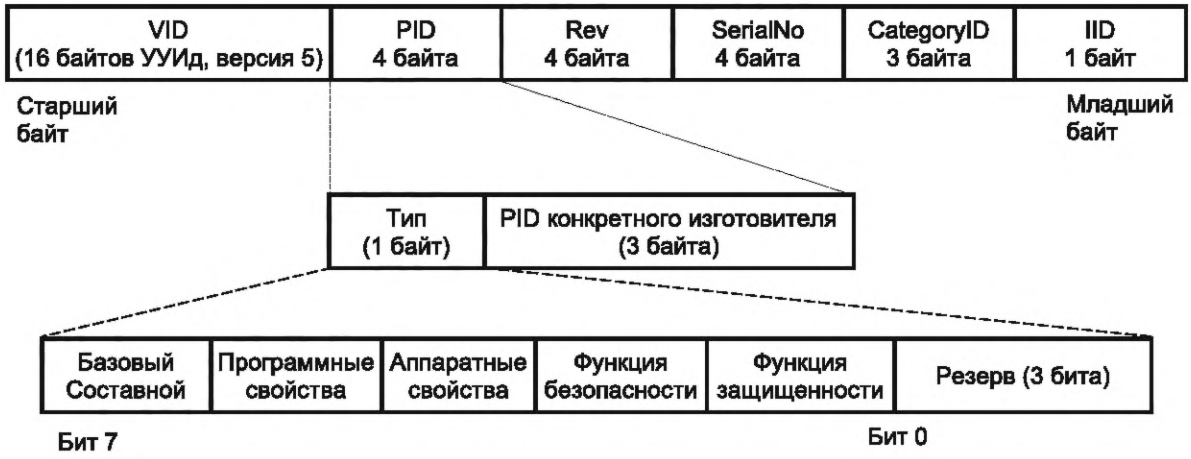

Рисунок В.1 — Состав идентификатора модуля

Таблица В.1 — Типы данных элементов идентификатора модуля

Имя	Тип данных	Длина, байты	Примечание
VID	UUID	16	Идентификатор изготовителя
PID	unsigned char	4	Идентификатор и свойства изделия
Rev	unsigned char	4	Номер версии
SerialNo	unsigned long	4	Серийный номер
CategoryID	unsigned long	3	Идентификатор категории
IID	unsigned char	1	Идентификатор экземпляра

Первым элементом является идентификатор изготовителя или VID, который имеет длину 16 байтов и должен быть назначен с помощью Универсально Уникального Идентификатора (UUID) в соответствии с IETF RFC 4122. Остальные 5 элементов должны быть заданы изготовителем.

Вторым элементом является идентификатор изделия или PID, который имеет длину 4 байта и должен содержать важную информацию о модуле (1 байт), и идентификатор изделия (3 байта). Первый байт должен отображать следующие 5 типов информации, представленные на рисунке В.1: является ли модуль базовым или составным, имеет ли модуль программные свойства, имеет ли модуль аппаратные свойства, выполняет ли модуль функцию, связанную с безопасностью, и выполняет ли модуль функцию, связанную с защищенностью. Значения каждого из этих полей представлены в таблице В.2. Последние 3 байта должны содержать фактический идентификатор изделия, состоящий из алфавитно-цифровых символов.

Третьим элементом должен быть номер версии (Rev) длиной 4 байта.

Четвертым элементом должен быть серийный номер (SerialNo) длиной 4 байта.

Пятым элементом должен быть идентификатор категории (CategoryID) модуля длиной 3 байта.

Шестым элементом должен быть идентификатор экземпляра (IID) длиной 1 байт, представленный числовыми символами в диапазоне от 0 до 255. Значением по умолчанию идентификатора экземпляра является 0. Если модуль имеет два или более модулей одного типа, то эти модули должны иметь разные идентификаторы экземпляра.

Все данные в идентификаторе модуля представляют в шестнадцатеричном и/или алфавитно-цифровом коде.

Таблица В.2 — Значения всех полей элемента Тип на рисунке В.1

Поле	Значение	Положение в байте
Базовый/Составной	0, если модуль базовый 1, если модуль составной	Бит 7 (старший бит)
Программные свойства	1, если есть программные свойства 0, если их нет	Бит 6
Аппаратные свойства	1, если есть аппаратные свойства 0, если их нет	Бит 5
Функция безопасности	1, если модуль выполняет функцию, связанную с безопасностью 0, если не выполняет	Бит 4
Функция защищенности	1, если модуль выполняет функцию, связанную с защищенностью 0, если не выполняет	Бит 3
Резерв	—	Биты 2—0

Идентификатор категории должен состоять из 5 элементов, представленных на рисунке В.2. Нулевой уровень, представленный на рисунке В.2, является верхним уровнем и классифицируется по 4 категориям, представленным в таблице В.3. Категории с первого уровня по четвертый уровень будут определены в последующих стандартах, таких как ИСО 22166-202.

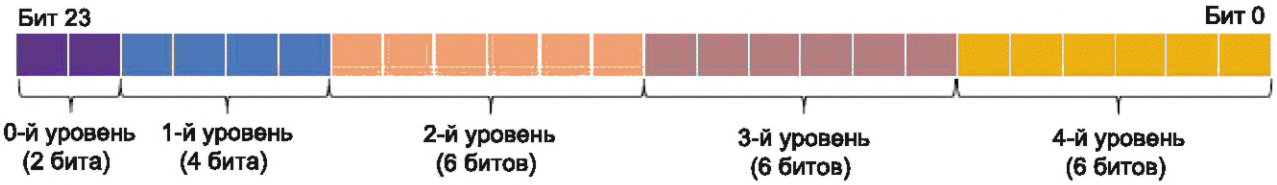

Рисунок В.2 — Поля идентификатора категории

Таблица В.3 — Классификация для нулевого уровня идентификатора категории на рисунке В.2

Классификация	Значение	Примечание
Модуль с аппаратными и программными свойствами	00	—
Аппаратный модуль	01	—
Программный модуль	10	—
Инструменты разработки/тестирования	11	Инструменты/модули для разработки/анализа/тестирования

Приложение С
(обязательное)

Представление общей информации

С.1 Общие положения

В данном приложении представлена информация, которую следует использовать при создании объекта для класса, определенного в разделе 4. Данная информация может быть представлена на языках XML или JSON. В настоящем стандарте использовано представление на языке XML. Для представления информации об ОИМ классе следует использовать приведенную ниже XML-схему, после которой приведен пример представления информации для ОИМ модуля на основе данной XML-схемы. Описания имен элементов представлены в таблице С.1. Суб-элементы представлены в таблицах С.2—С.8.

XML-схема для ОИМ модуля:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="GenInfo" type="xs:string"/>
  <xs:element name="IDnType" type="xs:string"/>
  <xs:element name="Properties" type="xs:string"/>
  <xs:element name="IOVariables" type="xs:string"/>
  <xs:element name="Services" type="xs:string"/>
  <xs:element name="Infra" type="xs:string"/>
  <xs:element name="SafeSecure" type="xs:string"/>
  <xs:element name="Modelling" type="xs:string"/>
  <xs:element name="Module">
    <xs:annotation>
      <xs:documentation>version = "1.1" </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="GenInfo"/>
        <xs:element ref="IDnType"/>
        <xs:element ref="Properties"/>
        <xs:element ref="IOVariables"/>
        <xs:element ref="Services"/>
        <xs:element ref="Infra"/>
        <xs:element ref="SafeSecure"/>
        <xs:element ref="Modelling"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример представления информации для ОИМ модуля на основе XML-схемы для ОИМ модуля:

```
<?xml version="1.0" encoding="UTF-8" ?>   <!-- version = "1.1" OK -->
<Module>
  <GenInfo>     </GenInfo>      <!-- See C.2 -->
  <IDnType>     </IDnType>      <!-- See C.3 -->
  <Properties>  </Properties>   <!-- See C.4 -->
  <IOVariables> </IOVariables>  <!-- See C.4 -->
  <Services>    </Services>     <!-- See C.6 -->
  <Infra>       </Infra>        <!-- See C.7 -->
  <SafeSecure>  </SafeSecure>   <!-- See C.8 -->
  <Modelling>   </Modelling>    <!-- See C.9 -->
</Module>
```

Таблица С.1 — XML-элемент для ОИМ модуля

Имя элемента	Описание
Module	Все характеристики модуля определены в данном элементе. Данный элемент описывает корневой элемент представления информации о модуле

С.2 Общая информация о модуле

XML-элементы, содержащие информацию для GenInfo, должны соответствовать таблице С.2. Для представления информации об элементах из таблицы С.2 следует использовать приведенную ниже XML-схему, после которой приведены примеры информации для GenInfo на языке XML.

