ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПНСТ 173—2016/ PAS 19450: 2015

СИСТЕМЫ ПРОМЫШЛЕННОЙ АВТОМАТИЗАЦИИ И ИНТЕГРАЦИЯ

Объектно-процессуальная методология

(PAS 19450:2015, IDT)

Издание официальное

Москва
Стандартинформ
2017

Предисловие

1 ПОДГОТОВЛЕН ООО «НИИ экономики связи и информатики «Интерэкомс» (ООО «НИИ «Интерэкомс») на основе собственного перевода на русский язык англоязычной версии стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 100 «Стратегический и инновационный менеджмент»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 5 декабря 2016 г. № 95-пнст

4 Настоящий стандарт идентичен международному документу PAS 19450:2015 «Системы промышленной автоматизации и интеграция. Объектно-процессуальная методология» (PAS 19450:2015 «Automation systems and integration — Object-Process Methodology», IDT)

5 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта и проведения его мониторинга установлены в ГОСТ Р 1.16—2011 (разделы 5 и 6).

Федеральное агентство по техническому регулированию и метрологии собирает сведения о практическом применении настоящего стандарта. Данные сведения, а также замечания и предложения по содержанию стандарта можно направить не позднее чем за девять месяцев до истечения срока его действия, разработчику настоящего стандарта по адресу: 123423, г. Москва, ул. Народного Ополчения, д. 32 и в Федеральное агентство по техническому регулированию и метрологии по адресу: 109074, г. Москва, Китайгородский проезд, д. 7, стр. 1.

В случае отмены настоящего стандарта соответствующее уведомление будет опубликовано в ежемесячно издаваемом информационном указателе «Национальные стандарты» и журнале «Вестник Федерального агентства по техническому регулированию и метрологии». Уведомление будет размещено также на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.gost.ru)

Содержание

Введение

Объектно-процессуальная методология (далее — OPM-методология) представляет собой компактный концептуальный подход, язык и методологию моделирования и представления знаний в системах автоматизации, применение которой варьируется от простых сборок из элементарных компонентов до сложных, многопрофильных динамических систем. OPM-методология пригодна для реализации и поддержки средствами, использующими информационные и компьютерные технологии. В настоящем стандарте определены язык и методология для различных аспектов OPM-методологии, предназначенных для создания общей основы для разработчиков архитектуры систем, проектировщиков и разработчиков OPM-совместимых средств моделирования всех видов систем.

OPM-методология предоставляет собой два семантически эквивалентных способа представления одной и той же модели: графический и текстовый. Набор иерархически связанных между собой объектно-процессуальных диаграмм (далее — OPD-диаграмм) определяет графическую модель, а набор автоматически сформированных последовательностей в подмножестве английского языка определяет текстовую модель, выраженную на объектно-процессуальном языке (далее — на OPL-языке). В графической визуальной модели каждая из OPD-диаграмм состоит из OPM-элементов в виде графических символов (иногда с аннотационной меткой). OPD-синтаксис полностью и последовательно определяет способы расположения графических элементов. Использование OPL-языка и OPM-методологии позволяет создавать такую соответствующую текстовую модель для каждой OPD-диаграммы, которая будет сохранять все ограничения графической модели. Поскольку синтаксис и семантика OPL-языка являются подмножеством английского естественного языка, эксперты в определенной области знаний смогут легко понимать текстовую модель.

OPM-обозначения (нотация) позволяют поддерживать концептуальное моделирование систем с формальным синтаксисом и семантикой, которые в целом служат основой для моделирования инженерных систем, в том числе системной архитектуры, процессов проектирования, разработки, поддержки жизненного цикла, информационного обмена и дальнейшего совершенствования систем. Кроме того, не связанная с конкретной областью применения OPM-методология обеспечивает возможность моделирования систем для всего научного, коммерческого и промышленного сообщества в целях разработки, изучения и анализа производственных, промышленных и коммерческих систем в собственных областях применения, тем самым позволяя компаниям объединять и обеспечивать интероперабельность различных навыков и компетенций в единую интуитивную формальную среду.

OPM-методология облегчает понимание общего представления системы на стадии ее создания, проведения испытаний, интеграции и ежедневного технического обслуживания, обеспечивая при этом работу в многопрофильной среде. Кроме того, с помощью OPM-методологии компании могут расширить их комплексное представление о функциональности системы, ее гибкости при закреплении персонала за определенными работами, управлении в непредвиденных ситуациях и восстановлении работоспособности системы после ее отказа. Системная спецификация является расширяемой для любой необходимой детали, охватывающей функциональные, структурные и поведенческие аспекты системы.

Одним из конкретных применений OPM-методологии являются составление и разработка технических стандартов. OPM-методология позволяет в первом приближении оценить применимость стандарта и выявлять его слабые стороны, тем самым значительно улучшая качество последующих проектов. При использовании OPM-методологии даже в качестве модели, основанной на тексте, система может расширяться за счет включения более подробной информации, причем базовая модель будет сохранять высокую степень формализованности и целостности.

Настоящий стандарт создает основу для работы разработчиков архитектуры систем и проектировщиков, которые могут использовать ее для наглядного и эффективного создания моделей информационных систем. Поставщики инструментальных средств, поддерживающих OPM-методологию, могут использовать настоящий стандарт в качестве официальной стандартной спецификации для создания программных средств с целью совершенствования своих возможностей концептуального моделирования.

Настоящий стандарт устанавливает требования к представлению нормативного текста, который должен соответствовать расширенной спецификации в форме Бэкуса — Наура (EBNF) для синтаксиса языка. В разделах 6—13 все элементы представлены лишь с минимальным упоминанием методологических аспектов; в разделе 14 представлены способы управления контекстом, связанные с масштабированием (in-zooming) и развертыванием (unfolding).

В настоящем стандарте используется ряд соглашений (общепризнанных подходов) относительно представления OPM-методологии. В частности, жирный шрифт типа Arial в тексте и жирный курсив шрифта типа Arial в подписях к рисункам, в таблицах и заголовках предназначен для наименований меток OPM-объектов, процессов, состояний и тэгов связей. Ключевые слова OPL-языка выделяются обычным шрифтом типа Arial с запятыми и периодически — жирным шрифтом типа Arial. Большинство диаграмм содержат как графические изображения OPD-диаграмм, так и текстовые эквиваленты OPL-языка. Поскольку они относятся к спецификациям языка, существенно использовать точные определения терминов; некоторые общеупотребительные термины могут иметь особый смысл при использовании OPM-методологии. В разделе B.6 поясняются другие соглашения, принимаемые при использовании OPM-методологии.

В приложении A представлен формальный синтаксис OPL-языка, представленный в расширенной форме Бэкуса — Наура (EBNF).

В приложении B содержатся соглашения и шаблоны, обычно используемые в приложениях OPM-методологии.

В приложении C рассмотрены различные аспекты OPM-методологии в качестве ее моделей.

В приложении D обобщены динамические и (имитационные) возможности моделирования OPM-методологии.

ПНСТ 173—2016/
PAS 19450:2015

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

СИСТЕМЫ ПРОМЫШЛЕННОЙ АВТОМАТИЗАЦИИ И ИНТЕГРАЦИЯ

Объектно-процессуальная методология

Automation systems and integration. Object-process methodology

Срок действия предстандарта — с 2017—06—01
по 2019—06—01

1 Область применения

В настоящем стандарте определена объектно-процессуальная методология (OPM-методология) с достаточно подробными для специалистов-практиков понятиями, семантикой и синтаксисом этой методологии как парадигмы моделирования и языка для создания концептуальных моделей с различной степенью детализации, а также для предоставления поставщикам средств моделирования приложений для создания моделей, понятных вышеупомянутым специалистам-практикам.

Хотя в настоящем стандарте приведен ряд поясняющих примеров использования OPM-методологии, в ней не предпринимались попытки полного описания всевозможных применений данной методологии.

2 Нормативные ссылки

В настоящем стандарте нормативные ссылки отсутствуют.

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1 **абстракция** (abstraction): Уменьшение степени детализации и модельной *полноты* системы (3.8) для достижения лучшего понимания.

3.2 **объекты, подвергаемые влиянию (изменениям) со стороны процесса** (affectee): Объект типа *transformee* (3.78), который зависит от возникновения *процесса* (3.58), т. е. от изменения его *состояния* (3.69).

Примечание — Объектом типа *affectee* может быть только *объект с внутренним состоянием* (3.66). *Объект без внутреннего состояния* (3.67) может только создаваться или использоваться, но не подвергаться изменениям его состояния.

3.3 **агент** (agent): *Реализатор* (3.17), которым могут быть человек или группа людей.

3.4 **атрибут** (attribute): *Объект* (3,39), который характеризует любую *сущность* (3.76), кроме самого себя.

3.5 **поведение** (behaviour): *Преобразование* (3.77) *объектов* (3.39) в результате применения модели *OPM-методологии* (см. 3.43), содержащей совокупность *сущностей* (3.76) и *связей* (3.36) объектов в модели.

Издание официальное

3.6 **бенефициар** (beneficiary): <система> *Заинтересованная сторона* (3.65), которая получает функциональную *выгоду* (3.82) от *работы* системы (3.46).

3.7 **класс** (class): Совокупность *сущностей* (3.76) с одними и теми же значениями *устойчивости* (3.50), отличительными свойствами и установленными ценностями, а также с одним и тем же набором *признаков (особенностей)* (3.21) и *состояний* (3.69).

3.8 **полнота** (completeness): <модель системы> Степень, в которой все детали системы раскрываются в модели.

3.9 **условная связь** (condition link): *Процедурная связь* (3.56) от *объекта* (3.39) или *состояния* объекта (3.69) к *процессу* (3.58), обозначающая процедурное ограничение.

3.10 **объекты, потребляемые процессом** (consumee): Объект типа *transformee* (3.78), который при возникновении *процесса* (3.58) потребляется или исключается.

3.11 **контекст** (context): <модель> Часть модели *OPM-методологии* (3.43), представленная в виде *OPD-диаграммы* (3.41) процесса и соответствующего текста на *OPL-языке* (3.42).

3.12 **управляющая связь** (control link): *Процедурная связь* (3.56) с дополнительной семантикой управления.

3.13 **модификатор управления** (control modifier): Символ, дополняющий *связь* (3.36) для введения в нее управляющей семантики и выполнения *управляющей связи* (3.12).

Примечание — Модификаторы управления — это буквы 'e' для *события* (3.18), или буквы 'c' — для условия.

3.14 **дискриминирующий атрибут** (discriminating attribute): *Атрибут* (3.4), чьи различные *значения* (3.81) определяют соответствующие специализированные взаимоотношения.

3.15 **эффект, воздействие** (effect): Изменение *состояния* (3.69) *объекта* (3.39) или *значения* (3.81) *атрибута* (3.4).

Примечание — Термин «*эффект*» применяется только к *объектам, имеющим внутреннее состояние* (3.66).

3.16 **элемент** (element): *Сущность* (3.76) или *связь* (3.36).

3.17 **реализатор, средство реализации** (enabler): <процесс> *Объект* (3.39), который создает возможности для *процесса* (3.58), но не способен *преобразовывать* его.

3.18 **событие** (event): <OPM> Момент создания (или появления) *объекта* (3.39), или момент перехода объекта в определенное *состояние* (3.69), или момент инициализации оценки *процесса* (3.58) при установке *предварительных условий* (3.53).

3.19 **событийная связь** (event link): *Управляющая связь* (3.12), обозначающая *событие* (3.18), происходящее с *объектом* (3.39) или с *состоянием* (3.69) *процесса* (3.58).

3.20 **экспонент** (exhibitor): *Сущность* (3.76), которая представляет *признак* [или характеризуется *особенностью* (3.21)] с помощью связи типа «представление-характеризация».

3.21 **признак, особенность** (feature): *Атрибут* (3.4) или *операция* (3.46).

3.22 **сворачивание** (folding): Способ *абстракции (обобщения)* (3.1), реализуемый путем скрытия сущностей типа *refineables* (3.61) развернутого объекта типа *refinee* (3.62).

Примечание 1 — Четырьмя видами свернутых сущностей типа *refineables* являются части (свернутые части), *признаки* (3.21) (свернутые признаки), специализации (свернутые специализации) и *экземпляры* (3.28) (свернутые экземпляры).

Примечание 2 — Свертывание в основном применяют к *объектам* (3.39). Что касается процесса, то его подпроцессы не упорядочены, что является достаточным для моделирования асинхронных систем, в которых временной порядок процессов не определен.

Примечание 3 — Противоположным термину «сворачивание» является термин «разворачивание» (3.80).

3.23 **функция** (function): *Процесс* (3.58), который приносит *бенефициару* (3.6) функциональную *выгоду* (3.82).

3.24 **объект типа general** (general): <OPM> Сущность типа *refineable* (3.61) со своими специализациями.

3.25 **информационный** (informatical): Термин, относящийся, например, к данным, информации, знаниям (или к информатике).

3.26 **наследование** (inheritance): Закрепление *элементов* (3.16) *объекта типа general* (3.24) в *OPM-методологии* (3.43) за их специализациями.

3.27 **входная связь** (input link): *Связь* (3.36) между *состоянием* источника (3.69) (поступающего на вход) *объекта* (3.39) и преобразующим *процессом* (3.58).

3.28 **экземпляр** (instance): <модель> Экземпляр *объекта* (3.39) или *процесса* (3.58), который является объектом типа *refinee* (3.62) в связи типа «классификация-инстанцирование».

3.29 **экземпляр** (instance): <*операционное*> Экземпляр *объекта* (3.39) или *процесса* (3.58), который является фактической, однозначно идентифицируемой *сущностью* (3.76) и который существует во время функционирования модели (3.46), например, во время моделирования или при ее реализации.

Примечание — Экземпляр процесса можно идентифицировать по действующим экземплярам *набора существующих объектов* (3.32) во время возникновения и начала процесса и появления конечных временных меток.