Таблица С.2 — XML-элементы для GenInfo

Имя элемента	Описание
GenInfo	Все характеристики определены в данном элементе. Данный элемент описывает общую информацию о модуле
ModuleName	Данный элемент определяет имя модуля
Description	Данный элемент содержит общую информацию о модуле
Manufacturer	Данный элемент определяет информацию об изготовителе модуля, например контактную информацию и наименование
Examples	Данный элемент содержит примеры типичного применения модуля

XML-схема для GenInfo:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="ModuleName" type="xs:string"/>
  <xs:element name="Description" type="xs:string"/>
  <xs:element name="Manufacturer" type="xs:string"/>
  <xs:element name="Examples" type="xs:string"/>
  <xs:element name="GenInfo">
    <xs:annotation>
      <xs:documentation>version = "1.1" OK</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ModuleName"/>
        <xs:element ref="Description"/>
        <xs:element ref="Manufacturer"/>
        <xs:element ref="Examples"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример представления информации для GenInfo на основе на XML-схемы для GenInfo:

```
<?xml version="1.0" encoding="UTF-8" ?>                <!-- version = "1.1" OK -->
<GenInfo>
  <ModuleName> Module Name </ModuleName>               <!-- string -->
  <Description> Description of module  </Description>  <!-- string -->
  <Manufacturer> manufacturer </Manufacturer>          <!-- string -->
  <Examples> list of use case </Examples>              <!-- string -->
</GenInfo>
```

С.3 Информация об идентификаторе модуля

XML-элементы, содержащие информацию о классе IDnType, должны соответствовать таблице С.3. Для представления информации об элементах из таблицы С.3 следует использовать приведенную ниже XML-схему, после которой приведены примеры информации для класса IDnType. Первый пример является примером базового модуля, а второй пример — примером составного модуля, обладающего как аппаратными, так и программными свойствами. Если существует только элемент HWList, то данный составной модуль является разновидностью аппаратного модуля. Если существует только элемент SWList, то данный составной модуль является разновидностью программного модуля. Если оба элемента, HWList и SWList, существуют одновременно, то данный составной модуль является разновидностью аппаратно-программного модуля.

Таблица С.3 — XML-элементы для идентификатора модуля

Имя элемента	Описание
IDnType	Идентификатор модуля, версия информационной модели, списки всех модулей, использованных в модуле, определены в данном элементе. Список модулей указывают только для составного модуля
ID	Данный элемент определяет уникальный регистрационный номер изделия, присваиваемый изготовителем модуля (см. приложение В). Данный элемент может иметь атрибут «type» для указания типа модуля, такого как базовый модуль или составной модуль. Если его значением является «Bas» или данный атрибут не определен, то данный модуль является базовым модулем. Если его значением является «Com», то данный модуль является составным модулем
InformationModelVersion	Данный элемент определяет номер версии использованной информационной модели
HWList	Данный элемент используется для модулей с аппаратными и программными свойствами, а также для аппаратных модулей в составном модуле. Он содержит список идентификаторов модулей, входящих в составной модуль. Данный элемент имеет один или несколько атрибутов «ModuleID», определяющих идентификаторы модулей, входящих в составной модуль
SWList	Данный элемент используется для программных модулей в составном модуле. Он содержит список идентификаторов модулей, входящих в данный составной модуль. Данный элемент имеет один или несколько атрибутов «ModuleID», определяющих идентификаторы модулей, входящих в данный составной модуль

XML-схема для идентификатора модуля:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="ModuleID" type="xs:string"/>
  <xs:element name="ID" type="xs:string"/>
  <xs:element name="InformationModelVersion" type="xs:float"/>
  <xs:element name="HWlist">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ModuleID" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SWlist">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ModuleID" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="IDnType">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ID"/>
        <xs:element ref="InformationModelVersion"/>
        <xs:element ref="HWlist"/>
        <xs:element ref="SWlist"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="type"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример представления информации для идентификатора базового модуля на основе XML-схемы для идентификатора модуля:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IDnType type="Bas"> <!-- example of basic module -->
  <ID> moduleId </ID>   <!-- provided by manufacturer and defined in Annex B-->
  <InformationModelVersion> 1.0 </InformationModelVersion>  <!-- string -->
</IDnType>
```

Пример представления информации для идентификатора составного модуля на основе XML-схемы для идентификатора модуля:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <IDnType type="Com">   <!-- example of a composite module -->
 <ID> moduleId  </ID><!-- ID for the composite module -->
 <InformationModelVersion> 1.0 </InformationModelVersion> <!-- string -->
 <HWlist> <!-- in the list, IIDs of modules are not defined -->
   <ModuleID>hwModuleID1  </ModuleID>
   <ModuleID>hwModuleID2  </ModuleID>
   <ModuleID>hwModuleID3  </ModuleID>
   <ModuleID>hwModuleID4  </ModuleID>
 </HWlist>
 <SWlist>
   <ModuleID>swModuleID1  </ModuleID>
   <ModuleID>swModuleID2  </ModuleID>
   <ModuleID>swModuleID3  </ModuleID>
   <ModuleID>swModuleID4  </ModuleID>
 </SWlist>
</IDnType>
```

С.4 Информация о классах Properties и IOVariables

Большая часть информации в классах Properties и IOVariables относится к исполнению модулей и используется в качестве параметров и/или переменных в программных свойствах. Для аппаратных свойств необходимо показать некоторые типичные применения и привести примеры информации о входных и выходных переменных.

В отношении информации для класса Properties, определенного в 4.3.3, обязательно должны быть указаны имя и значение характеристики, а также ее тип данных. Единицы измерения данного значения и относящееся к нему описание могут быть включены для пояснения использования.

Информация для класса IOVariables, определенного в 4.3.4, включает имя, значение и тип данных переменной, а также тип входных и выходных данных. Единицы измерения данного значения и относящееся к нему описание могут быть включены для пояснения использования. Следует обратить внимание на то, что переменные, которые одновременно могут быть использованы в качестве входной и выходной переменной, должны быть помечены как «входная/выходная».

XML-элементы, содержащие информацию об элементах Properties, Inputs и Outputs, должны соответствовать таблице С.4. Для представления информации об элементах из таблицы С.4 следует использовать приведенные ниже XML-схемы для классов Properties и IOVariables, после которых приведен пример информации для классов Properties и IOVariables.

Таблица С.4 — XML-элементы для классов Properties и IOVariables

Имя элемента	Описание
Properties	Все относящиеся к модулю характеристики определены в данном элементе. Примечание — Если требуется пространство имен, то оно будет определено в последующих стандартах, таких как ИСО 22166-202.
Property	Данный элемент определяет отдельную характеристику элемента «Properties». Данный элемент может иметь атрибут «Complex» для данных типа массива и комплексных типов данных, таких как структуры и классы. Если данный атрибут имеет значение «array» или «class», то элемент «Property» будет иметь структуру массива или класса. Если данный аргумент не определен, то элементу «Property» соответствует простой тип данных
IOVariables	В данном элементе определены все переменные, которые подразделяются на три типа: входные, выходные и входные/выходные. Данный элемент может иметь атрибут «Name», определяющий имя домена. Тогда все переменные, определенные в элементе «IOVariables», будут относиться к тому же домену

Окончание таблицы С.4

Имя элемента	Описание	
Inputs	Данный элемент определяет входные переменные	
Outputs	Данный элемент определяет выходные переменные	
InOuts	Данный элемент определяет переменные, которые могут одновременно выступать в качестве входных и выходных переменных	
Input	Данный элемент используется для переменной, заданной в элементе «IOVariables» только как входная переменная	Данный элемент может иметь атрибут «Complex» для данных типа массива и комплексных типов данных, таких как структуры и классы. Если данный атрибут имеет значение «array» или «class», то переменные имеют структуру массива или класса. Если данный аргумент не определен, то для него может быть определен новый простой тип данных в классе DataProfile (см. таблицу 4.7)
Output	Данный элемент используется для переменной, заданной в элементе «IOVariables» только как выходная переменная	
InOut	Данный элемент используется для переменной, заданной в элементе «IOVariables» как одновременно входная и выходная переменная	
name	Данный элемент определяет имя характеристики или переменной, используемой в модуле (например, encoder, ratedCurrent, MaxCurrent, P_Coeff)	
value	Данный элемент используется при инициализации характеристики или переменной. Если характеристика или переменная имеет тип «array», то данный атрибут имеет одно или несколько значений, задаваемых с помощью атрибута «Item»	
type	Данный элемент определяет тип данных характеристики или переменной; примерами являются Boolean, Integer, UnlimitedNatural, Real, String, Enumeration, void и any, представленные в таблице С.5	
complex	Данный элемент определяет типы данных array, class, vector или pointer из таблицы С.5, примерами которых являются Boolean, Integer, UnlimitedNatural, Real, String, Enumeration, void и any	
complexName	Данный элемент определяет имя комплексного типа данных; например, в объявлении «cylinderType class {float32 length; float32 radius;} cylinder;» элементу complexName соответствует «cylinderType», а элементу name соответствует «cylinder»	
unit	Данный элемент определяет единицу измерения характеристики или переменной (например, ампер, метр, Цельсий, безразмерный и т. д.)	
description	Данный элемент содержит пояснение, относящееся к характеристике или переменной.	
additionalInfo	Данный элемент содержит дополнительную информацию о переменных. Он может не иметь ни одного или иметь несколько значений, задаваемых с помощью атрибута «Item», каждое значение которого состоит из пары «name» (имя) и «value» (значение)	

В таблице С.5 представлен абстрактный тип данных, используемый в определении классов, и типы данных как ключевые слова языка XML, которые должны быть использованы в настоящем стандарте. Если типом данных элемента является перечислимый тип данных, то следует использовать новый атрибут «enumeration». Элемент «unit» означает размерную характеристику физической величины, примерами которой являются метр, обороты в минуту, секунда, бит в секунду, ампер, вольт, ватт и Цельсий. В частности, элемент «unit» имеет специальную единицу «none», которая означает отсутствие физической величины.

Таблица С.5 — Абстрактный тип данных для определения классов и типы данных как ключевые слова языка XML

Абстрактный тип данных, используемый при определении классов	Типы данных как ключевые слова языка XML	Описание
Boolean	bool	Истина(1) или Ложь(0)
Integer	int	Целочисленный тип данных по умолчанию в зависимости от ОС

Окончание таблицы С.5

Абстрактный тип данных, используемый при определении классов	Типы данных как ключевые слова языка XML	Описание
Integer	int8	Целое со знаком длиной 1 байт
	int16	Целое со знаком длиной 2 байта
	int32	Целое со знаком длиной 4 байта
	int64	Целое со знаком длиной 8 байтов
UnlimitedNatural	uint	Целочисленный тип данных без знака по умолчанию в зависимости от ОС
	uint8	Целое без знака длиной 1 байт
	uint16	Целое без знака длиной 2 байта
	uint32	Целое без знака длиной 4 байта
	uint64	Целое без знака длиной 8 байтов
Real	Float32	Число с плавающей запятой длиной 4 байта
	Float64	Число с плавающей запятой длиной 8 байтов
String	string	Последовательность алфавитно-цифрового текста или других символов, разновидность класса
Enumeration	enumeration	Перечислимый тип данных
void	void	Тип данных, который не имеет значений и используется для представления пустого множества
class	class	Комплексный тип данных, такой как структура или класс, который должен иметь имя
any	any	Используется для хранения в переменной значений любого типа
Не применимо	array	Тип данных массив, разновидность комплексного типа данных
	pointer	Тип данных указатель, разновидность комплексного типа данных
	vector	Совокупность объектов, разновидность класса

XML-схема для класса Properties:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="value">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Item" type="xs:float"/>
  <xs:element name="Property">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="value" minOccurs="0"/>
        <xs:element ref="Property" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="name" use="optional"/>
```

```
      <xs:attribute type="xs:string" name="type" use="optional"/>
      <xs:attribute type="xs:string" name="value" use="optional"/>
      <xs:attribute type="xs:string" name="unit" use="optional"/>
      <xs:attribute type="xs:string" name="description" use="optional"/>
      <xs:attribute type="xs:string" name="complex" use="optional"/>
      <xs:attribute type="xs:string" name="complexName" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Properties">
    <xs:annotation>
      <xs:documentation>example of class type property</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Property" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML-схема для класса IOVariables:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Value" type="xs:float"/>
  <xs:element name="Item">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="Value"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="type" type="xs:string"/>
  <xs:element name="value" type="xs:float"/>
  <xs:element name="unit" type="xs:string"/>
  <xs:element name="description" type="xs:string"/>
  <xs:element name="additionalInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Input">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="name" minOccurs="0"/>
        <xs:element ref="type" minOccurs="0"/>
        <xs:element ref="value" minOccurs="0"/>
        <xs:element ref="unit" minOccurs="0"/>
        <xs:element ref="description" minOccurs="0"/>
        <xs:element ref="additionalInfo" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="name" use="optional"/>
      <xs:attribute type="xs:string" name="type" use="optional"/>
      <xs:attribute type="xs:float" name="value" use="optional"/>
      <xs:attribute type="xs:string" name="unit" use="optional"/>
      <xs:attribute type="xs:string" name="description" use="optional"/>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="Output">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="name" minOccurs="0"/>
        <xs:element ref="type" minOccurs="0"/>
        <xs:element ref="unit" minOccurs="0"/>
        <xs:element ref="description" minOccurs="0"/>
        <xs:element ref="additionalInfo" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="name" use="optional"/>
      <xs:attribute type="xs:string" name="type" use="optional"/>
      <xs:attribute type="xs:string" name="unit" use="optional"/>
      <xs:attribute type="xs:string" name="description" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Inout">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="name" minOccurs="0"/>
        <xs:element ref="type" minOccurs="0"/>
        <xs:element ref="unit" minOccurs="0"/>
        <xs:element ref="description" minOccurs="0"/>
        <xs:element ref="additionalInfo" minOccurs="0"/>
        <xs:element ref="Inout" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="name" use="optional"/>
      <xs:attribute type="xs:string" name="type" use="optional"/>
      <xs:attribute type="xs:string" name="unit" use="optional"/>
      <xs:attribute type="xs:string" name="description" use="optional"/>
      <xs:attribute type="xs:string" name="className" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Inputs">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Input" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Outputs">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Output" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="InOuts">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Inout"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="IOVariables">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Inputs"/>
        <xs:element ref="Outputs"/>
        <xs:element ref="InOuts"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример представления информации для классов Properties и IOVariables на основе XML-схем для классов Properties и IOVariables:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Properties> <!-- properties -->
 <Property name="maxRatedCurrent" type="float32" unit="ampere" description = "maximum of
rated current for motor"  value ="15" />  <!-- example of single property -->
 <Property name="maxRatedVoltage" type="float32" unit="volt" description =
"maximum of rated voltage for motor">
   <value> 5 </value>
 </Property>   <!--  example of single property -->
 <Property name="StatusEmerg" type="boolean" unit="none" description =
"existence of emergency stop button"  value ="True" />
 <Property name="SizeBd" type="uint8" unit="mm" description = "size of board" value ="30"
/>
 <Property name="Weight" type="uint16" unit="gram" description = "weight of product"
value ="450000" />
 <Property name="NoPixelDisplay" type="uint32" unit="none" description = "pixel number"
value ="345600" />
 <Property name="NoPixelCamera" type="uint64" unit="none" description = "pixel number"
value ="8294400" />
 <!-- following is example of array: float32 [6] initPos  -->
 <Property name="initPose" complex ="array" type="float32" unit="none"
description="initial pose, x,y,z, yaw,roll,pitch, in order. Unit of yaw, roll, and pitch = degree " >
   <value>
     <Item>  0 </Item>
     <Item>  0 </Item>
     <Item>  0 </Item>
     <Item>  10 </Item>
     <Item>  20 </Item>
     <Item>  10 </Item>
   </value>
 </Property>   <!--  example of array type property -->
<!-- following is example of class: -->
<!-- class cylinderType { -->
<!-- float32 length; -->
<!-- float32 radius  -->
<!-- } cylinder;  -->
 <Property name="cylinder" complex = "class" complexName="cylinderType">
   <Property name="length" type="float32" value = "100.0" unit="millimetre"
description = "length of cylinder" />
   <Property name="radius" type="float32" value = "3.0" unit="millimetre"
description = "radius of cylinder" />
 </Property>   <!--  example of class type property -->
</Properties>

<IOVariables>
 <Inputs>
  <Input name="controlvalue1" type="float32" value="5.0" unit="ampere"
description = "current control command to motor" />
   <Input>
    <name> controlvalue2 </name>
    <type> float32 </type>
    <value> 5.0 </value>
    <unit> voltage </unit>
    <description> voltage control of motor </description>
    <additionalInfo>
       <Item>
          <Name> min Voltage   </Name>
          <Value> -5.0 </Value>
       </Item>
       <Item>
          <Name> max voltage   </Name>
```