3.30 **инструментальное средство** (instrument): *Средство реализации* (3.17), не являющееся человеком.

3.31 **инициализация** (invocation): <процесс> Активация одного *процесса* (3.58) с помощью другого процесса.

3.32 **набор существующих объектов** (involved object set): Совокупность из *набора предварительно обработанных объектов* (3.54) и *набора апостериорно обработанных объектов* (3.52).

3.33 **детализированный контекст** (in-zoom context): *Сущности* (3.76) и *связи* (3.36) в пределах сущности, находящейся в *детализированном* состоянии.

3.34 **детализация** (in-zooming): <объект> *Разворачиваемая* (3.80) часть *объекта* (3.39), которая указывает на пространственное упорядочение составных объектов.

3.35 **детализация** (in-zooming): <процесс> *Разворачиваемая* (3.80) часть *процесса* (3.58), которая указывает на временное частичное упорядочение составных процессов.

3.36 **связь** (link): Графическое выражение *структурного соотношения* (3.73) или *процедурного соотношения* (3.57) между двумя *сущностями* (3.76) *OPM-методологии* (3.43).

3.37 **метамодель** (metamodel): Модель языка моделирования (или его часть).

3.38 **факт модели** (model fact): Установленное соотношение между двумя сущностями (3.76) в *OPM-методологии* (3.43) или двумя *состояниями* (3.69) в модели OPM-методологии.

3.39 **объект** (object): <OPM> *Элемент* (3.16) модели, характеризующий *сущность* (3.76), которая должна или может существовать физически или *информационно* (3.25).

3.40 **класс объектов** (object class): Совокупность *объектов* (3.39), имеющих одну и ту же *структуру* (3.74) и алгоритм *преобразования* (3.77).

3.41 **объектно-процессуальная диаграмма, OPD-диаграмма** (Object-Process Diagram; OPD): Графическое представление модели (или части модели) в *объектно-процессуальной методологии* (3.43), в которой *объекты* (3.39) и *процессы* (3.58) в рассматриваемой области представляются среди объектов и процессов вместе со *структурными связями* (3.72) и *процедурными связями* (3.56).

3.42 **объектно-процессуальный язык, OPL-язык** (Object-Process Language; OPL): Разновидность английского естественного языка, который текстуально характеризует модель *объектно-процессуальной методологии* (3.43), как *объектно-процессуальная диаграмма* (3.42) характеризует ее графически.

3.43 **объектно-процессуальная методология** (Object-Process Methodology; OPM): Формальный язык и метод определения сложных, многопрофильных систем с помощью единственной функциональной структурной поведенческой унифицированной модели, которая использует бимодальное графически-текстовое представление *объектов* (3.39) в системе и их *преобразование* (3.77) или их использование в *процессах* (3.58).

3.44 **древо OPD-объектов** (OPD object tree): Древовидный граф, чьим корнем является *объект* (3.39), детализируемый посредством *уточнения* (3.63).

3.45 **древо OPD-процессов** (OPD process tree): Древовидный граф, корнем которого является *системная диаграмма* (3.75), а каждый узел представляет собой *объектно-процессуальную диаграмму* (3.42), получаемую путем *детализации* (3.35) *процесса* (3.58), приведенного на его предшествующей OPD-диаграмме (или системной диаграмме), и в каждой из которых ребра графа направляются от уточненного процесса на родительской OPD-диаграмме к аналогичному процессу на дочерней OPD-диаграмме.

Примечание — Уточнение модели *объектно-процессуальной методологии* (3.43), как правило, происходит путем декомпозиции процесса посредством его детализации, поэтому древо OPD-процессов служит в качестве основного средства навигации по OPM-модели.

3.46 **операция** (operation): *Процесс* (3.58), который выполняет *сущность* (3.76), характеризующая любую другую сущность, кроме самой себя.

3.47 **выходная связь** (output link): *Связь* (3.36) между преобразующим *процессом* (3.58) и выходным (целевым) *состоянием* (3.69) *объекта* (3.39).

3.48 **обобщение** (out-zooming): <объект> Операция, обратная *детализации* (3.34) *объекта* (3.39).

3.49 **обобщение** (out-zooming): <процесс> Операция, обратная *детализации* (3.35) *процесса* (3.58).

3.50 **устойчивость** (perseverance): *Свойство* (3.60) *сущности* (3.76), которое может быть статическим, определяющим *объект* (3.39), или динамическим, определяющим *процесс* (3.58).

3.51 **постусловие** (postcondition): <процесс> Условие, которое является результатом успешного выполнения *процесса* (3.58).

3.52 **набор апостериорно обработанных объектов** (postprocess object set): Совокупность *объектов* (3.39), оставшихся или появившихся в результате завершения *процесса* (3.58).

Примечание — Набор апостериорно обработанных объектов может содержать *объекты, обладающие внутренними состояниями* (3.66), в которых специфические *состояния* (3.69) будут возникать в результате выполнения процесса.

3.53 **предварительное условие** (precondition): <процесс> условие, необходимое для запуска *процесса* (3.58).

3.54 **набор предварительно обработанных объектов** (preprocess object set): Совокупность *объектов* (3.39), предназначенных для их оценки до начала *процесса* (3.58).

Примечание — Набор предварительно обработанных объектов может включать в себя *структурированные объекты* (3.66), для которых конкретные *состояния* (3.69) необходимы в случае реализации какого-либо процесса.

3.55 **исходный смысл** (primary essence): <система> Смысл большинства *сущностей* (3.76) в системе, который может быть либо *информационным* (3.25), либо физическим.

3.56 **процедурная связь** (procedural link): Графическое обозначение *процедурного отношения* (3.57) в *OPM-методологии* (3.43).

3.57 **процедурное отношение** (procedural relation): Соединение или связь между *объектом* (3.39) или *состоянием* объекта (3.69) и *процессом* (3.58).

Примечание 1 — Процедурные отношения определяют, как система действует, выполняя свою *функцию* (3.23) и обозначая зависящую от времени или условную инициализацию процессов, которые преобразуют объекты.

Примечание 2 — *Инициализация* (3.31) или исключающая *связь* (3.36) означают временный объект в потоке сигналов управления между двумя процессами.

3.58 **процесс** (process): *Преобразование* (3.77) в системе одного или нескольких *объектов* (3.39).

3.59 **класс процесса** (process class): Шаблон для процессов (3.58), которые выполняют *преобразование* (3.77) одного и того же *объекта* (3.39) в соответствии с шаблоном.

3.60 **свойство** (property): Моделирующая аннотация, общая для всех *элементов* (3.16) определенного вида, которая служит для различения этого *элемента.*

Примечание 1 — Мощность ограничений, метки путей и теги *структурной связи* (3.72) часто являются аннотациями свойств.

Примечание 2 — В отличие от *атрибута* (3.4) значение свойства не может изменяться в процессе моделирования или оперативной реализации. Каждый вид элемента обладает собственным набором свойств.

Примечание 3 — Свойство является атрибутом элемента в *мета-модели* (3.37) *OPM-методологии* (3.43).

3.61 **сущность типа refineable** (имя существительное) (refineable): <OPM> *Сущность* (3.76), поддающаяся *детализации* (3.63), которая может быть сущностью типа *whole* (3.83), *экспонентом* (3.20), сущностью типа *general* (3.24) или *классом* (3.7).

3.62 **сущность типа refinee** (refinee): *Сущность* (объект) (3.76), детализирующая сущность типа *refineable* (3.61), которая может быть частью, *признаком* (3.21), *специализацией* или *экземпляром* (3.29).

Примечание — Каждый из четырех видов объектов типа *refinees* имеет соответствующую сущность типа *refineable* (типа «часть-целое», «признак-представитель», «детализация-обобщение», «экземпляр-класс»).

3.63 **уточнение** (refinement): <модель> Детализация, которая позволяет увеличивать степень подробности и соответствующую *полноту* (3.8) модели.

3.64 **объект типа resultee** (resultee): Объект типа *transformee* (3.78), который приводит к реализации *процесса* (3.58).

3.65 **заинтересованная сторона** (stakeholder): <OPM> Индивидуум, организация или группа людей, которая имеет собственные интересы или может затрагиваться системой, разрабатываемой, разработанной или развернутой.

3.66 **объект с внутренним состоянием** (stateful object): *Объект* (3.39), обладающий определенными *состояниями* (3.69).

3.67 **объект без внутреннего состояния** (stateless object): *Объект* (3.39), не обладающий определенными *состояниями* (3.69).

3.68 **состояние** (state): <объект> Возможная ситуация или положение *объекта* (3.39).

Примечание — В *объектно-процессуальной методологии* (3.43) не существует понятия *состояние процесса* (3.58): например, «начальное», «действующее» или «конечное» состояние (в рамках существующей модели). Вместо этого *объектно-процессуальная методология* определяет и моделирует такие подпроцессы, как запуск, обработка или окончание. См. также обсуждение метамодели для объектно-процессуальной методологии в приложении С.

3.69 **состояние** (state): <система> «Мгновенный снимок» модели системы, сделанный в определенный момент времени и показывающий все существующие экземпляры *объекта* (3.39), текущие состояния каждого экземпляра *объекта с внутренним состоянием* (3.66) и экземпляры *процесса* (3.58) вместе со значениями соответствующих истекших времен с момента выполнения этого «снимка».

3.70 **выражение состояния** (state expression): *Уточнение* (3.63), включающее в себя выявление соответствующего подмножества из множества *состояний* (3.69) *объектов* (3.39).

3.71 **подавление состояния** (state suppression): *Абстракция* (3.1), включающая в себя сокрытие соответствующего подмножества из множества *состояний* (3.69) *объектов* (3.39).

3.72 **структурная связь** (structural link): Графическое обозначение структурного соотношения (3.73) в *объектно-процессуальной методологии* (3.43).

3.73 **структурное соотношение** (structural relation): Функционально инвариантная связь или связь между сущностями.

Примечание — Структурные соотношения сохраняются в системе по крайней мере в течение определенного промежутка времени. Они обеспечивают структурный аспект системы, а также они не могут наступать при условиях, которые зависят от времени.

3.74 **структура** (structure): <OPM> Совокупность *объектов* (3.39) в модели *объектно-процессуальной методологии* (3.43) и невременных отношений (или связей между ними).

3.75 **системная диаграмма** (System Diagram; SD): *Объектно-процессуальная диаграмма* (3.41) с единственным системным *процессом* (3.58), указывающим на системную *функцию* (3.23) и *объекты* (3.39), соединенные с данной функцией для отображения общего *контекста* (3.11) для анализа системы верхнего уровня.

Примечание — Системная диаграмма является корнем *древа OPD-процесса* (3.45) и не имеет степени детализации вне общего контекста, т. е. отсутствует детализированный объект *refinee* (3.62). Любая объектно-процессуальная диаграмма, кроме системной диаграммы, является узлом на древе *OPD-процесса,* обусловленным *уточнением* (3.63).

3.76 **сущность** (thing): <OPM> *Объект* (3.39) или *процесс* (3.58).

3.77 **преобразование** (transformation): Создание (формирование, построение) или потребление (удаление, уничтожение) *объекта* (3.39) или изменение *состояния* (3.69) объекта.

Примечание — Преобразование может выполнять только *процесс* (3.58).

3.78 **объект типа transformee** (transformee): *Объект* (3.39), который позволяет преобразовывать (создавать, потреблять или воздействовать) *процесс* (3.58).

3.79 **преобразующая связь** (transforming link): Связь, указывающая на потребление, воздействие или взаимодействие.

3.80 **разворачивание** (unfolding): *Уточнение* (3.63), которое позволяет дополнительно детализировать сущность типа *refinee* (3.62), а также включать другие *сущности* (3.76) и *связи* (3.36) между ними.

Примечание 1 — Существуют четыре вида разворачивания: структурное (составных частей), функциональное (функциональных особенностей), по специализации и по экземплярам.

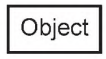
Примечание 2 — Разворачивание в основном применяют к *объектам* (3.39) для раскрытия информации о разворачиваемом объекте.

3.81 **значение** (value): <атрибут> *Состояние* (3.69) *атрибута* (3.4).

3.82 **значение** (value): <функциональный> Выигрыш, измеряемый в затратах, который данная *функция* (3.23) обеспечивает для системы.

3.83 **сущность типа whole** (whole): Комплексная *сущность* (3.76), состоящая из двух или более частей, каждая из которых имеет такую же устойчивость (3.50), как и совокупность частей вместе.

4 Условные обозначения



Объект (object)

Физический объект (physical object)

Объект внешней среды (environmental object)

Процесс (process)

Физический процесс (physical process)

Процесс внешней среды (environmental process)

Состояние (state)

Агрегация — является частью (aggregation-participation)

Представление-характеризация (exhibition-characterization)

Обобщение-специализация (generalization-specialization)

Классификация-инстанцирование (classification-instantiation)

Однонаправленная тегированная структурная связь (unidirectional tagged structural link)

Двунаправленная тегированная структурная связь (bidirectional tagged structural link)

Агентская связь (agent link)

Инструментальная связь (instrument link)

Взаимодействующая связь (effect link)

Потребительская связь (consumption link)

Результирующая связь (result link)


Пара входных-выходных связей (input-output link pair)


Инструментальная ситуационная связь (instrument event link)


Потребительская ситуационная связь (consumption event link)


Инструментальная условная связь (instrumental condition link)


Потребительская условная связь (consumption condition link)


Инициализирующая связь (invocation link)


Самоинициализирующаяся связь (self-invocation link)


Передержанная исключающая связь (over-time exception link)


Недодержанная исключающая связь (under-time exception link)

5 Соответствие настоящему стандарту

Применение настоящего стандарта создателями инструментария и конечными пользователями, вероятно, будет происходить поэтапно в течение достаточно продолжительного времени, наиболее подходящими представляются несколько видов критериев соответствия.

a) При частичном (символическом) соответствии OPM-методологии следует использовать ее лингвистическую часть, а именно OPM-семантику и синтаксис:

1) используя только OPM-символы, определенные в разделе 4, и со смыслом, указанным в настоящем стандарте;

2) используя только OPM-элементы, определенные в разделах 7—12, со смыслом и положениями, указанными в настоящем стандарте.

b) Полное соответствие OPM-методологии должно требовать:

1) соответствия с a);

2) соответствия подхода и схемы моделирующих систем с OPM-методологией, как это определено в разделах 6—14.

c) Соответствие с требованиями к разработчику инструментария требует:

1) соответствия с a);

2) обеспечения выполнения b) — пользователи должны руководствоваться и придерживаться b) на основе формальной трактовки a);

3) поддержки OPL-языка в соответствии с определениями расширенной формы Бэкуса — Наура (EBNF), указанными в приложении A.