```
          <Value> 5.0 </Value>
        </Item>
    </additionalInfo>
   </Input>
 </Inputs>
 <Outputs>
  <Output name="encoder" type ="uint32" unit = "none" description = "state of motor" />
  <Output>
   <name> Status </name>
   <type> uint8 </type>
   <unit> None </unit>
   <description> Status </description>
   <additionalInfo>
      <Item>
         <Name> IDLE  </Name>
         <Value> 0 </Value>
      </Item>
      <Item>
         <Name> WORKING  </Name>
         <Value> 1 </Value>
      </Item>
      <Item>
         <Name> ERROR  </Name>
         <Value> 2 </Value>
      </Item>
   </additionalInfo>
  </Output>
 </Outputs>
 <InOuts>
  <Inout name="pose" type = "" className="poseType">
    <Inout name="x" type="float32" unit="none" description = "position x" />
    <Inout name="y" type="float32" unit="none" description = "position y" />
    <Inout name="z" type="float32" unit="none" description = "position z" />
    <Inout name="yaw" type="float32" unit="degree" description = "yaw" />
    <Inout name="roll" type="float32" unit="degree" description = " roll" />
    <Inout>
      <name> pitch </name>
      <type> float32 </type>
      <unit> degree </unit>
      <description> pitch </description>
      <additionalInfo>
        <Item>
           <Name> min </Name>
           <Value> -100 </Value>
        </Item>
        <Item>
           <Name> max </Name>
           <Value> 100 </Value>
        </Item>
     </additionalInfo>
   </Inout>
  </Inout>
 </InOuts>
</IOVariables>
```

С.5 Информация о сервисах модуля

XML-элементы, содержащие информацию о классе Services, должны соответствовать таблице С.6. Для представления информации о классе Services на основе таблицы С.6 следует использовать приведенную ниже XML-схему, после которой приведены два примера информации для класса Services (или Functions/Capabilities), один из которых представлен на языке IDL, а второй — на языке XML.

Данная информация обычно используется в программах, написанных на таких языках, как C/C++, Java и Python. Поэтому важно определить порядок аргументов и получить дополнительные возвращаемые через

аргументы значения. Типом данных аргументов, используемых в качестве дополнительных возвращаемых значений, является pointer (указатель) в языке C/C++ или reference type (ссылочный тип) в языках Java и Python.

Таблица С.6 — XML-элементы для класса Services

Имя элемента	Описание
Services	В данном элементе определена вся информация для класса Services. Данный элемент описывает информацию о сервисах (или функциях/возможностях) модуля
Service	Данный элемент представляет один информационный элемент, относящийся к сервису. Данный элемент имеет атрибут «type» для определения используемого языка описания, такого как IDL или XML. Если значением атрибута «type» является IDL, то предоставляемый сервис описан на языке IDL, а значением элемента является путь к соответствующему IDL-файлу. Если значением атрибута «type» является XML, то предоставляемый сервис описан на языке XML, а значения элементов представлены в других XML-элементах
ID	Данный элемент определяет идентификатор интерфейса, который модуль использует или предоставляет
PVType	Данный элемент используется для классификации типа интерфейса модуля, значением которого является «Physical» или «Virtual». Если модуль предоставляет физический интерфейс, например механический интерфейс, то значением элемента является «Physical». Если модуль предоставляет программный интерфейс, например API (интерфейс прикладного программирования), то значением элемента является «Virtual»
MOType	Данный элемент объявляет предоставляемый интерфейс обязательным («Mandatory») или факультативным («Optional»)
Properties	Данный элемент определяет характеристики, которые модуль должен использовать (или задать) при использовании предоставленного интерфейса. Данный элемент имеет два атрибута, «Name» и «Value», которые соответствуют имени и значению характеристики данного модуля
Methods	Данный элемент определяет методы, которые должен использовать данный модуль. В данном элементе может быть представлен один или несколько методов
Method	Данный элемент определяет отдельный метод из методов, перечисленных в элементе «Methods»
MethodName	Данный элемент определяет имя предоставляемого метода
ArgSpec	Данный элемент определяет список аргументов, каждый из которых имеет 3 атрибута: «Type», «Name» и «InOutType». Атрибут «Type» представляет тип данных аргумента, определенный в таблице С.5. Атрибут «Name» представляет имя аргумента данного метода. Атрибут «IOType» представляет направление, в котором передается данный аргумент и которое может быть входным, выходным или двунаправленным. Если атрибут «Type» представляет комплексное число, то необходимо использовать дополнительный атрибут «inDataType» для того, чтобы определить тип данных действительной части, содержащейся в комплексном типе данных, таком как массив или класс
RetType	Данный элемент определяет тип данных, возвращаемых данным методом
additionalInfo	Данный элемент содержит дополнительную информацию о переменных. Он может не иметь ни одного или иметь несколько значений, задаваемых с помощью атрибута «Item», каждое значение которого состоит из пары «name» (имя) и «value» (значение)

XML-схема для класса Services:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Value" type="xs:string"/>
  <xs:element name="Item">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="Value"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="MethodName">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="ArgSpec">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="Type" minOccurs="0"/>
        <xs:element ref="Name" minOccurs="0"/>
        <xs:element ref="IOType" minOccurs="0"/>
        <xs:element ref="additionalInfo" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="Type" use="optional"/>
      <xs:attribute type="xs:string" name="Name" use="optional"/>
      <xs:attribute type="xs:string" name="IOType" use="optional"/>
      <xs:attribute type="xs:string" name="inDataType" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="RetType">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="MOType">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="ReqProvType">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="additionalInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="Type" type="xs:string"/>
  <xs:element name="IOType" type="xs:string"/>
  <xs:element name="Method">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="MethodName"/>
        <xs:element ref="ArgSpec" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="RetType" minOccurs="0"/>
        <xs:element ref="MOType"/>
        <xs:element ref="ReqProvType"/>
        <xs:element ref="additionalInfo" minOccurs="0"/>
       </xs:sequence>
     </xs:complexType>
  </xs:element>
  <xs:element name="ID" type="xs:string"/>
  <xs:element name="PVType">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="Methods">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Method" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="additionalInfo" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Properties">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="Name" use="optional"/>
          <xs:attribute type="xs:string" name="Value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="Service">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="ID" minOccurs="0"/>
        <xs:element ref="PVType" minOccurs="0"/>
        <xs:element ref="MOType" minOccurs="0"/>
        <xs:element ref="Properties" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="Methods" minOccurs="0"/>
        <xs:element ref="additionalInfo" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="type" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Services">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Service" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример информации для класса Services на основе CIMServicePackage, определенного на языке IDL в 4.3.6.