6 Принципы и концепции OPM-методологии

6.1 Принципы OPM-моделирования

6.1.1 Моделирование как деятельность, способствующая достижению целей

Системная функция и цель моделирования должны определять объем и степень детализации OPM-модели. Комплексная составная система может быть связана с множеством заинтересованных сторон, включая бенефициаров, владельцев, пользователей и контрольно-надзорные органы, а также

множеством аппаратных и программных компонентов, отражающих различные аспекты и имеющих отношение к каждой из заинтересованных сторон. Основные функции и ожидания выгоды для заинтересованных сторон в целом и бенефициаров в частности должны определять цели моделирования, что, в свою очередь, будет определять область применения системной модели.

***Пример** — На производственном предприятии, производящем виджеты, менеджер по маркетингу, отвечающий за тарифы, поставку и даты, не принимает во внимание станки, которые используются на предприятии в качестве средств изготовления виджетов и не учитывает их при маркетинге. Однако с точки зрения менеджера по техническому обслуживанию эти станки, безусловно, подвергаются воздействиям, поскольку они изнашиваются в процессе эксплуатации, поэтому их необходимо технически поддерживать как для предотвращения разрушения, так и для определения момента возможной поломки. Таким образом, OPM-модель производственного предприятия для менеджера по маркетингу будет существенно отличаться от таковой, создаваемой для менеджера по техническому обслуживанию.*

6.1.2 Унификация функций, структуры и модели поведения

Структурной OPM-моделью системы должна быть совокупность физических и информационных (логических) объектов, связанных структурными взаимоотношениями. На протяжении всего срока службы системы может происходить создание и разрушение структурных взаимоотношений.

В OPM-модели поведения системы, называемой ее динамикой, должны отражаться механизмы, которые с течением времени могут воздействовать на системы для преобразования объектов системы, т. е. объектов, которые являются внутренними по отношению к системе, и/или объектов окружающей среды, т. е. объектов, которые в последнем случае являются внешними по отношению к системе.

Сочетание структуры системы и ее поведения позволяет системе выполнять функцию, которая должна придавать системе (функциональную) ценность (по крайней мере одной из заинтересованных сторон, которая является потребителем системы). OPM-модель интегрирует функциональные (утилитарные), структурные (статические) и поведенческие (динамические) аспекты системы в единую унифицированную модель. Поддерживая внимание с точки зрения общей системной функции, эта унификация «структура-поведение» дает единую согласованную систему критериев для понимания системы интересов и повышения ее интуитивного понимания (при соблюдении формального синтаксиса).

6.1.3 Определение функциональной ценности

Функциональная ценность, создаваемая процессом моделирования системы, должна отражать функцию системы, воспринимаемую основным бенефициаром (или группой бенефициаров). Идентификация и классификация этого первичного процесса и системной функции является важным первым шагом в построении OPM-модели в соответствии с рекомендуемой методологией OPM-подхода. Соответствующая функциональная метка или имя должны уточнять и подчеркивать основную цель моделирования системы и функциональную ценность, которую система должна давать основному бенефициару. Моделирование с использованием OPM-методологии следует начинать с определения, именования и отображения функции системы в качестве основного процесса.

Примечание — Подобный подход часто провоцирует споры между членами коллектива по разработке системной архитектуры на этой ранней стадии и является чрезвычайно полезным, поскольку позволяет выявлять различия и часто даже неверные представления у участников споров относительно системы, которые они адресуют специалисту по архитектуре системы, модели и дизайну.

После того как установлено, что функция системы совпадет с ожиданиями функциональной ценности главного бенефициара, разработчик модели должен идентифицировать и добавить интересы других основных заинтересованных сторон в OPM-модель.

6.1.4 Связь между функциональностью и поведением модели

Ценность функции для бенефициара часто выражается в терминах процессов, которые определяют то, что происходит, поведение, а не конечную цель, функциональную ценность, которая должна являться основной для первичного процесса. Разработчику модели следует различать функции и модели поведения для создания четкой и однозначной модели системы. Это имеет большое значение, поскольку во многих ситуациях системная функция может реализовываться с использованием различных концепций, каждая из которых, в свою очередь, может по-разному реализовывать различные конструкции и модели поведения.

***Пример** — Рассмотрим систему, обеспечивающую людям переправу через реку на их транспортных средствах. Две очевидные концепции — это статическая конструкция, позволяющая автомобилям форсировать реку, и динамический передвижной элемент, перевозящий автомобили.*

Соответствующие конструкции системы — это мосты и паромы. Хотя функция и первичный процесс — River Crossing (Переправа через реку) — одинаковы для обеих конструкций, тем не менее они резко различаются по своей структуре и модели поведения.

Неспособность распознавать различия между функцией и моделью поведения может приводить к преждевременному выбору неоптимальной конструкции. В приведенном выше примере это может приводить к принятию решения о строительстве моста без учета, возможно, более эффективного варианта паромной переправы.

6.1.5 Установка границ системы

Системная среда должна представлять собой совокупность объектов, которые могут находиться вне системы, но взаимодействовать с ней, возможно, изменяя саму систему и ее окружение. Разработчик модели должен различать эти объекты внешней среды, которые не являются частью системы, начиная от системных объектов, которые являются частью этой системы. Разработчик не в состоянии разрабатывать архитектуру системы, ее конструкцию или же манипулировать структурой и моделью поведения внешней среды, даже если последние могут влиять на систему (или находиться под ее влиянием).

6.1.6 Ясность и полнота компромиссных решений

Реальным системам присущи избыточная детальность и усложненность, поэтому очевидно, что создание этих систем может приводить к компромиссам, которые должны балансировать между двумя противоположными критериями: ясности и полноты. Ясностью системы должна быть степень однозначного понимания структуры и модели поведения модели системы передачи, а полнотой системы — степень определения всех деталей системы. Эти два атрибута модели конфликтуют друг с другом. С одной стороны, полнота требует полной обусловленности деталей системы, но с другой стороны, потребность в ясности устанавливает верхний предел для степени детализации в отдельной диаграмме модели, после чего понимание системы может снизиться из-за затруднений и перегрузок.

Установление надлежащего баланса в процессе разработки модели требует тщательного управления контекстом. Усиление выражения полноты в данной диаграмме модели часто приводит к снижению ее ясности. Тем не менее разработчик модели может извлечь выгоду от объединения информации, представляемой OPM-моделью всей системы, имея только одну четкую и однозначную диаграмму, которая, однако, может быть не полной, и другой моделью, которая будет более подробно концентрироваться на полноте определенной части системы.

6.2 Основные понятия OPM-методологии

6.2.1 Бимодальное представление

OPM-модель должна быть бимодальной (двухуровневой), выражая семантически эквивалентные графические и текстовые представления. Каждая графическая диаграмма OPM-модели (т. е. OPD-диаграмма) должна иметь эквивалентный OPM-текстовый раздел, состоящий из одного или нескольких OPM-предложений на OPL-языке.

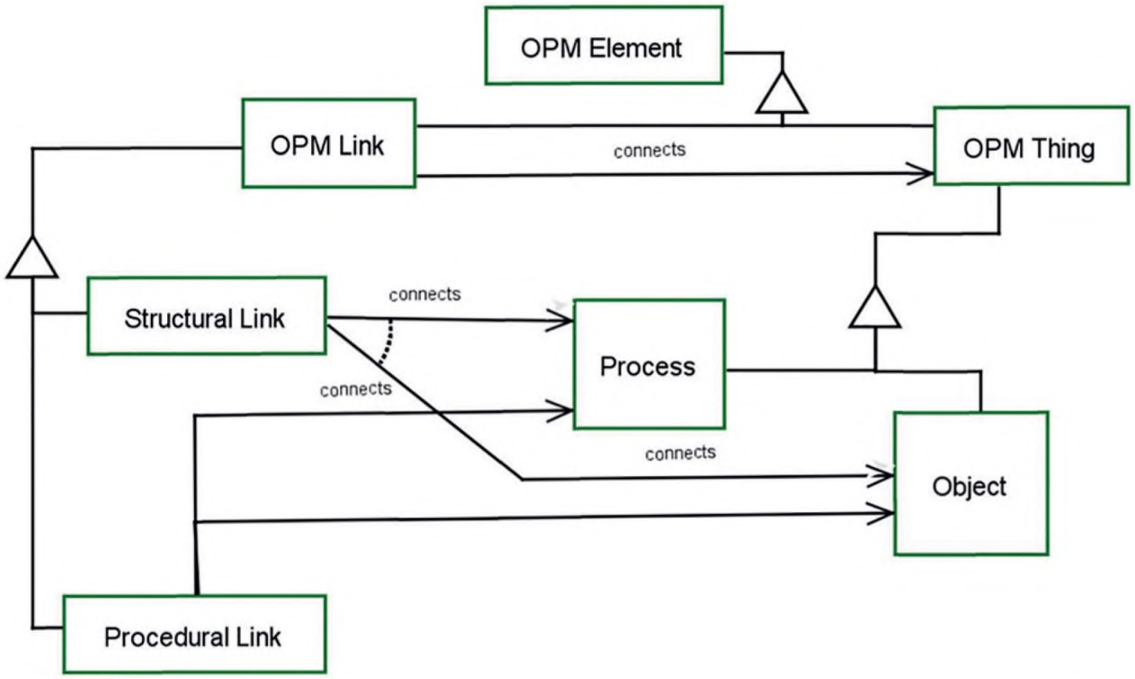
Примечание 1 — Бимодальное графически-текстовое представление OPM-модели способствует привлечению нетехнических заинтересованных сторон к выявлению требований и начальному концептуальному моделированию системы на стадии ее разработки. Это позволяет привлекать заинтересованные стороны к активному участию в разработках и обнаруживать ошибки вскоре после их непреднамеренного появления. Бимодальное представление также помогает начинающим пользователям OPM-методологии оперативно знакомиться с семантикой графического OPM-метода при проверке текста и связанной с ним графикой.

Примечание 2 — В приложении А определен OPL-синтаксис, использующий допущения ИСО/МЭК 14977.

Примечание 3 — Для большинства OPD-рисунков настоящего стандарта соответствующий раздел OPL-предложений сопровождается графической OPD-диаграммой.

6.2.2 Элементы моделирования в OPM-методологии

Элементы как основные композиционные блоки любой системы, смоделированные в OPM-методологии, должны быть двух видов: сущностей и связей. Элементы моделирования объекта и процесса определяют эти сущности в контексте модели. Моделируемые элементы связи должны определять ассоциации между сущностями в контексте модели. Объекты должны быть как без внутреннего состояния, так и с внутренним состоянием, а связи должны быть либо процедурными, либо структурными. На рисунке 1 представлен общий обзор OPM-метамодели.



OPM Element — OPM-элемент; OPM Link — OPM-связь; OPM Thing — OPM-сущность; connects — соединяет; Structural Link — структурная связь; Process — процесс; Object — объект; Procedural Link — процедурная связь

Рисунок 1 — Общая структура OPM-метамодели

В OPM-модели ее элементы должны иметь уникальные символы, текстовое выражение, синтаксические ограничения и семантическую интерпретацию. В рамках той же модели каждая моделируемая сущность также должна иметь уникальное имя, определяющее отношение к модели заинтересованных сторон, а уникальный источник и назначение сущности должны различаться за счет использования различных связей (тегированных связей). Смоделированная связь вместе с источником и назначением сущности должны быть OPM-структурой, которая должна иметь соответствующее OPL-предложение.

Сразу же после идентификации смоделированная сущность может появляться в любом соответствующем контексте этой сущности, а также может появляться несколько раз в контексте для повышения степени ее понимания.

6.2.3 OPM-сущности: объекты и процессы

Объект должен быть сущностью, который после его создания будет существовать (или может существовать) физически или информационно. Объединения объектов должны составлять структуру объекта моделируемой системы, т. е. статический, структурный аспект системы. Состояние объекта должно быть его конкретной ситуативной классификацией в определенный момент времени на протяжении всего срока службы. В любой момент времени объект может находиться в одном из своих состояний или в процессе перехода между двумя своими состояниями (как следствие процесса, воздействующего в настоящее время на этот объект).

Процесс должен быть сущностью, которая характеризует преобразование объектов в данной системе. Этот процесс всегда связан (и происходит) с одним или несколькими объектами; он никогда не протекает изолированно. Процесс преобразует объекты, создавая их, используя их или изменяя их состояние. Таким образом, процессы дополняют объекты, создавая динамический, поведенческий аспект системы.

Примечание — Процессы проверки для определения того, какие подпроцессы действуют на момент проверки, позволяют выявлять состояние процесса. OPM-методология не определяет в явном виде модель состояния процесса (см. метамодель процесса в приложении С).

6.2.4 OPM-связи: процедурные и структурные

Процедурные связи должны характеризовать процедурные взаимоотношения, которые будут указывать способ работы системы при выполнении надлежащей функции, назначении зависящих от времени или условно инициированных процессов, которые преобразуют объекты.

Структурные связи должны характеризовать структурные взаимоотношения, которые будут определять ассоциации, сохраняемые в системе в течение по крайней мере определенного промежутка времени (т. е. статический аспект системы), и не должны зависеть от временных условий.

6.2.5 OPM-управление контекстом

OPM-методология предоставляет детальные способы управления контекстными границами модели для облегчения понимания и повышения ее ясности. Исходя из исходного контекста функциональной модели, разработчик модели должен использовать уточненную структуру объекта и декомпозицию процесса с целью детализации модели на каждом этапе пошаговой детализации, содержащей контекстный акцент.

Для реализации системной функции в иерархическую сеть субпроцессов необходимо включить ряд нетривиальных процессов. Иерархия процессов должна частично упорядочивать их, т. е. некоторые процессы должны заканчиваться до начала выполнения других процессов, тогда как некоторые другие процессы могут протекать параллельно или альтернативно. При любой степени детализации в иерархии процессов любой из них в системе должен обеспечивать или способствовать функциональной значимости как части своего предшествующего процесса.

Фундаментальной единицей при управлении контекстом является OPD-диаграмма, которая характеризует моделирование элементов конкретного контекста. Разворачивание и увеличение масштаба новой диаграммы обеспечивают получение структурных и процедурных связей между контекстами. Хотя любая OPD-диаграмма может содержать любое количество элементов, в ней должны появляться только те элементы, которые имеют отношение к конкретному контексту.

Управление контекстом для имен, меток сущностей и связей должны составлять полную OPM-модель, в которой отдельные фрагменты модели согласуют связи и взаимодействия между элементами модели, обуславливающими модель поведения. Имена сущностей должны быть уникальными при управлении контекстом.

6.2.6 Реализация OPM-модели

6.2.6.1 Зависимость концептуальных моделей от динамических моделей

При построении модели с помощью OPM-методологии разработчики моделей должны понимать различия между концептуальной моделью, которую они создают, и оперативным появлением этой модели, которую они могут использовать для оценки поведения системы. Практикующие разработчики моделей обладают интуитивным пониманием этих различий, реально воспринимая появление смоделированных экземпляров элементов при создании модели, даже если эти элементы весьма абстрактны. Тем не менее те из них, кто не знаком с моделированием, используя OPM-методологию, могут посчитать данную спецификацию несколько запутанной.

Модель OPM-методологии является формальной основой, в которой экземпляры объекта и процесса взаимодействуют между собой посредством связей. Поскольку OPM-модель обладает формальной основой, сходной со структурой системы, а элементы модели также взаимодействуют между собой с помощью связей, то разработчик моделей может имитировать поведение системы путем оперативного создания экземпляра объекта/процесса, а затем отслеживать поток сигналов исполнения, осуществляемый посредством соединений и семантических OPM-правил. Наличие экземпляров сущности переводит абстрактные концептуальные модели в более конкретную динамическую форму.

В приложении D описаны OPM-средства поддержки мероприятий по моделированию, однако поскольку пользователи настоящего стандарта будут строить свои собственные OPM-модели, они должны иметь в виду, что поведение моделируемой системы реализуется только при существовании операционных экземпляров сущностей. Появление связи между двумя сущностями не будет подразумевать модель поведения до тех пор, пока не будут существовать эти операционные экземпляры. Термин «время исполнения», т. е. время, когда эти операционные экземпляры сущностей будут существовать, подразумевается в каждом положении настоящего стандарта.

Примечание — Термин «экземпляр» также может встречаться и с другим значением в представлении отношения «классификация-инстанцирование». В этом случае «экземпляр» можно будет считать типичным объектом класса *refinee*.

6.2.6.2 Реализация OPM-модели

Концептуальная основа OPM-методологии имеет возможности для выполнения модельных расчетов. Для успешного использования этих возможностей разработчику модели необходимо понимать различия между моделью как представление шаблона структуры и поведением экземпляра модели, реализуемым для выполнения какой-либо функции, для которой эта модель является шаблоном. Модель имеет архитектурную форму, частично основанную на компоновке структуры и процедуры,

которую разработчик модели может подробно расширять по мере разработки своей модели. Эта модель, отражающая соответствующие детали, является реализуемой в качестве имитационной модели, т. е. она способна реализовывать ресурсы с помощью процессов преобразования объектов, а также создавать функциональную ценность для бенефициара.

6.2.6.3 OPD-навигация и OPL-компоновка

Настоящий стандарт содержит средства создания модельных OPM-диаграмм и соответствующих OPL-текстов. Способы детализации и разворачивания, описанные в разделе 14, дают возможность связывать OPD-диаграммы с соответствующими OPL-текстами для выражения связей в текстовой форме. Тем не менее поскольку существует множество способов маркировки связей, часть из которых могут быть специфическими при инструментальной реализации, в разделе 14 не определены метки для их присвоения и идентификации последовательных иерархических уровней, связей между соответствующими OPD-диаграммами или с соответствующими OPL-сегментами.

7 OPM-синтаксис и семантика сущностей

7.1 Объекты

7.1.1 Описание объектов

Объект должен быть сущностью, которая реально существует или имеет возможность физического или информационного существования. С временной точки зрения существование объекта должно быть устойчивым. До тех пор пока никакой процесс не воздействует на объект, он должен оставаться в своем явном или неявном состоянии.

OPM-объект является абстрактным идентификатором категории для шаблона структуры, свойств и особенностей (признаков), т. е. атрибутов и операций, которые применимы к операционным объектам-экземплярам данной категории. В рамках действующих ограничений модели может существовать любое неотрицательное число действующих объектов-экземпляров.

7.1.2 Представление объектов

Прямоугольник, содержащий надпись — имя объекта, должен графически символизировать наличие объекта модели. На рисунке 2 приведен пример отображения объекта Vehicle Occupant Group (Группа пассажиров транспортного средства). В OPL-тексте имя объекта представляется надписью с заглавными буквами у каждого слова имени.

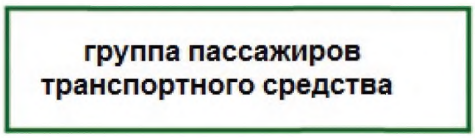


Рисунок 2 — Графическое обозначение объекта

Примечание — Соглашения относительно наименований процессов обсуждаются в В.6.3.

7.2 Процессы

7.2.1 Описание процесса

Процесс должен быть сущностью, которая способна преобразовывать один или несколько объектов. При этом преобразованием может быть формирование (конструирование, создание), действие или потребление (разрушение, уничтожение). Процесс должен иметь положительное значение для времени выполнения операций.

OPM-процесс является абстрактным идентификатором категории для шаблона преобразования. Для конкретной операционной реализации экземпляр процесса является специфическим проявлением шаблона процесса, который определяет соответствующую категорию. Операционный процесс-экземпляр позволяет преобразовывать один или несколько операционных объектов-экземпляров.

Примечание 1 — Один процесс с помощью инициализирующей связи (см. 9.5.2.5.2) может непосредственно вызывать другой процесс, что будет приводить к появлению вызванного процесса, создающего временный объект, который немедленно будет потребляться вызываемым процессом.

Примечание 2 — Воздействие процесса на объект, как правило, приводит к изменению состояния этого объекта, однако существуют постоянные процессы, воздействие которых состоит в поддержании их текущего

состояния. Вместо того чтобы вызывать изменения, семантикой устойчивого процесса является поддержание объекта в его текущем состоянии.

Пример — Процесс Existing (Существование) является наиболее известным устойчивым процессом, описывающим статическое (неявное) состояние существования процесса. Примеры других устойчивых процессов: Holding (Удержание), Maintaining (Поддержание), Keeping (Содержание), Staying (Сдерживание), Waiting (Ожидание), Prolonging (Продление), Extending (Расширение), Delaying (Задерживание), Occupying (Захватывание), Persisting (Существование), Continuing (Продолжение), Supporting (Содействие), Withholding (Остановка) и Remaining (Сохранение). Для биологических объектов процесс Existing (Существование) предполагает Living (Проживание) и предназначен для активного поддержания необходимых жизненных функций.

7.2.2 Представление процессов

Эллипс с меткой внутри (с именем процесса) графически символизирует наличие абстрактной категории процесса. На рисунке 3 показан процесс Automatic Crash Responding (Автоматическое реагирование на аварию). В OPL-тексте имя процесса необходимо выделять жирным шрифтом с заглавными буквами у каждого слова.

Рисунок 3 — Графическое представление процесса

Примечание — Соглашения относительно наименований процессов обсуждаются в В.6.3.

7.3 OPM-сущности

7.3.1 Определение OPM-сущностей

OPM-сущность должна быть либо объектом, либо процессом, причем последние во многих отношениях симметричны и имеют много общего с точки зрения взаимоотношений, таких как агрегирование, обобщение и характеризация. Объект существует тогда, когда процесс протекает с одним или несколькими объектами. OPM-объекты и OPM-процессы зависят друг от друга в том смысле, что процесс необходим для преобразования объекта, тогда как для преобразования по меньшей мере одного объекта должен протекать или возникать процесс.

7.3.2 Тест типа «объект-процесс»

Для эффективного применения OPM-методологии разработчик модели должен четко различать объекты и процессы, что является необходимым условием успешности анализа и проектирования системы. По умолчанию объект необходимо идентифицировать с помощью существительного. Тест типа «объект-процесс» дает разработчикам моделей критерии различения этих существительных, используемых для процессов, от существительных, используемых для объектов. Получение правильного ответа на вопрос о том, относится ли данное существительное к объекту или процессу, является важным и фундаментальным для OPM-методологии.

Для того чтобы считаться процессом, слово или словосочетание должно отвечать каждому из трех следующих критериев:

- наличие временной ассоциации, при которой используемое существительное ассоциируется с временем;
- наличие отглагольной ассоциации, при которой используемое существительное получают из глагола, или имеет общий корень с глаголом, или имеет синоним, который ассоциируется с глаголом;
- наличие преобразования объекта, при котором используемое существительное возникает, происходит, представляет, выполняет, преобразует, изменяет или модифицирует по меньшей мере один объект или поддерживает его в текущем состоянии.

Пример — Flight (Полет) — это существительное, которое характеризует процесс, поскольку оно отвечает всем трем критериям данного теста, а именно:

a) оно имеет временную ассоциацию;

b) оно ассоциируется с глаголом «летать»;

c) оно преобразует существительное Airplane (Самолет) путем изменения его атрибута местонахождения от источника до места назначения.

7.3.3 Общие свойства OPM-сущности

Все OPM-сущности должны обладать следующими тремя общими свойствами:

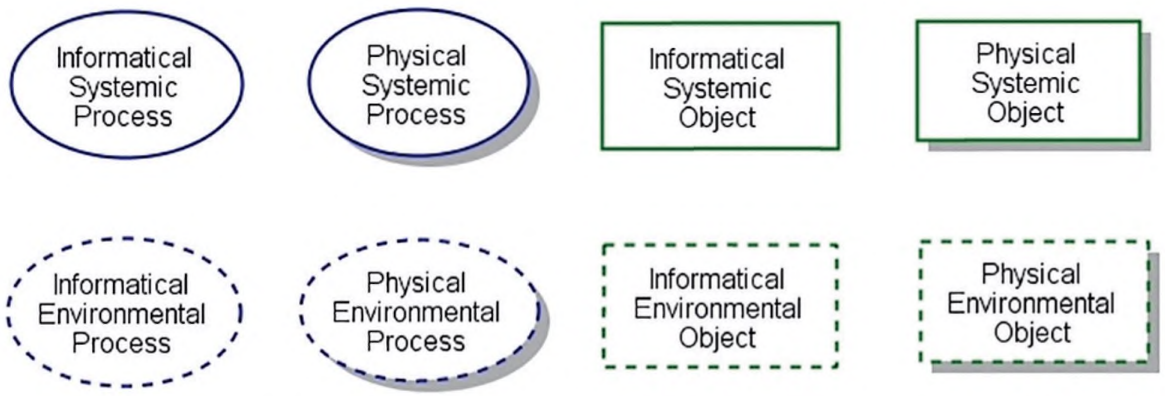
- Perseverance (устойчивость): свойство, которое относится к стабильности сущности и характеризует, статична или динамична ли она как процесс. Соответственно, допустимое значение для атрибута сущности Perseverance — это статическое или динамическое.

- Essence (смысл): свойство, которое относится к природе сущности и характеризует, физическая она или информационная. Соответственно, допустимое значение общего атрибута сущности Essence — это физическое или информационное.

- Affiliation (принадлежность): свойство, которое относится к области применения сущности и характеризует, системная она (т. е. является ли она частью системы) или относится к внешней среде (т. е. является ли она частью среды, окружающей систему). Соответственно, атрибут сущности Affiliation — это системное или относящееся к внешней среде.

Примечание — Хотя объекты являются постоянными (т. е. они обладают статической устойчивостью), а процессы являются временными (т. е. они обладают динамической устойчивостью), могут существовать крайние примеры постоянных процессов (см. 7.2.1), а также временные объекты (см. 9.5.2.5.1).

Графически, как это показано на рисунке 4, наличие «тени» будет обозначать физические OPM-сущности, а пунктирные линии — OPM-сущности внешней среды. Все восемь возможных комбинаций Perseverance-Essence-Affiliation из общих свойств OPM-сущности показаны на рисунке 4. Нижняя часть рисунка 4 выражает (слева направо и сверху вниз) OPL-предложения, соответствующие указанным графическим элементам.



Процесс **Informatical Systemic Process** — это информационный системный процесс.
Процесс **Physical Systemic Process** — это физический системный процесс.
Объект **Informatical Systemic Object** — это информационный системный объект.
Объект **Physical Systemic Object** — это физический системный объект.
Процесс **Informatical Environmental Process** — это информационный процесс внешней среды.
Процесс **Physical Environmental Process** — это физический процесс внешней среды.
Объект **Informatical Environmental Object** — это информационный объект внешней среды.
Объект **Physical Environmental Object** — это физический объект внешней среды.

Рисунок 4 — Комбинация из общих атрибутов OPM-сущностей

7.3.4 Значения по умолчанию для общих свойств сущности

Значение по умолчанию общего свойства Affiliation (принадлежность) должно быть систематическим.