```
interface Service: CIMService
{
   uint8 initialize(int64 val1, float32 val2); //services provided by module
   uint8 finalize(int32 val1, float32 val2, int32 val3);
   uint8 read(int fd, uint8[] buf, uint16 nbytes);
}
```

Пример информации для класса Services на основе XML-схемы для класса Services:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Services>
 <Service type = "IDL">
    ./IDL/AppLayer/sdo.idl
 </Service>
 <Service type = "IDL">
    ./IDL/AppLayer/sync.idl
 </Service>
 <Service type = "IDL">
    ./IDL/DeviceDriver/CAN.idl
 </Service>
  <Service type = "XML">
   <ID> NMT </ID>ID>
   <PVType> Virtual </PVType>
   <MOType> MANDATORY </MOType>
   <Methods>
     <Method>
       <MethodName value = "NMT_SearchNode" />
       <ArgSpec Type="int8" Name="iNodeID" IOType="IN" />
       <RetType value="bool" />
       <MOType value = "MANDATORY" />
       <ReqProvType value="REQUIRED" />
       <additionalInfo>
          <Item>
             <Name> initialValue </Name>
             <Value> 0 </Value>
          </Item>
       </additionalInfo>
     </Method>
     <Method>
       <MethodName value = "NMT_AddNode" />
       <ArgSpec>
       <Type> int8 </Type>
       <Name> iNodeID </Name>
       <IOType> IN </IOType>
          <additionalInfo>
             <Item>
                <Name> minValue </Name>
                <Value> 20 </Value>
             </Item>
          </additionalInfo>
       </ArgSpec>
       <ArgSpec Type="int8" Name="iType" IOType="IN" />
       <ArgSpec Type="int" Name="iLifeTime" IOType="IN" />
       <RetType value="bool" />
       <MOType value = "MANDATORY" />
       <ReqProvType value="REQUIRED" />
     </Method>
     <Method>
       <MethodName value = "NMT_GetNodeInfo" />
       <ArgSpec Type="int8" Name="iNodeID" IOType="IN" />
       <ArgSpec Type="pointer" inDataType="int" Name="iType" IOType="OUT" />
       <ArgSpec Type="pointer" inDataType="int" Name="pTime" IOType="OUT" />
```

```
      <ArgSpec Type="pointer" inDataType="uint8" Name="pLifeTime" IOType="OUT" />
      <RetType value="bool" />
      <MOType value = "MANDATORY" />
      <ReqProvType value="REQUIRED" />
    </Method>
    <additionalInfo>
      <Item>
        <Name> InvokeFirstService </Name>
        <Value> NMT_SearchNode </Value>
      </Item>
    </additionalInfo>
  </Methods>
  <additionalInfo>
   <Item>
      <Name> CANID </Name>
      <Value> 0FF </Value>
   </Item>
  </additionalInfo>
</Service>
<Service type = "XML">
 <ID> RS485Driver </ID>
 <PVType value="Virtual" />
 <MOType> OPTIONAL </MOType>
 <Properties Name = "Baud" Value ="9600" />
 <Properties Name = "STOPBit" Value ="No" />
 <Methods>
   <Method>
      <MethodName value = "initialize" />
      <ArgSpec Type="int64" Name="val1" IOType="IN" />
      <ArgSpec Type="float32" Name="val2" IOType="IN" />
      <RetType value="uint8" />
      <MOType value = "MANDATORY" />
      <ReqProvType value="PROVIDED" />
   </Method>
   <Method>
      <MethodName value = "initialize" />
      <ArgSpec Type="int64" Name="val1" IOType="IN" />
      <ArgSpec Type="float32" Name="val2" IOType="IN" />
      <ArgSpec Type="float32" Name="val3" IOType="IN" />
      <RetType value="uint8" />
      <MOType value = "OPTIONAL" />
      <ReqProvType value="PROVIDED" />
   </Method>
   <Method>
      <MethodName value = "finalize" />
      <ArgSpec Type="int32" Name="val1" IOType="IN" />
      <ArgSpec Type="float32" Name="val2" IOType="IN" />
      <ArgSpec Type="int32" Name="val3" IOType="IN" />
      <RetType value="uint8" />
      <MOType value = "MANDATORY" />
      <ReqProvType value="PROVIDED" />
   </Method>
   <Method>
      <MethodName value = "read" />
      <ArgSpec Type="int" Name="fd" IOType="IN" />
      <ArgSpec Type="pointer" inDataType ="uint8" Name="val2" IOType="INOUT" />
      <ArgSpec Type="uint16" Name="nbytes" IOType="IN" />
      <MOType value = "MANDATORY" />
      <ReqProvType value="PROVIDED" />
      <additionalInfo>
         <Item>
            <Name> minValue </Name>
```

```
            <Value> 0 </Value>
          </Item>
        </additionalInfo>
    </Method>
  </Methods>
 </Service>
</Services>
```

С.6 Информация об инфраструктуре

Данная информация определяет тип инфраструктурной поддержки и/или защиты от неблагоприятного воздействия окружающей среды, например источники питания, шину данных и степень защиты (код IP). Шины питания зависят от вида энергии, подаваемой на модуль, включая потребляемую мощность, и от вида энергии поставляемой модулем, если это имеет место. Шина данных зависит от типа связи, подключенной к модулю (например, Ethernet, EtherCAT, CAN, USB и RS422). Степень защиты определяет код IP, соответствующий данному модулю в соответствии с МЭК 60529. XML-элементы, содержащие информацию о классе Infrastructure, должны соответствовать таблице С.7. Для представления информации о классе Infrastructure на основе таблицы С.7 следует использовать приведенную ниже XML-схему, после которой приведен пример информации для класса Infrastructure.

Примечание — Шина данных в классе Infrastructure является шиной, которую модули совместно используют в системе.

Информация об элементе Databuses включает тип протокола, скорость передачи данных и поддерживаемый API, а примерами типа протокола являются CAN2.0, EtherCAT, Ethernet и RS485.

Таблица С.7 — XML-элементы для класса Infrastructure

Имя элемента	Описание
Infra	В данном элементе определены все характеристики для класса Infrastructure. Данный элемент содержит информацию об инфраструктуре модуля
Powers	Данный элемент определяет вид энергии, которую модуль использует или поставляет. К видам энергии относятся электрическая энергия, пневматическая энергия и гидравлическая энергия. Примечание — В настоящем стандарте определен элемент ElecPower, представляющий электрическую энергию.
ElecPower	Данный элемент определяет электрическую энергию. Атрибутами данного элемента являются «value» (величина) и «unit» (единица измерения). Величина означает количество энергии, выраженное в единицах измерения, а атрибут «unit» может иметь значения «watt» (ватт) или «W» (Вт) или «mW» (мВт)
RatedPower	Данный элемент определяет номинальную мощность. Атрибутами данного элемента являются «value» (величина) и «unit» (единица измерения)
MaxPower	Данный элемент определяет максимальную мощность. Атрибутами данного элемента являются «value» (величина) и «unit» (единица измерения)
DataBuses	Данный элемент определяет шины данных, которые используются в модуле
Databus	Данный элемент определяет одну из шин данных, используемых в модуле
ConnectorType	Данный элемент определяет тип соединителя для шины данных, используемого в модуле. Значением атрибута «type» (тип), который определяет тип соединителя, используемого в модуле, является «MALE» (ВИЛКА) или «FEMALE» (РОЗЕТКА)
TypePhyMac	Данный элемент определяет типы протоколов физического и канального уровней, используемых в модуле (например, CAN, EtherCAT, Ethernet, RS232, RS485)
TypeNetTrans	Данный элемент определяет типы протоколов сетевого и транспортного уровней, используемых в модуле (например, IP, TCP, TCP/IP, CANopen). Значения данного элемента определены подробно в ИСО 22166-202
TypeApp	Данный элемент определяет типы прикладных уровней, используемых в модуле. Один или несколько прикладных сервисов могут быть приведены, например: POD, OBD, SYNC. Значения будут подробно определены в ИСО 22166-202

Окончание таблицы С.7

Имя элемента	Описание
Speed	Данный элемент определяет скорость передачи данных, которую обеспечивает модуль
DBType	Данный элемент определяет тип системы управления базой данных, используемой в модуле
IP	Данный элемент определяет код IP по МЭК 60529
additionalInfo	Данный элемент содержит дополнительную информацию о переменных. Он может не иметь ни одного или иметь несколько значений, задаваемых с помощью атрибута «Item», каждое значение которого состоит из пары «name» (имя) и «value» (значение)