Любая нетривиальная система, как правило, обладает большим количеством объектов и процессов с одними и теми же общими значениями сущности Essence (смысл).

Пример — Системы обработки данных являются информационными, хотя они имеют физические компоненты. Система пассажироперевозок, например железнодорожная сеть или авиационная система, является физической, хотя и обладает информационными компонентами.

Первичное свойство сущности Essence системы должно быть таким же, как и у большинства сущностей в рамках данной системы.

Значением по умолчанию общего свойства сущности Essence в рамках системы должна быть первичная сущность Essence данной системы.

Примечание — Вспомогательное средство может давать возможность разработчику модели определять первичную сущность Essence системы как средства для установления по умолчанию общего атрибута сущности Essence.

OPL-язык, соответствующий диаграмме, не отражает значения по умолчанию общих свойств сущности, если она не имеет связей хотя бы еще с одной сущностью, например, в процессе моделирования. Сразу же после появления связей с другими сущностями первая из этих сущностей, формирующая общие свойства, должна (в случае необходимости) объединяться в OPL-фразы, описывающие эти связи.

7.3.5 Состояния объекта

7.3.5.1 Объекты с и без внутреннего состояния

Состоянием объекта следует считать возможную ситуацию, в которой может оказаться объект. Состояние объекта имеет смысл только в контексте объекта, к которому оно принадлежит (т. е. объект, который обладает определенными состояниями).

Объектом без внутреннего состояния следует считать объект, который не имеет определенного состояния.

Объектом с внутренним состоянием следует считать объект с заданной совокупностью допустимых состояний. Во временной модели (runtime model) в любой момент времени любой экземпляр операционного объекта с внутренним состоянием находится в определенном допустимом состоянии (или существует в состоянии перехода между двумя допустимыми состояниями как следствие воздействия процесса на данный объект).

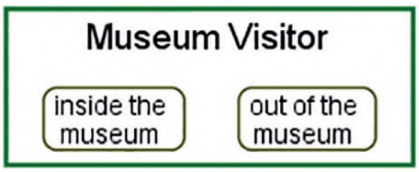
Примечание 1 — В зависимости от поведения модели операционные экземпляры объекта могут находиться в различных состояниях.

Примечание 2 — Соглашения относительно наименования состояний объектов обсуждаются в В.6.4.

7.3.5.2 Представление состояния объекта

Прямоугольник с закругленными углами и пометкой с обозначением объекта, к которому он принадлежит, графически отображает состояние данного объекта. В OPL-тексте обозначение состояния объекта должно выполняться жирным шрифтом строчными буквами.

Пример — На рисунке 5 изображен объект Museum Visitor (Посетитель музея), имеющий два состояния — inside the museum (внутри музея) и out of the museum (вне музея). Нижеприведенное графическое представление должно быть эквивалентно соответствующему OPL-предложению.



Объект **Museum Visitor** (Посетитель музея) может находиться в состоянии **inside the museum** (внутри музея) или **out of the museum** (вне музея).

Рисунок 5 — Представление объекта с двумя внутренними состояниями

7.3.5.3 Начальное/конечное состояния объекта и его состояние по умолчанию

Начальным состоянием объекта должно быть его состояние в момент начала работы системы или его состояние по факту его формирования системой в процессе работы. Конечным состоянием объекта должно быть его состояние в момент завершения системой своей работы или по факту его использования в процессе работы. Состоянием объекта по умолчанию должно быть состояние, в котором объект, скорее всего, будет находиться на этапе выборочной проверки.

Примечание 1 — Объект может не иметь начального состояния или же иметь несколько начальных/конечных состояний или состояний по умолчанию. Одно и то же состояние может быть произвольной комбинацией из начального и/или конечного состояний и/или состояния по умолчанию.

Примечание 2 — Наличие начальных и конечных состояний особенно полезно для тех объектов, которые обладают жизненным циклом, например, для таких объектов, как продукт, организм или система.

Примечание 3 — Если объект имеет несколько начальных состояний, то каждому начальному состоянию можно присвоить вероятность нахождения создаваемого объекта в определенном состоянии (см. 12.7).

7.3.5.4 Представление начального/конечного состояния объекта и его состояния по умолчанию

Графически утолщенный контур прямоугольника со скругленными углами обозначает начальное состояние объекта, двойной контур — конечное состояние объекта, а однонаправленная диагональная стрелка — состояние объекта по умолчанию. Соответствующие OPL-предложения поясняют определения состояний.

Пример — На рисунке 6 приведен объект Specification (Спецификация) с начальным/конечным состояниями и состоянием по умолчанию, а под этим рисунком — соответствующие OPL-предложения.

Состояние **preliminary** (предварительное) объекта **Specification** (Спецификация) является начальным.

Состояние **approved** (одобренное) объекта **Specification** (Спецификация) является состоянием, устанавливаемым по умолчанию.

Состояние **cancelled** (аннулированное) объекта **Specification** (Спецификация) является конечным.

Рисунок 6 — Объект с начальным/конечным состояниями и состоянием по умолчанию

7.3.5.5 Значения атрибутов

Поскольку атрибут является объектом, то значение атрибута должно соответствовать состоянию в том смысле, что значение должно быть состоянием атрибута. Объект может иметь атрибут, который представляет собой другой объект, в течение некоторого промежутка времени (в период существования объекта) имеющий этот атрибут, а значение этого атрибута может быть состоянием другого объекта.

Пример — Если считать объект Temperature (Температура) с состоянием degrees Celsius (в градусах по шкале Цельсия) атрибутом объекта Engine (Двигатель), то значением этого атрибута будет 75.

Примечание 1 — Поскольку атрибут является структурированным объектом, то допустимым значением этого атрибута будет элемент множества допустимых состояний этого объекта. С помощью пронумерованного перечня или набора из одного или нескольких диапазонов значений можно определять множество допустимых значений атрибута.

Примечание 2 — Значение свойства должно быть зафиксировано и не может изменяться во время работы с моделью.

Атрибуты со значениями, выраженными в соответствующих единицах измерений, должны графически выражать единицу измерений на OPD-диаграмме (в скобках под именем атрибута объекта), а также единицу измерений в тексте — после имени атрибута объекта в соответствующих OPL-предложениях, например, объект **Temperature** (Температура) с состоянием **degrees Celsius** (в градусах по шкале Цельсия).

8 Анализ синтаксиса и семантики OPM-связей

8.1 Обзор процедурных связей

8.1.1 Виды процедурных связей

Процедурная связь должна быть одной из трех видов:

- преобразующая связь (transforming link): связь, которая соединяет объект типа *transformee* (объект, который преобразуется процессом) или одно из его состояний с процессом преобразования

объекта модели, а именно с процессом формирования, использования или изменения состояния этого объекта в результате выполнения процесса;

- разрешающая связь (enabling link): связь, которая соединяет объект типа *enabler* (объект, который позволяет выполнять процесс, но не преобразуется этим процессом), т. е. агент, инструментальное средство или их состояние, с процессом моделирования, обеспечивающего наличие этого процесса;

- управляющая связь (control link): связь, которая является преобразующей или разрешающей связью с дополнительной семантикой и предназначена для управления исполнением с целью моделирования события, которое будет инициализировать связанный процесс, условия для поддержания процесса или соединения двух процессов, обозначающих инициализацию или исключение.

Примечание — Объекты типа *transformee* и *enabler* являются ролями (функциями) объекта, которые могут иметь отношение к процессу, с которым они связаны, поэтому последний может выполнять роль объекта типа *enabler* для одного процесса и роль объекта типа *transformee* — для другого процесса.

8.1.2 Принцип уникальности процедурной связи OPM-методологии

Каждый процесс должен соединяться с преобразующей связью по меньшей мере одного объекта или его состояния. При любой конкретной степени абстракции объект или любое из его состояний должны выполнять только роль в качестве элемента модели по отношению к процессу, с которым он связан: объектом может быть объект типа *transformee*, *enabler*, инициатор или условный объект. При заданной степени абстракции объект или его состояние должны устанавливать связь с процессом только с помощью одной процедурной связи.

8.1.3 Процедурные связи, зависящие от состояния объекта

Каждая процедурная связь может квалифицироваться как процедурная связь, зависящая от состояния объекта, которая должна связывать процесс с определенным состоянием объекта.

8.2 Операционная семантика и поток команд управления исполнением

8.2.1 Механизм управления событиями, условиями, действиями

Парадигма Event-Condition-Action (события-условия-действия) должна обеспечивать операционную OPM-семантику и поток команд управления исполнением. В момент создания или возникновения объекта с системной точки зрения (или в момент перехода объекта в определенное состояние) должно происходить событие. Во время выполнения те объекты, которые являются источником связи с процессом, например, средство реализации процесса, возникновение события должно инициализировать оценку предварительных условий для каждого процесса, с которым объект связывается в качестве источника этой связи.

В начале оценки предварительных условий для процесса событие должно прекращать свое существование в этом процессе. Если (и только если) в результате оценки устанавливается соответствие предварительным условиям, то необходимо запустить процесс и выполнять действия.

Начальные характеристики процесса имеют два предварительных условия/требования:

a) наличие инициализирующего события и

b) выполнение предварительных условий.

Таким образом, события и предварительные условия/требования совместно определяют OPM-поток команд управления исполнением процесса.

Примечание — Инициализация и исключение являются событиями-условиями-действиями, которые происходят только между процессами.

Поток сигналов контроля исполнения должен быть следствием последовательностей Event-Condition-Action, которые начинаются с инициализации системной функции внешним событием и заканчиваются после завершения действия этой функции.

8.2.2 Наборы предварительно и апостериорно обработанных объектов

Набор предварительно обработанных объектов позволяет определять предварительные условия, предназначенные для их выполнения перед началом процесса. Эта задача может быть комплексной и включать сложные логические выражения или просто предполагать существование одного или нескольких объектов, возможно, в определенных состояниях. Типичными объектами в этом наборе являются объекты типа *consumees*, т. е. объекты, потребляемые процессом, и объекты типа *affectees*, т. е. объекты, подвергаемые влиянию со стороны процесса, и средства реализации процесса. Некоторые из этих объектов могут иметь дополнительное условие относительно потока сигналов контроля исполнения, т. е. условную связь. Каждый процесс должен иметь набор предварительно обработанных объектов по крайней мере с одним объектом, возможно, находящимся в определенном состоянии.

Набор апостериорно обработанных объектов должен определять апостериорные условия, предназначенные для их выполнения по завершении. Эта задача может быть комплексной и включать сложные логические выражения или просто предполагать существование одного или нескольких объектов, возможно, в определенных состояниях. Типичными объектами в этом наборе являются объекты типа *resultees*, т. е. объекты, которые образуются в данном процессе, объекты типа *affectees*, т. е. объекты, которые подвергаются влиянию со стороны процесса. Каждый процесс должен иметь набор апостериорно обработанных объектов по меньшей мере с одним объектом, возможно, находящимся в определенном состоянии.

Примечание 1 — Взаимное пересечение указанных наборов объектов для одного и того же процесса включает в себя объекты типа *enablers* и *affectees*. Объекты типа *consumees* являются только элементами набора предварительно обработанных объектов, в то время как объекты типа *resultees* — только элементами набора апостериорно обработанных объектов.

Примечание 2 — Семантика операционных экземпляров объектов в имеющемся наборе объектов описана в 14.2.2.4.4.

8.2.3 Сравнение пропущенной семантики состояния с ожидаемой семантикой безусловных связей

Процесс для набора предварительно обработанных объектов включает в себя как условные связи (см. 9.5.3), так и безусловные связи, т. е. процедурные связи без модификатора состояния. Отличительным аспектом условных связей является их «пропущенная семантика», которая предусматривает пропуск или обход процесса, если объект-источник операционного экземпляра в условной связи отсутствует. Без квалификации условной связи отсутствие этого экземпляра заставляет ждать другого события и операционных экземпляров всех объектов-источников (возможно, в определенном состоянии), тем самым обеспечивая выполнение предварительных условий.

Наличие одной или нескольких безусловных связей (а также одной или нескольких условных связей) необходимо для выполнения предварительных условий и начала процесса, однако при наличии одной или нескольких безусловных/условных связей с невыполненными предварительными условиями может возникать конфликт между семантикой ожидания для первых связей и пропущенной семантикой для вторых связей. Для разрешения этого конфликта пропущенная семантика для условных связей должна быть сильнее, чем семантика ожидания для безусловных связей, а поток сигналов управления исполнением должен обходить процесс, который не начинает действовать или создавать исключение.

Даже если только одно из условий, связанное с условными связями и с процессом, не существует, оценка выполнения предварительных условий должна признаваться недействительной, в этом случае контроль исполнения пропустит процесс, а событие произойдет для последующего процесса (процессов) с помощью инициализирующей связи определенного вида (см. 9.5.2.5 и 14.2.2).

Примечание 1 — Не существует результирующей событийной связи или результирующей условной связи, поскольку имеются исходящие процедурные связи, относящиеся к набору апостериорно обработанных объектов. После завершения процесса он создает набор этих объектов без дополнительного условия, так что нет условия относительно создания объектов типа *resultees* или изменения объектов типа *affectees*. Создание объекта (возможно, в определенном состоянии) в наборе апостериорно обработанных объектов может послужить в качестве события или условия для последующего процесса (процессов).

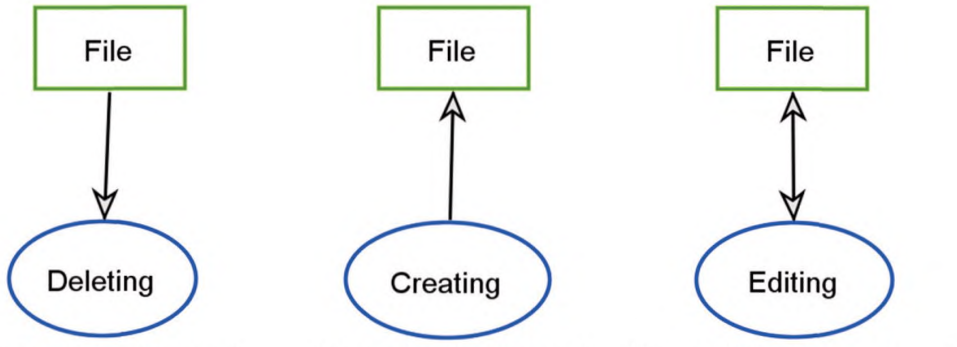
Примечание 2 — Для обеспечения надежного потока управления исполнением при любых обстоятельствах разработчик модели может имитировать процесс его преждевременного окончания без завершения процедуры обработки исключительных ситуаций (см. 9.5.4).

9 Процедурные связи

9.1 Преобразующие связи

9.1.1 Виды преобразующих связей

Преобразующая связь (transforming link) должна определять соединение между процессом и его объектом типа *transformee* (т. е. объектом, который использует, создает или изменяет состояние процесса). Рисунок 7 иллюстрирует эти три вида преобразующих связей, и ниже — соответствующие OPL-предложения.



Процесс **Deleting** (Удаление) использует объект **File** (Файл)

Процесс **Creating** (Создание) создает объект **File** (Файл)

Процесс **Editing** (Редактирование) взаимодействует с объектом **File** (Файл)

Рисунок 7 — Преобразующие связи (transforming links): слева — потребительская связь (consumption link), посередине — результирующая связь (result link), справа — взаимодействующая связь (effect link)

Объект типа *transformee* должен выполнять свою роль по отношению к данному процессу. Тот же объект в другом процессе может выполнять иную роль.

9.1.2 Потребительская связь

Потребительская связь должна быть преобразующей связью, определяющей, что связанный процесс потребляет (разрушает, уничтожает) связанный объект типа *consumee*.

Графически однонаправленная стрелка на рисунке 7, указывающая связь от объекта типа *consumee* к потребительскому процессу, должна обозначать потребительскую связь.

Синтаксис OPL-предложения для потребительской связи должен выражаться следующим образом: Процесс **Processing** (Обработка) потребляет объект типа **Consumee**.

Существование объекта типа *consumee* должно быть предварительным условием (или его частью) активизации процесса. Если объект типа *consumee* не существует, т. е. не существует операционный экземпляр объекта типа *consumee*, то процесс инициализации должен ожидать появления этого объекта.

Потребление должно осуществляться сразу же после активации процесса, если разработчику модели с течением времени снова не потребуется моделирование потребления этого объекта. В этом случае потребительская связь должна иметь свойство, которое будет указывать на скорость потребления объекта типа *consumee*, а сам объект должен иметь атрибут, который будет указывать на доступное количество.

Примечание 1 — Разработчик модели может делать исключение, если число объектов будет меньше нормы времени для ожидаемой продолжительности процесса.

Примечание 2 — Обозначения свойств связей см. в 11.1.

Пример 1 — Steel Rod (Стальной пруток) является объектом типа consumee для процесса Machining (Обработка), который создает объект типа resultee Shaft (Вал). После активации процесса Machining он будет потреблять объект Steel Rod.

Пример 2 — Water (Вода) является объектом типа consumee для процесса Irrigating (Орошение). Объект типа consumee имеет атрибут Quantity (Количество), измеряемое в литрах, со значением 1000, а потребительская связь имеет свойство Flow Rate (Скорость потока) в л/с, со значением 50. В этом случае, если процесс Irrigating не будет прерван, то он будет длиться 20 секунд, и он будет потреблять объект Water с заданным значением свойства Flow Rate.

9.1.3 Результирующая связь

Результирующая связь должна быть преобразующей связью, определяющей, что связанный процесс создает (формирует, производит) связанный объект, который является объектом типа *resultee*.

Графически однонаправленная стрелка на рисунке 7, указывающая на связь от процесса *Creating* к объекту типа *resultee*, должна означать результирующую связь.

Синтаксис OPL-предложения для результирующей связи должен выражаться следующим образом: Процесс **Processing** (Обработка) создает объект типа **Consumee**.

Формирование объекта типа *resultee* должно происходить сразу же после завершения процесса, если разработчику модели с течением времени не потребуется моделировать создание данного объекта.

В последнем случае результирующая связь должна обладать свойством, которое будет указывать на скорость формирования этого объекта, а сам объект типа *resultee* должен иметь атрибут, который будет указывать на его доступное количество.

Примечание — Обозначения свойств связей см. в 11.1.

Пример 1 — Steel Rod (Стальной пруток) является объектом типа consumee для процесса Machining (Обработка), который формирует объект типа resultee Shaft (Вал). После завершения процесса Machining он будет формировать объект Shaft.

Пример 2 — Gasoline (Бензин) и Diesel Oil (Дизельное топливо) являются объектами типа resultees процесса Refining (Очистка), который потребляет объект Crude Oil (Сырая нефть). Объекты типа resultees Gasoline и Diesel Oil имеют один и тот же атрибут Quantity (Количество) ($м^3$). Результирующая связь от процесса Refining к объекту Gasoline обладает свойством Gasoline Yield Rate (Норма выхода бензина) в $м^3$/ч, со значением 1000, а результирующая связь от процесса Refining к объекту Diesel Oil — свойством Diesel Oil Yield Rate (Норма выхода дизельного топлива) в $м^3$/ч, со значением 800. Если предположить, что достаточно объекта Crude Oil (если процесс Refining активирован и продолжается в течение 10 ч), то это будет давать 10 000 $м^3$ Gasoline и 8000 $м^3$ Crude Oil.

9.1.4 Взаимодействующая связь

Взаимодействующей связью должна быть трансформирующая связь, указывающая на то, что связанный процесс влияет на связанный объект, которым является объект типа *affectee*, т. е. этот процесс вызывает неопределенное изменение состояния объекта типа *affectee*.

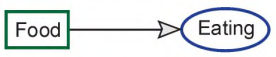
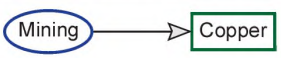
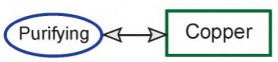
Графически двунаправленная стрелка на рисунке 7, каждый наконечник которой указывает направления между воздействующим процессом и подвергаемым воздействию объектом, должна обозначать взаимодействующую связь.

Синтаксис OPL-предложения для взаимодействующей связи должен выражаться следующим образом: Процесс **Processing** (Обработка) взаимодействует с объектом типа **Affectee**.

9.1.5 Обзор основных преобразующих связей

В таблице 1 приведены все преобразующие связи.

Таблица 1 — Обзор основных преобразующих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Потребительская связь	Процесс потребляет объект	 Объект Food (Пища) потребляется процессом Eating (Прием пищи)	Потребляемый объект	Потребляющий процесс
Результирующая связь	Процесс формирует объект	 Процесс Mining (Добыча ископаемых) создает объект Copper (Медь)	Создаваемый процесс	Созданный объект
Взаимодействующая связь	Процесс взаимодействует с объектом путем изменения одного его состояния на другое	 Процесс Purifying (Рафинирование) взаимодействует с объектом Copper (Медь)	Воздействующий процесс и подвергающийся воздействию объект являются и источниками, и получателями	

9.2 Разрешающие связи

9.2.1 Виды разрешающих связей

Разрешающей связью (enabling link) должна быть процедурная связь, указывающая на средство реализации данного процесса, которым для данного процесса должен быть объект, необходимый для возникновения этого процесса. Существование и состояние средства реализации после завершения процесса должны оставаться такими же, как и перед началом процесса.

Двумя видами разрешающей связи должны быть агентская связь (agent link) и инструментальная связь (instrument link).

Объект типа *enabler* должен существовать во время выполнения процесса, который он «разрешает». Если с системной точки зрения объект типа *enabler* во время протекания процесса перестает существовать, то процесс должен немедленно прерываться.

Примечание 1 — Средство реализации (реализатор) — это роль (функция), которую объект должен выполнять по отношению к данному процессу. Один и то же объект может быть объектом типа *enabler* для одного процесса, и объектом типа *transformee* — для другого процесса.

Примечание 2 — Для обеспечения надежного потока управления исполнением при любых обстоятельствах разработчик модели может моделировать процесс его преждевременного прерывания (до его завершения) как процесс обработки исключительных ситуаций (см. 9.5.4).

9.2.2 Агент и агентская связь

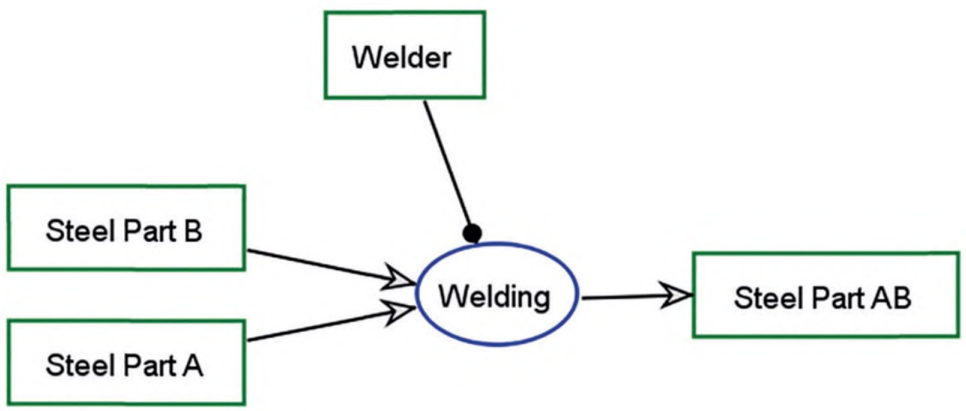
Агентом должен быть человек (либо группа людей), способный принимать надлежащие решения и взаимодействующий с системой с целью обеспечения или контроля соответствующего процесса на протяжении всего его выполнения.

Агентской связью должна быть разрешающая связь от объекта-агента к процессу, который он разрешает, указывая, что объект агент необходим для активации и выполнения связанного процесса.

Графически линия с зачерненным кружком, напоминающим черный «леденец на палочке», проходящая от объекта-агента до процесса, который он разрешает, должна обозначать агентскую связь.

Синтаксис OPL-предложения для агентской связи должен выражаться следующим образом: Объект **Agent** (агент) управляет процессом **Processing** (Обработка).

Пример 1 — На OPD-диаграмме (см. рисунок 8) Welder (Сварщик) — это объект агент для процесса Welding (Сварка). Выполнение процесса Welding для сваривания объекта Steel Part A (Стальная деталь А) с объектом Steel Part B (Стальная деталь В) для создания объекта Steel Part AB (Стальная деталь АВ) требует человека-оператора (агента) Welder. Welder — это объект агент для процесса Welding, однако Welding не преобразует объект Welder, а процесс Welding невозможен без объекта Welder.



Объект-агент **Welder** (Сварщик) управляет процессом **Welding** (Сварка).
Процесс **Welding** (Сварка) использует объекты **Steel Part A** (Стальная деталь А) и **Steel Part B** (Стальная деталь В).
Процесс **Welding** (Сварка) создает объект **Steel Part AB** (Стальная деталь АВ).

Рисунок 8 — Диаграмма с примером агентской связи

Пример 2 — На OPD-диаграмме (см. рисунок 8), если по какой-либо причине объект-агент Welder (Сварщик) перед завершением процесса Welding (Сварка) исчезает, то данный процесс должен преждевременно прерываться, а создание объекта Steel Part AB (Стальная деталь АВ) прекратится, хотя процесс Welding уже использовал объекты Steel Part A (Стальная деталь А) и Steel Part B (Стальная деталь В).

9.2.3 Инструмент и инструментальная связь

Инструментом должен быть неодушевленный объект или любое средство реализации, не способное к принятию самостоятельных решений относительно процесса или невозможное без существования и наличия самого инструмента.

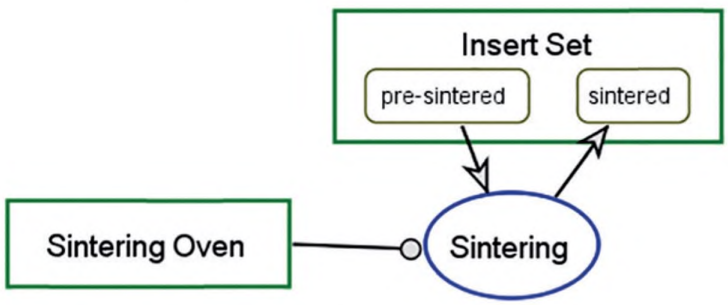
Инструментальной связью должна быть разрешающая связь от инструментального объекта к процессу, который он разрешает, указывая, что объект-инструмент необходим для активации и выполнения связанного процесса.

Графически линия с незаполненным кружком на конце, напоминающим белый «леденец на палочке», проходящая от инструментального объекта к процессу, обозначает инструментальную связь.

Синтаксис OPL-предложения для инструментальной связи должен выражаться следующим образом: Процесс **Processing** (Обработка) требует объекта **Instrument** (Инструментальное средство).

Пример 1 — Процесс Manufacturing (Производство) не может потреблять или изменять (не учитывая износ) состояние объекта Machine (Станок), который позволяет преобразовывать объект Bar Stock (Запас прутковых заготовок) в объект Machined Part (Обработанная деталь). В данном контексте Machine является инструментом для процесса Manufacturing.

Пример 2 — На OPD-диаграмме (см. рисунок 9) объект Sintering Oven (Агломерационная печь) является объектом-инструментом для объекта Insert Set (Набор вкладышей), поскольку без процесса Sintering (Спекания) их производство невозможно, однако поскольку объект Insert Set преобразуется (т. е. его состояние изменяется от предварительного к спеченному), то не принимая во внимание износ объект Sintering Oven остается неизменным в результате протекания процесса Sintering.



Объект **Insert Set** (Набор вкладышей) может находиться в состоянии **pre-sintered** (предварительно спеченное) или **sintered** (спеченное).