XML-схема для класса Infrastructure:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Value" type="xs:float"/>
  <xs:element name="Item">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="Name" minOccurs="0"/>
        <xs:element ref="Value" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="RatedPower">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:byte" name="value"/>
          <xs:attribute type="xs:string" name="unit"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="MaxPower">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:float" name="value"/>
          <xs:attribute type="xs:string" name="unit"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="additionalInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ElecPower">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="RatedPower"/>
        <xs:element ref="MaxPower"/>
        <xs:element ref="additionalInfo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="ConnectorType" type="xs:string"/>
  <xs:element name="TypePhyMac" type="xs:string"/>
  <xs:element name="TypeNetTrans" type="xs:string"/>
  <xs:element name="TypeApp">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Speed">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:short" name="value" use="optional"/>
          <xs:attribute type="xs:string" name="unit" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="Databus">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ConnectorType"/>
        <xs:element ref="TypePhyMac"/>
        <xs:element ref="TypeNetTrans"/>
        <xs:element ref="TypeApp"/>
        <xs:element ref="Speed"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Powers">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ElecPower"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DataBuses">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Databus" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="additionalInfo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DBType" type="xs:string"/>
  <xs:element name="IP" type="xs:string"/>
  <xs:element name="Infra">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Powers"/>
        <xs:element ref="DataBuses"/>
        <xs:element ref="DBType"/>
        <xs:element ref="IP"/>
        <xs:element ref="additionalInfo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример информации для класса Infrastructure на основе XML-схемы для класса Infrastructure:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Infra>
 <Powers>
  <ElecPower>
    <RatedPower value= "43" unit = "watt" />
    <MaxPower value= "54.2" unit = "watt" />
    <additionalInfo>
      <Item>
        <Name> Max Current (unit:Ampere) </Name>
        <Value> 1 </Value>
      </Item>
    </additionalInfo>
  </ElecPower>
 </Powers>
 <DataBuses> <!-- list of data buses used in Module -->
  <Databus>
    <ConnectorType> DB9 </ConnectorType>  <!-- D-Sub 9-->
    <TypePhyMac> CAN </TypePhyMac>
    <TypeNetTrans> CANopen </TypeNetTrans>
    <TypeApp> <!-- list for app. Layer of CANopen -->
      <Item> OBD </Item>
      <Item> NMT </Item>
      <Item> SDO </Item>
      <Item> PDO </Item>
      <Item> SYNC </Item>
    </TypeApp>
    <Speed value = "1000" unit="kbps" />  <!-- 1 Mbps, unit Kbps -->
  </Databus>
  <Databus>
    <ConnectorType> RJ45 </ConnectorType> <!-- Jack for EtherNet -->
    <TypePhyMac> EtherCAT </TypePhyMac>
    <TypeNetTrans> IPTCP </TypeNetTrans>
    <TypeApp> Modbus </TypeApp>
    <Speed value = "10000" unit="kbps" /> <!-- 10 Mbps, unit Kbps -->
  </Databus>
  <additionalInfo>
    <Item>
      <Name> Maximum speed of EtherCAT </Name>
      <Value> 100000 </Value>
    </Item>
  </additionalInfo>
 </DataBuses>
 <DBType> SQL DBMS </DBType> <!-- DBMS type  -->
 <IP> IP23CH </IP><!-- Ingress Protection IEC 60529-->
 <additionalInfo>
  <Item>
    <Name> Minimum IP Code </Name>
    <Value> 32 </Value>
  </Item>
 </additionalInfo>
</Infra>
```

С.7 Информация о безопасности и защищенности

Класс SafeSecure обеспечивает информацию об уровне эффективности защиты для каждой функции безопасности и информацию, связанную с защищенностью, для каждой функции защищенности, поддерживаемой модулем. XML-элементы, содержащие информацию о классе SafeSecure, должны соответствовать таблице С.8. Для представления информации о классе SafeSecure на основе таблицы С.8 следует использовать приведенную ниже XML-схему, после которой приведен пример информации для класса SafeSecure.

Атрибут «PL» (УЭЗ) должен иметь следующие значения: {n, a, b, c, d, e}, где «n» обозначает отсутствие уровня эффективности защиты, а «a»—«e» обозначают PLa—PLe (см. ИСО 13849-1). Киберзащищенность имеет следующие значения: {0, 1, 2, 3, 4}, где «0» обозначает отсутствие мер по обеспечению защищенности, а «1»—«4»

обозначают уровень защищенности «SL1»—«SL4» (см. МЭК 62443-4-2:2019). Меры по обеспечению физической защищенности определены в ИСО 22166-1, 5.6. Уровень безопасности для каждой функции безопасности должен быть представлен с использованием УЭЗ или УПБ.

Таблица С.8 — XML-элементы для класса SafeSecure

Имя элемента	Описание
SafeSecure	В данном элементе определены все атрибуты, относящиеся к классу SafeSecure. Данный элемент содержит информацию о безопасности и защищенности модуля
Safety	Данный элемент определяет уровень эффективности защиты функции безопасности, реализуемой модулем, или уровень эффективности защиты общей безопасности модуля, если модуль реализует две или более функций безопасности
Overall	Данный элемент представляет общий уровень эффективности защиты для безопасности или киберзащищенности модуля. Атрибут «mode» (режим) имеет значение «PL» или «SIL», которое обозначает УЭЗ или УПБ, соответственно
SafetyType	Данный элемент представляет уровень эффективности защиты указанной отдельной функции безопасности, которую задают с помощью атрибута «type» (вид). Возможные значения данного атрибута представлены в графе «Тег» таблицы 4.28. Атрибут «mode» (режим) имеет значение «PL» или «SIL», которое обозначает УЭЗ или УПБ, соответственно
Security	Данный элемент определяет уровень защищенности функции защищенности, который обеспечивает модуль, или уровень общей защищенности модуля, если модуль реализует две или более функций защищенности
PhysicalSecurity	Данный элемент определяет уровень физической защищенности модуля. Значениями могут быть: None(0), LatchSensor(1), LockwithKey(2), LockwithActuator(3) (см. таблицу 4.36)
CyberSecurity	Данный элемент определяет уровень защищенности функции киберзащищенности. Значения {0, 1, 2, 3, 4} (см. таблицу 4.36)
SecType	Данный элемент представляет уровень защищенности указанной отдельной функции защищенности, которую задают с помощью атрибута «type» (вид). Возможные значения данного атрибута представлены в графе «Тег» таблицы 4.29
additionalInfo	Данный элемент содержит дополнительную информацию о переменных. Он может не иметь ни одного или иметь несколько значений, задаваемых с помощью атрибута «Item», каждое значение которого состоит из пары «name» (имя) и «value» (значение)

XML-схема для класса SafeSecure:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="Overall">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="mode" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="SafetyType">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="type" use="optional"/>
          <xs:attribute type="xs:string" name="mode" use="optional"/>
          <xs:attribute type="xs:byte" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
```

```
    </xs:complexType>
  </xs:element>
  <xs:element name="SecType">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="type" use="optional"/>
          <xs:attribute type="xs:byte" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="PhysicalSecurity" type="xs:string"/>
  <xs:element name="CyberSecurity">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Overall"/>
        <xs:element ref="SecType" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Value" type="xs:float"/>
  <xs:element name="Item">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="Value"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Safety">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Overall" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="SafetyType" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Security">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="PhysicalSecurity"/>
        <xs:element ref="CyberSecurity"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="additionalInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SafeSecure">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Safety"/>
        <xs:element ref="Security"/>
        <xs:element ref="additionalInfo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример информации для класса SafeSecure на основе XML-схемы для класса SafeSecure:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SafeSecure>
  <Safety>
    <Overall mode="PL"> d </Overall>  <!-- choose one PL from a, b, c, d, e, and none. -->
    <Overall mode="SIL"> 3 </Overall> <!-- choose one SIL from 0, 1, 2, and 3. -->
    <!-- individual safety measure. see Table 4.5 -->
    <SafetyType type = "ESTOP" mode="PL"> d </SafetyType>
    <SafetyType type = "ESTOP" mode="SIL" value="3" />
    <SafetyType type = "SRSC" mode="SIL" value = "2" />
  </Safety>
  <Security>
    <!-- None,LatchSensor,LockwithKey,LockwithActuator, see sub-clause 5.6 in ISO 22166-1 -->
    <PhysicalSecurity>  None    </PhysicalSecurity>
    <CyberSecurity>   <!-- list security functions provided by module -->
      <Overall>  2  </Overall>  <!-- overall security level of a module -->
      <SecType type = "HU_IA">  3  </SecType > <!-- individual security fct. see Table 4.10 -->
      <SecType type = "ACNT_MGT" value = "2"  />  <!--  see Table 4.10 -->
    </CyberSecurity>
  </Security>
  <additionalInfo>
    <Item>
      <Name> Minimum overall PL </Name>
      <Value> 2 </Value>
    </Item>
    <Item>
      <Name> Minimum overall security </Name>
      <Value> 1 </Value>
    </Item>
  </additionalInfo>
</SafeSecure>
```

С.8 Информация о классе Modelling

Класс Modelling обеспечивает информацию, которую модуль предоставляет для моделирования, например, тип симулятора, 3D модель и формат описания 3Dмодели, такой как файлы универсального формата описания роботов (URDF). XML-элементы, содержащие информацию о классе Modelling, должны соответствовать таблице С.9. Для представления информации о классе Modelling на основе таблицы С.9 следует использовать приведенную ниже XML-схему, после которой приведен пример информации для класса Modelling.