Процесс **Sintering** (Спекание) требует инструментального объекта **Sintering Oven** (Печь для спекания).

Процесс **Sintering** (Спекание) изменяет состояние объекта **Insert Set** (Набор вкладышей) из состояния **pre-sintered** (предварительно спеченное) на состояние **sintered** (спеченное).

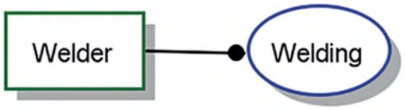
Рисунок 9 — Пример инструментальной связи

Пример 3 — На OPD-диаграмме (см. рисунок 9), если в процессе Sintering (Спекание) объект Sintering Oven (Печь для спекания) перестает существовать (например, из-за сильного растрескивания), то процесс Sintering прерывается, а объект Insert Set (Набор вкладышей) перестает находиться в состоянии sintered (спеченное), хотя он уже вышел из состояния pre-sintered (предварительно спеченное).

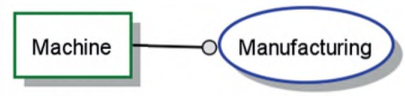
9.2.4 Обзор основных разрешающих связей

В таблице 2 приведены основные разрешающие связи.

Таблица 2 — Обзор основных разрешающих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Агентская связь	Агент — это человек или группа людей, способных вызывать процесс, с которым он связан, однако не способных преобразовываться этим процессом	 Физический объект-агент **Welder** (Сварщик) выполняет физический процесс **Welding** (Сварка)	Агент — это разрешающий объект	Разрешенный процесс

Окончание таблицы 2

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Инструментальная связь	Инструмент — это неодушевленный объект, способный вызывать процесс, с которым он связан, однако не способный преобразовываться этим процессом	 Физический объект **Machine** (Станок) требуется для физического процесса **Manufacturing** (Производство)	Инструмент — это разрешающий объект	Разрешенный процесс

9.3 Определяющие состояние преобразующие связи

9.3.1 Определяющая состояние потребительская связь

Определяющей состояние потребительской связью (consumption link) должна быть потребительская связь из определенного состояния объекта типа *consumee* к связанному процессу, который потребляет (разрушает, уничтожает) объект. Существование объекта *consumee* в определенном состоянии должно быть предварительным условием (или частью предварительного условия), предназначенным для активизации процесса. Если объект *consumee* не находится в этом состоянии, то процесс активизации должен ожидать появления объекта *consumee*, существующего в этом определенном состоянии.

Графически однонаправленная стрелка, исходящая от заданного состояния объекта к процессу, который потребляет этот объект, должна обозначать определяющую состояние потребительскую связь.

Синтаксис определяющей состояние потребительской связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) потребляет объект **specified-state Object** (Определяющий состояние объект).

Подобное потребление должно происходить сразу же после активации процесса, если разработчику модели с течением времени не потребуется моделировать потребление этого объекта. В этом случае потребляющая связь должна обладать свойством, которое будет указывать на скорость потребления объекта типа *consumee*, а сам объект типа *consumee* должен иметь атрибут, который будет указывать на его доступное количество.

Примечание 1 — Разработчик модели может сделать исключение, если количество объектов будет меньше временной нормы для ожидаемой продолжительности процесса.

Примечание 2 — Обозначения свойств связей см. в 11.1.

Пример 1 — Объект Steel Rod (Стальной пруток), находящийся в состоянии pre-heat-treated (предварительная термическая обработка), является объектом типа consumee для процесса Machining (Обработка), который формирует объект типа resultee Shaft (Вал) в том же состоянии. При этом определяющая состояние потребительская связь от процесса Machining к состоянию pre-heat-treated объекта Shaft будет обозначать соответствующую спецификацию модели.

Пример 2 — Развивая пример 1, предположим, что объект Steel Rod находится в состоянии pre-heat-treated и имеет атрибут Quantity [units] (Количество [единиц]) со значением 600. При этом определяющая состояние потребительская связь будет обладать свойством Rate [units/hour] (Скорость потребления [ед/ч]) со значением 60. При выполнении процесса Machining он за 10 ч работы потребит 600 объектов Steel Rods.

9.3.2 Определяющая состояние результирующая связь

Определяющей состояние результирующей связью (result link) должна быть результирующая связь от процесса до определяющего состояния объекта типа *resultee*, который создает (формирует, порождает) процесс. Существование объекта типа *resultee* в данном состоянии должно быть апостериорным условием (или частью апостериорного условия) после завершения процесса.

Графически однонаправленная стрелка, исходящая от процесса к заданному состоянию объекта, должна обозначать определяющую состояние результирующую связь.

Синтаксис определяющей состояние результирующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) создает объект **specified-state Object** (Определяющий состояние объект).

Подобное создание объекта типа *resultee*, находящегося в конкретном состоянии, должно происходить сразу же после завершения процесса, если разработчику модели с течением времени не потребуется моделировать формирование этого объекта. В этом случае результирующая связь должна обладать свойством, которое будет указывать на скорость создания этого объекта типа *resultee*, а сам объект типа *resultee* должен иметь атрибут, который будет указывать на его доступное количество в данном конкретном состоянии.

Примечание 1 — Обозначения свойств связей см. в 11.1.

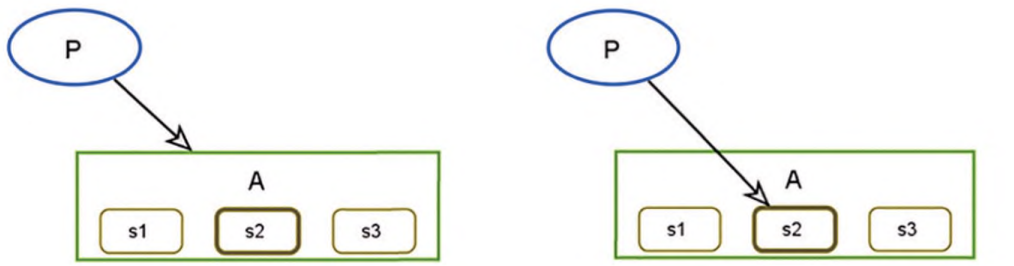
Примечание 2 — В процессе выполнения операционная модель может состоять из нескольких операционных экземпляров, каждый из которых может находиться в различных состояниях.

Пример 1 — Объект Steel Rod (Стальной пруток) в состоянии pre-heat-treated (предварительная термическая обработка) является объектом типа consumee для процесса Machining (Обработка), который создает объект типа resultee Shaft (Вал) в том же состоянии. При этом определяющая состояние результирующая связь от процесса Machining до состояния pre-heat-treated объекта Shaft будет обозначать соответствующую спецификацию модели.

Результирующая связь, создающая структурированный объект с начальным состоянием, на диаграмме должна подходить к прямоугольнику объекта (или к любому из его состояний, кроме начального).

Примечание 3 — Разработчику модели, изображенной в правой части рисунка 10, может понадобиться OPL-обозначение, однако размещение OPL-обозначения в левой части этого рисунка может уменьшать неоднозначность.

Пример 2 —



Объект **A** может находиться в состоянии **s1, s2** или **s3**. Состояние **s2** является начальным. Процесс **P** создает объект **A**.

Объект **A** может находиться в состоянии **s1, s2** или **s3**. Состояние **s2** является начальным. Процесс **P** создает состояние **s2** объекта **A**.

Рисунок 10 — Правильно (слева) и неправильно (справа) обозначенная результирующая связь с объектом, имеющим начальное состояние

9.3.3 Определяющие состояние воздействующие связи

9.3.3.1 Входные и выходные воздействующие связи

Входной связью источника должна быть связь от заданного состояния объекта (входного источника) к преобразующему процессу, тогда как выходной связью получателя должна быть связь от преобразующего процесса к заданному состоянию объекта (конечного потребителя). Указанные связи при моделировании могут создавать три возможные ситуации (в контексте одного объекта, соединенного с единственным процессом), а именно:

a) входная/выходная воздействующая связь, определяющая состояние как исходного источника, так и конечного получателя;

b) входная воздействующая связь, определяющая состояние только исходного источника;

c) входная воздействующая связь, определяющая состояние только конечного получателя.

9.3.3.2 Определяющая состояние входная/выходная воздействующая связь

Определяющей состояние входной/выходной воздействующей связью должна быть пара воздействующих связей, при которых исходная связь источника соединяется с процессом, на который влияет заданное состояние объекта типа *affectee*, а выходная связь получателя соединяется с тем же процессом с другим состоянием выходной связи получателя для того же объекта типа *affectee*. Существование

объекта типа *affectee* в состоянии исходной связи источника должно быть предварительным условием (или его частью) для оказания воздействия на активацию процесса. Существование объекта в выходном состоянии получателя должно быть апостериорным условием (или его частью) после завершения воздействия процесса.

Графически пара стрелок, одна из которых направлена от исходного состояния источника объекта типа *affectee* к воздействующему процессу (исходная связь источника), а другая, аналогичная стрелка направлена от этого процесса до выходного состояния получателя объекта типа *affectee* при завершении процесса (выходная связь получателя), будет обозначать определяющую состояние входную/выходную воздействующую связь.

Синтаксис определяющей состояние входной/выходной воздействующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) изменяет состояние объекта **Object** (Объект) из состояния **input-state** (входное состояние) на состояние **output-state** (выходное состояние).

Пример 1 — На OPD-диаграмме (см. рисунок 11) изображены определяющая состояние потребительская связь и результирующая связь. Процесс Machining (Обработка) может только потреблять объект Raw Metal Bar (Необработанные металлические прутки) в состоянии cut (отрезки) и создавать Part (Деталь) в состоянии pre-tested (предварительно испытанная). Процессы Cutting (Резка) и Testing (Испытания) являются связанными с окружающими условиями, причем процесс Cutting должен предшествовать процессу Machining для изменения состояния объекта Raw Metal Bar с состояния pre-cut (предварительная резка) на состояние cut (отрезки), тогда как процесс Testing изменяет состояние объекта Part из состояния pre-tested в состояние tested (испытанная).

Примечание 1 — В случае определяющей состояние входной/выходной результирующей связи сразу же после начала воздействия процесса он может вызывать появление объекта для выхода из своего исходного состояния источника, однако объект достигает своего выходного состояния получателя только после завершения процесса. Между началом процесса и завершением процесса объект типа *affectee* будет находиться в переходном состоянии между этими двумя состояниями.

Пример 2 — На OPD-диаграмме (см. рисунок 11) процесс Cutting (Резка) переводит объект Raw Metal Bar (Необработанные металлические прутки) из состояния pre-cut (предварительная резка) в состояние cut (резка). Пока процесс Cutting действует, состояние объекта Raw Metal Bar является переходным и связанным с процессом Cutting: процесс Cutting выбирает этот объект с состоянием pre-cut, однако не выбирает его с состоянием cut после завершения процесса. Хотя в процессе Cutting состояние объекта Raw Metal Bar является неопределенным, этот объект можно частично разрезать и повторно использовать (или же в основном разрезать и не использовать). В любом случае процесс Machining невозможен, поскольку объект не находится в состоянии cut.

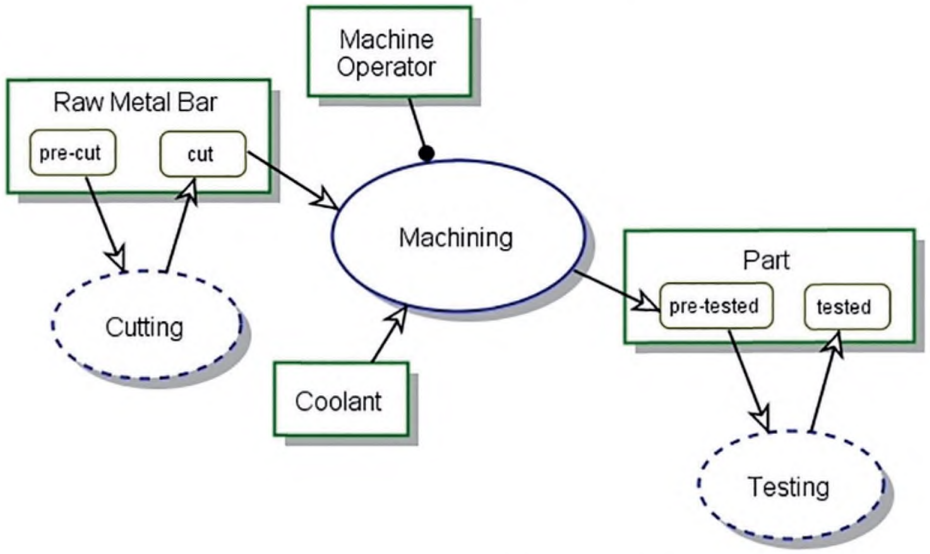


Рисунок 11, лист 1 — Определяющие состояние потребительская и результирующая связи

Физический объект **Raw Metal Bar** (Необработанный металлический пруток) имеет состояние **physical** (физическое).

Физический объект **Raw Metal Bar** (Необработанный металлический пруток) может находиться в состоянии **pre-cut** (предварительная резка) или в состоянии **cut** (отрезок).

Физический объект **Machine Operator** (Оператор станка) является объектом в состоянии **physical** (физическое).

Физический объект **Coolant** (Охладитель) является объектом в состоянии **physical** (физическое).

Физический процесс **Machining** (Обработка) является процессом в состоянии **physical** (физическое).

Физический процесс **Machining** требует физического объекта **Coolant** (Охладитель).

Физический объект **Machine Operator** (Оператор станка) выполняет физический процесс **Machining** (Обработка).

Физический объект **Part** (Деталь) является объектом в состоянии **physical** (физическое).

Физический объект **Part** (Деталь) может находиться в состоянии **pre-tested** (предварительно испытанная) или в состоянии **tested** (испытанная).

Связанный с внешней средой процесс **Testing** (Испытание) является процессом **environmental** (связанный с внешней средой) в состоянии **physical** (физическое).

Связанный с внешней средой процесс **Cutting** (Резка) является процессом **environmental** (связанный с внешней средой) в состоянии **physical** (физическое).