Таблица С.9 — XML-элементы для класса Modelling

Имя элемента	Описание
Modelling	В данном элементе определена вся информация, относящаяся к моделированию. Данный элемент содержит информацию об имитационном моделировании модуля. У модуля может быть одна или несколько имитационных моделей
SimulationModel	Данный элемент определяет имитационную модель, представляющую модуль. Имитационная модель состоит из симулятора (или моделирующей программы), формата описания модели и формата 3D модели
Simulator	Данный элемент определяет симулятор (или моделирующую программу) (например, Gazebo, Webots, RoboDK, SimSpark, OpenRave)
ModelFile	Данный элемент определяет форматы описания модели (MDF) и форматы 3D модели (3DF). Формат описания представляет модель модуля, внешнюю среду, средства визуализации и управления. Атрибут «type» имеет одно из следующих значений: Если описание относится к одному из форматов MDF, например URDF или SDF, то значением данного атрибута являются унифицированные указатели ресурса (URL) или пути к MDF-файлу, представляющему модуль. Если описание относится к одному из форматов 3DF, например STL, OBJ, 3DS, DAE или FBX, то значением данного атрибута является унифицированный указатель ресурса (URL) или путь к 3DF-файлу, представляющему модуль

Окончание таблицы С.9

Имя элемента	Описание
DynamicSW	Данный элемент определяет всю информацию, относящуюся к исполняемой форме. Данный элемент содержит подробные сведения об исполняемой форме, которая должна быть выполнена для обеспечения работы модуля. У данного элемента есть три атрибута ExeFileURL, SheelCmd и Properties, представленные в таблице С.10
additionalInfo	Данный элемент содержит дополнительную информацию о переменных. Он может не иметь ни одного или иметь несколько значений, задаваемых с помощью атрибута «Item», каждое значение которого состоит из пары «name» (имя) и «value» (значение)

XML-схема для класса Modelling:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="Item">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="Name" minOccurs="0"/>
        <xs:element ref="Value" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Property">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="name" use="optional"/>
          <xs:attribute type="xs:string" name="type" use="optional"/>
          <xs:attribute type="xs:string" name="unit" use="optional"/>
          <xs:attribute type="xs:string" name="description" use="optional"/>
          <xs:attribute type="xs:byte" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExeFileURL" type="xs:string"/>
  <xs:element name="ShellCmd" type="xs:string"/>
  <xs:element name="Properties">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Property" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Value" type="xs:string"/>
  <xs:element name="Simulator" type="xs:string"/>
  <xs:element name="ModelFile">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="type" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="DynamicSW">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ExeFileURL"/>
        <xs:element ref="ShellCmd"/>
```

```
        <xs:element ref="Properties"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="additionalInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SimulationModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Simulator"/>
        <xs:element ref="ModelFile" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="DynamicSW"/>
        <xs:element ref="additionalInfo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Modelling">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="SimulationModel"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример информации для класса Modelling на основе XML-схемы для класса Modelling:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Modelling>
  <SimulationModel> <!-- the 1-st simulation model to be used -->
    <Simulator> Gazebo </Simulator> <!-- Simulation programs. optional -->
    <ModelFile type="URDF"> <!-- model description file type and its file path for model
of module -->
     <Item> ../../modelDescrtionFileName1 </Item> <!-- the 1st path of model description file-->
<Item> ../../modelDescrtionFilePath2 </Item> <!-- the 2nd path of model description file -->
    </ModelFile>
    <ModelFile type="STL"> <!-- 3D graphic model type and its file path for model of
module -->
      <Item> ../../module3DGraphicFileName1 </Item> <!--the 1st path of 3Dgraphic file
model file -->
      <Item> ../../module3DGraphicFilePath2 </Item> <!--the 2nd path of 3Dgraphic file
model file -->
    </ModelFile>
    <DynamicSW>
      <ExeFileURL> ../../sim.exe </ExeFileURL>
      <ShellCmd> ../../sim.exe -r </ShellCmd> <!--example of shell command -->
      <Properties>
        <Property name="Shell cmd Arg1" type="int8" unit="none" description = "verbose"
value ="0" />
        <Property name="Shell cmd Arg2" type="uint8" unit="none" description = "Warning
level"  value ="2" />
      </Properties>
    </DynamicSW>
    <additionalInfo>
      <Item>
        <Name> kinematics model(Axis) </Name>
        <Value> 6 Axis </Value>
      </Item>
```

```
      <Item>
        <Name> additional info </Name>
        <Value> added </Value>
      </Item>
    </additionalInfo>
  </SimulationModel>
</Modelling>
```

С.9 Информация о классе ExecutableForm

Класс ExecutableForm представляет исполняемую форму, связанную с информацией, необходимой модулю при его выполнении. XML-элементы, содержащие информацию о классе ExecutableForm, должны соответствовать таблице С.10. Для представления информации о классе ExecutableForm на основе таблицы С.10 следует использовать приведенную ниже XML-схему, после которой приведен пример информации для класса ExecutableForm.

Таблица С.10 — XML-элементы для класса ExecutableForm

Имя элемента	Описание
ExeForm	В данном элементе определена вся информация, относящаяся к исполняемой форме. Данный элемент содержит подробные сведения об исполняемой форме, которая должна быть выполнена для обеспечения работы модуля. У данного элемента есть три атрибута ExeFileURL, SheelCmd и Properties
ExeFileURL	Данный элемент представляет унифицированные указатели ресурса (URL) или пути к файлам, необходимым модулю в начале его выполнения
ShellCmd	Данный атрибут используется для выполнения с помощью ExeFileURL, Properties и LibURLs разных задач, таких как прогон программ, исполнение системных команд, работа с файлами и папками, управление системными ресурсами
Properties	Данный элемент представляет дополнительные характеристики, требующиеся или необходимые для выполнения модуля
LibURLs	Данный элемент представляет унифицированные указатели ресурса (URL) или пути к библиотекам, необходимым модулю в начале его выполнения
additionalInfo	Данный элемент содержит дополнительную информацию о переменных. Он может не иметь ни одного или иметь несколько значений, задаваемых с помощью атрибута «Item», каждое значение которого состоит из пары «name» (имя) и «value» (значение)

XML-схема для класса ExecutableForm:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">
  <xs:element name="Item">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="Name" minOccurs="0"/>
        <xs:element ref="Value" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Property">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="name" use="optional"/>
          <xs:attribute type="xs:string" name="type" use="optional"/>
          <xs:attribute type="xs:string" name="unit" use="optional"/>
          <xs:attribute type="xs:string" name="description" use="optional"/>
          <xs:attribute type="xs:byte" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="ExeFileURL" type="xs:string"/>
  <xs:element name="ShellCmd" type="xs:string"/>
  <xs:element name="Properties">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Property" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Value" type="xs:string"/>
  <xs:element name="LibURLs">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExeForm">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ExeFileURL"/>
        <xs:element ref="ShellCmd"/>
        <xs:element ref="Properties"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="additionalInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExecutableForm">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="LibURLs"/>
        <xs:element ref="ExeForm"/>
        <xs:element ref="additionalInfo"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример информации для класса ExecutableForm на основе XML-схемы для класса ExecutableForm:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ExecutableForm>
  <LibURLs>
    <Item> ../lib/windows/modulelib.dll </Item>
    <Item> www.module/org/windows/lib/modulelib.dll </Item>
  </LibURLs>
  <ExeForm>
    <ExeFileURL> www.module.org/windows/exe/module1.exe</ExeFileURL>
    <ShellCmd> ./module1.exe -r </ShellCmd> <!--this is example of shell command -->
    <Properties>
      <Property name="Shell cmd Arg1" type="integer" unit="none" description = "verbose"
value ="0" />
      <Property name="Shell cmd Arg2" type="uint8" unit="none" description = "Warning
level"  value ="2" />
    </Properties>
  </ExeForm>
```

```
  <additionalInfo>
    <Item>
      <Name> update date </Name>
      <Value> Jun. 24, 2023 </Value>
    </Item>
    <Item>
      <Name> additional info </Name>
      <Value> added </Value>
    </Item>
  </additionalInfo>
</ExecutableForm>
```

Приложение D
(справочное)

Как использовать информационные модели

Информационные модели модулей обеспечивают разные виды информации, которая может помочь проектировщикам, разработчикам или интеграторам при создании нового модуля или составного модуля. Часть этой информации может быть использована на ранней стадии создания, часть — на стадиях проектирования и разработки, а часть — на стадии эксплуатации.

Примечание — ОИМ является абстрактной моделью, не предназначенной для непосредственной реализации. Пользователям необходимо использовать конкретные информационные модели программных модулей, аппаратных модулей и модулей с аппаратными и программными свойствами.

На рисунке D.1 представлены взаимосвязи между информацией о модуле и заинтересованными в ней лицами, когда информация о модуле представлена на основе общей информационной модели.

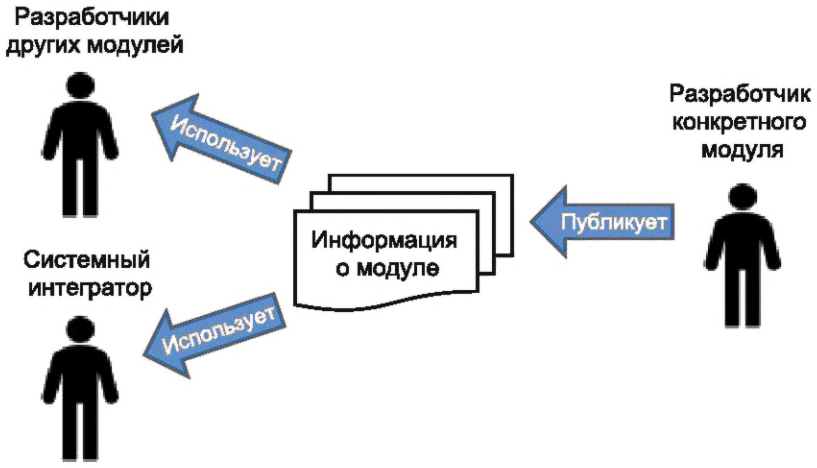

Рисунок D.1 — Взаимосвязи между информацией о модуле и заинтересованными в ней лицами

Информационные модели структурированы в виде нескольких субмоделей, как описано в 4.2. Пользователи в первую очередь могут получить сведения об идентификаторах и общую информацию, которая включает описание, сведения об изготовителе, примеры, идентификатор, типы модулей и аппаратно-программные свойства модуля. Если начальная информация о модуле соответствует требованиям пользователей, то они могут ознакомиться с более подробной информацией о модуле с помощью информации, хранящейся во внутренних классах. Если необходимо, они могут разработать дополнительную информацию о модуле, включая его модификацию и расширение его функций или сервисов.

В частности, интеграторы модулей, как пользователи, имеют возможность разрабатывать информацию для новых программных и аппаратных модулей, которые необходимы для построения нового сочетания модулей.

Информация, содержащаяся в общей информационной модели, может быть использована на разных стадиях жизненного цикла модуля. В таблицах D.1—D.3 представлена информация, предназначенная для использования на каждой из четырех стадий жизненного цикла: проектирование, разработка, эксплуатация и техническое обслуживание.

Таблица D.1 — Использование информации об аппаратных модулях (включая составные модули)

Имя тега или внутреннего класса	Проектирование	Разработка	Эксплуатация	Техническое обслуживание
GenInfo	M	M	X	M
IDnType	M	M	X	M
Properties	M	M	X	M
IOVariables	X	X	X	X
Status	X	X	X	X

Окончание таблицы D.1

Имя тега или внутреннего класса	Проектирование	Разработка	Эксплуатация	Техническое обслуживание
Services [a]	O	O	X	O
Infra	M	M	M	M
SafeSecure	M	M	M	M
Modelling	O	O	X	O
ExecutableForm	O	O	X	O
M: обязательно, O: факультативно, X: не используется. [a] Сервисы используют при объединении с другими модулями, примеры представлены в таблицах 4.2 и 4.3.				

Таблица D.2 — Использование информации о модулях с аппаратными и программными свойствами (включая составные модули)

Имя тега или внутреннего класса	Проектирование	Разработка	Эксплуатация	Техническое обслуживание
GenInfo	M	M	O	M
IDnType	M	M	O	M
Properties	M	M	O	M
IOVariables	O	O	O	O
Status	O	O	O	O
Services [a]	M	M	M	M
Infra	M	M	O	M
SafeSecure	M	M	O	M
Modelling	O	O	X	O
ExecutableForm	M	M	M	M
M: обязательно, O: факультативно, X: не используется. [a] Сервисы используют при объединении с другими модулями, примеры представлены в таблице 4.1.				

Таблица D.3 — Использование информации о программных модулях и составных модулях, включая программные модули и/или модули с аппаратными и программными свойствами

Имя тега или внутреннего класса	Проектирование	Разработка	Эксплуатация	Техническое обслуживание
GenInfo	M	M	O	M
IDnType	M	M	O	M
Properties	M	M	M	M
IOVariables	M	M	M	M
Status	M	M	M	M
Services	M	M	M	M
Infra	M	M	M	M
SafeSecure	M	M	M	M
Modelling	O	O	X	O
ExecutableForm	M	M	M	M
M: обязательно, O: факультативно, X: не используется.				

Информацию, предназначенную для стадий проектирования, разработки и технического обслуживания, используют главным образом пользователи при выполнении их задач.

Информацию, предназначенную для стадии эксплуатации, используют главным образом автоматизированные процессы, включая информацию о других модулях в системе, состоящей из нескольких модулей. Информация, отмеченная как «обязательная» в таблицах D.1 — D.3, должна быть предоставлена разработчиком модуля. Автоматические средства, работающие с подобными моделями, должны отказаться от выполнения или отклонить модель как недопустимую, если в ней отсутствует информация, помеченная как «обязательная». Однако никаких дополнительных требований не предъявляется к пользователю модуля в отношении использования какой-либо информации из модели. Информация, помеченная как «не используется», не должна присутствовать в модели любого модуля соответствующего типа, однако в явном виде не определено, как автоматические средства должны действовать в данной ситуации.

В таблицах D.1 — D.3 представлены информационные модели трех типов модулей: аппаратных модулей, модулей с аппаратными и программными свойствами и программных модулей соответственно. Таблица D.3 также соответствует составным модулям, состоящим из программных модулей и модулей с аппаратными и программными свойствами, причем последние могут включать и аппаратные модули.

Приложение Е
(справочное)

Формат представления класса

Классы определены в настоящем стандарте с использованием табличного формата, представленного в таблице Е.1.

Таблица Е.1 — Табличный формат для представления класса

Описание: <Описание класса>				
Выведен из: <Имя порождающего класса> или <не применимо>				
Атрибуты				
<Имя атрибута>	<Тип атрибута>	Обязательный или факультативный	Число объектов	Описание
…	…	…	…	…
Методы				
<Имя метода с типами и именами параметров>	<Тип возвращаемого значения>	Обязательный или факультативный	Описание	
…	…	…	…	

Атрибутами перечислимого типа данных являются перечислимые значения или константы.

Пример представления класса MyClass приведен в таблице Е.2.

Таблица Е.2 — Пример представления класса MyClass

Описание: Это пример.				
Выведен из: ОИМ				
Атрибуты				
attribute_1	String	M	1	Данный атрибут использован для примера
attribute_2	UnlimitedNatural	M	1	Данный атрибут использован для примера
attribute_3	Real	O	N	Данный атрибут имеет тип списка из N объектов
Методы				
method_1( )	void	M	Описание метода method_1	
method_2 (paraName_1: Integer, paraName_2: String)	Integer	M	Описание метода method_2 с параметрами	
Типы параметров методов отделены двоеточием «:».				

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным и межгосударственным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ISO 22166-1:2021	MOD	ГОСТ Р 60.2.0.1—2022 (ИСО 22166-1:2021 «Роботы и робототехнические устройства. Модульный принцип построения сервисных роботов. Часть 1. Общие требования»
IETF RFC 4122	—	*
IEEE/Open Group 1003.1-2017	—	*

* Соответствующий национальный стандарт отсутствует. До его принятия рекомендуется использовать перевод на русский язык данного международного стандарта.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандарта:

- MOD — модифицированный стандарт.

Библиография

[1] ISO 11179-5:2015 Information technology — Metadata registries (MDR) — Part 5: Naming principles

[2] ISO 13482:2014 Robots and robotic devices — Safety requirements for personal care robots

[3] ISO 13849-1:2023 Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design

[4] ISO/IEC 19505-1:2012 Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure

[5] ISO/IEC 19516 Information technology — Object Management Group — Interface definition language (IDL)

[6] IEC 60529 Degrees of protection provided by enclosures (IP Code)

[7] IEC 62061 Safety of machinery — Functional safety of safety-related control systems

[8] IEC 62443-4-2:2019 Security for industrial automation and control systems — Part 4-2: Technical security requirements for IACS components

УДК 006.034:004.94:007.52:621.865.8:681.5 ОКС 25.040.30

Ключевые слова: роботы, робототехнические устройства, модульный принцип, сервисные роботы, модули, информационные модели

Технический редактор *В.Н. Прусакова*
Корректор *Л.С. Лысенко*
Компьютерная верстка *Е.О. Асташина*

Сдано в набор 13.11.2024. Подписано в печать 03.12.2024. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 7,44. Уч.-изд. л. 6,70.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении в ФГБУ «Институт стандартизации»
для комплектования Федерального информационного фонда стандартов,
117418 Москва, Нахимовский пр-т, д. 31, к. 2.
www.gostinfo.ru info@gostinfo.ru