Связанный с внешней средой процесс **Cutting** (Резка) изменяет состояние физического объекта **Raw Metal Bar** (Необработанный металлический пруток) в состоянии **pre-cut** (предварительная резка) или в состоянии **cut** (отрезок).

Физический процесс **Machining** (Обработка) потребляет физический объект **Raw Metal Bar** (Необработанный металлический пруток) в состоянии **cut** (отрезок).

Физический процесс **Machining** (Обработка) переводит объект **Part** (Деталь) в состояние **pre-tested** (предварительно испытанная).

Связанный с внешней средой процесс **Testing** изменяет состояние объекта **Part** (Деталь) из состояния **pre-tested** (предварительно испытанная) на состояние **tested** (испытанная).

Рисунок 11, лист 2

Примечание 2 — Если активный воздействующий процесс преждевременно прерывается, т. е. процесс остается незавершенным, то состояние любого объекта типа *affectee* будет оставаться неопределенным до тех пор, пока процесс обработки исключительных ситуаций не разрешит объекту занять одно из его допустимых состояний.

9.3.3.3 Определяющая состояние входная воздействующая связь

Определяющей состояние входной воздействующей связью должна быть пара воздействующих связей, при которых исходная связь источника устанавливается с процессом, на который воздействует заданное исходное состояние объекта типа *affectee*, а выходная связь получателя устанавливается с тем же процессом для того же объекта типа *affectee* (без определения его конкретного состояния). Состояние выходной связи получателя для объекта должно быть условием по умолчанию (или частью условия по умолчанию) или же при отсутствии подобного состояния — распределение вероятности состояния объекта, которое должно определять выходное состояние получателя для этого объекта (см. 12.7).

Существование объекта типа *affectee* в исходном состоянии источника является предварительным условием (или его частью) для активации воздействующего процесса. Существование объекта типа *affectee* в любом из его состояний должно быть апостериорным условием (или частью апостериорного условия) после завершения воздействующего процесса.

Графически пара стрелок, одна из которых направлена от исходного состояния источника объекта типа *affectee* к воздействующему процессу (исходная связь), а другая, аналогичная стрелка направлена от этого процесса до объекта типа *affectee* (но не к любому его состоянию), будет обозначать определяющую состояние входную воздействующую связь.

Синтаксис определяющей состояние входной воздействующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) изменяет состояние объекта **Object** (Объект) из состояния **input-state** (исходное состояние).

9.3.3.4 Определяющая состояние выходная воздействующая связь

Определяющей состояние выходной воздействующей связью должна быть пара воздействующих связей, при которых исходная связь источника устанавливается с процессом, на который воздействует объект типа *affectee* (без указания конкретного состояния), а выходная связь получателя устанавливается с тем же процессом с выходным состоянием получателя того же объекта типа *affectee*. Существование объекта типа *affectee* должно быть предварительным условием (или частью предварительного условия) для активации воздействующего процесса. Существование объекта типа *affectee* при выходном состоянии получателя должно быть апостериорным условием (или частью апостериорного условия) после завершения воздействующего процесса.

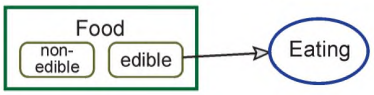
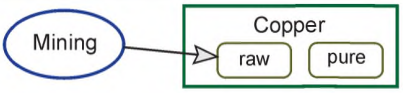
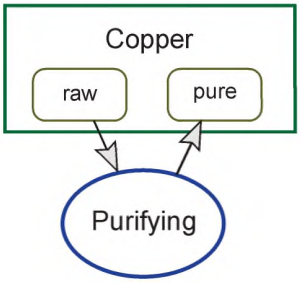
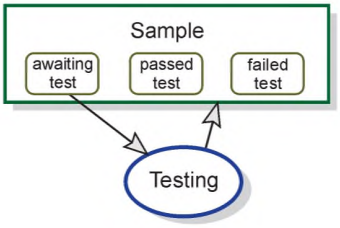
Графически пара стрелок, одна из которых направлена от объекта типа *affectee* (без определения конкретного состояния), а другая, аналогичная стрелка направлена от этого процесса до выходного состояния получателя (выходная связь), будет обозначать определяющую состояние выходную воздействующую связь.

Синтаксис определяющей состояние воздействующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) изменяет состояние объекта **Object** (Определяющий состояние объект) на состояние **specified-state** (определяющее состояние).

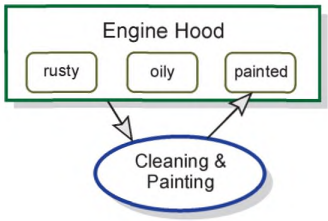
9.3.4 Обзор определяющих состояние преобразующих связей

В таблице 3 приведены все определяющие состояние преобразующие связи.

Таблица 3 — Обзор определяющих состояние преобразующих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Определяющая состояние потребительская связь	Процесс потребляет объект, только если объект находится в заданном состоянии	 Процесс **Eating** (Прием пищи) потребляет объект **Food** (Пища) в состоянии **edible** (съедобная)	Состояние объекта типа *consumee*	Процесс
Определяющая состояние результирующая связь	Процесс создает объект в заданном состоянии	 Процесс **Mining** (Добыча руды) придает объекту **Copper** (Медь) состояние **raw** (необогащенная)	Процесс	Состояние объекта типа *resultee*
Пара определяющих состояние входной/выходной воздействующих связей (состоит из одной определяющей состояния *входной связи* и одной определяющей состояние *выходной связи*)	Процесс изменяет состояние объекта из заданного исходного состояния (посредством *входной связи*) на заданное выходное состояние (посредством *выходной связи*)	 Процесс **Purifying** (Рафинирование) изменяет состояние объекта **Copper** (Медь) из состояния **raw** (необогащенная) на состояние **pure** (чистая)	Состояние источника объекта типа *affectee* Воздействующий процесс	Воздействующий процесс Состояние получателя для объекта типа *affectee*
Пара определяющих состояние входных воздействующих связей (состоит из одной определяющей состояния *входной связи* и одной не определяющей состояние *выходной связи*)	Процесс изменяет состояние объекта из заданного исходного состояния в любое выходное состояние	 Процесс **Testing** (Испытание) изменяет состояние объекта **Sample** (Образец) из состояния **awaiting test** (ожидание испытания)	Состояние источника для объекта типа *affectee* Воздействующий процесс	Воздействующий процесс Объект типа *affectee*

Окончание таблицы 3

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Пара определяющих состояние выходных действующих связей (состоит из одной не определяющей состояния *входной связи* и одной определяющей состояние *выходной связи*)	Процесс изменяет состояние объекта из любого исходного состояния в заданное выходное состояние	 Процесс **Cleaning & Painting** (Очистка & Окраска) изменяет состояние объекта **Engine Hood** (Капот двигателя) на состояние **painted** (окрашенный)	Объект типа *affectee* Воздействующий процесс	Воздействующий процесс Состояние получателя объекта типа *affectee*

9.4 Определяющие состояние разрешающие связи

9.4.1 Определяющая состояние агентская связь

Определяющей состояние агентской связью (agent link) должна быть агентская связь между определенным состоянием агента и процессом, причем этому агенту необходим процесс активации и выполнения.

Графически линия с заполненным кружком на конце, напоминающим «черный леденец на палочке», проходящая от заданного состояния объекта-агента до процесса, его разрешающего, должна обозначать определяющую состояние агентскую связь.

Синтаксис определяющей состояние агентской связи должен выражаться следующим OPL-предложением: Объект **Specified-state Agent** (Определяющий состояние агент) управляет процессом **Processing** (Обработка).

Примечание — Обозначения имен состояния не имеют начальных заглавных букв, за исключением тех случаев, когда они находятся в начале OPL-предложения.

Пример — Необходимо, чтобы объект Pilot (Пилот) был в состоянии sober (адекватный), чтобы его можно было бы считать агентом процесса Flying (Полет) на объекте Airplane (Самолет). Соответствующее OPL-предложение таково: Объект Sober Pilot (Адекватный пилот) управляет процессом Flying (Полет).

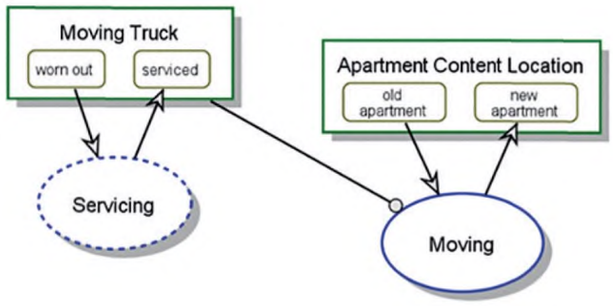
9.4.2 Определяющая состояние инструментальная связь

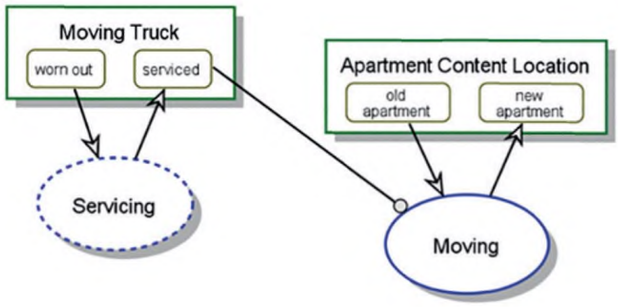
Определяющей состояние инструментальной связью (instrument link) должна быть инструментальная связь между указанным состоянием инструментальных средств и процессом, причем инструментальным средствам понадобится процесс активации и выполнения.

Графически линия с незаполненным кружочком на конце, напоминающим «белый леденец на палочке», которая проходит от заданного состояния объекта-агента до процесса, позволяет обозначать определяющую состояние инструментальную связь.

Синтаксис определяющей состояние инструментальной связи должен выражаться следующим OPL-предложением: Процесс **Processing** (Обработка) требует объекта **Specified-state Instrument** (Определяющий состояние инструмент).

Пример — OPD-диаграмма (см. рисунок 12) иллюстрирует различия между основной связью и определяющими состояние инструментальными связями. В левой части рисунка объект Moving Truck (Подвижная тележка) является инструментальным средством для процесса Moving (Перемещение) и означает, что состояние этого объекта не имеет значения, тогда как в правой части данного рисунка квалифицированное состояние serviced (обслуживаемое) Moving Truck является инструментальным средством для процесса Moving, которое означает, что процесс Moving будет иметь место только тогда, когда объект Moving Truck будет находиться в состоянии serviced.





Физический объект **Moving Truck** (Подвижная тележка) является объектом в состоянии **physical** (физическое).

Физический объект **Moving Truck** (Подвижная тележка) может находиться в состоянии **worn out** (изношенный) или **serviced** (обслуживаемый).

Связанный с внешней средой процесс **Servicing** (Обслуживание) является процессом **environmental** (связанный с внешней средой) и состоянием **physical** (физическое).

Процесс **Servicing** (Обслуживание) изменяет состояние физического объекта **Moving Truck** (Подвижная тележка) из состояния **worn out** (изношенное) на состояние **serviced** (обслуживаемое).

Физический объект **Apartment Content Location** (Положение содержимого помещения) является объектом с состоянием **physical** (физическое).

Физический объект **Apartment Content Location** (Положение содержимого помещения) может находиться в состоянии **old apartment** (старое помещение) или в состоянии **new apartment** (новое помещение).

Физический процесс **Moving** (Перемещение) является процессом в состоянии **physical** (физическое).

Физический процесс **Moving** (Перемещение) требует объекта **Moving Truck** (Подвижная тележка).

Физический процесс **Moving** (Перемещение) изменяет состояние физического объекта **Apartment Content Location** (Положение содержимого помещения) из состояния **old apartment** (старое помещение) на состояние **new apartment** (новое помещение).

Физический объект **Moving Truck** (Подвижная тележка) является объектом в состоянии **physical** (физическое).

Физический объект **Moving Truck** (Подвижная тележка) может находиться в состоянии **worn out** (изношенный) или в состоянии **serviced** (обслуживаемый).

Связанный с внешней средой процесс **Servicing** (Обслуживание) является процессом **environmental** (связанный с внешней средой) и состоянием **physical** (физический).

Процесс **Servicing** (Обслуживание) изменяет состояние физического объекта **Moving Truck** (Подвижная тележка) из состояния **worn out** (изношенное) на состояние **serviced** (обслуживаемое).

Физический объект **Apartment Content Location** (Положение содержимого помещения) является объектом в состоянии **physical** (физическое).

Физический объект **Apartment Content Location** (Положение содержимого помещения) может находиться в состоянии **old apartment** (старое помещение) или в состоянии **new apartment** (новое помещение).

Физический процесс **Moving** (Перемещение) является процессом в состоянии **physical** (физическое).

Физический процесс **Moving** (Перемещение) требует объекта **Moving Truck** (Подвижная тележка) в состоянии **serviced** (обслуживаемый).

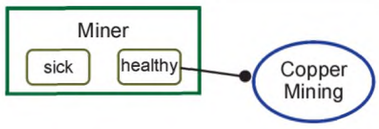
Физический процесс **Moving** (Перемещение) изменяет состояние физического объекта **Apartment Content Location** (Положение содержимого помещения) из состояния **old apartment** (старое помещение) на состояние **new apartment** (новое помещение).

Рисунок 12 — Инструментальная связь (слева) в сравнении с определяющей состояние инструментальной связью (справа)

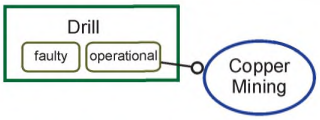
9.4.3 Обзор определяющих состояние разрешающих связей

В таблице 4 приведены все определяющие состояние разрешающие связи (enabling links).

Таблица 4 — Обзор определяющих состояние разрешающих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Определяющая состояние агентская связь	Агент-человек дает возможность процессу находиться в заданном состоянии	 Состояние **healthy** (здоровый) объекта **Miner** (Горняк) управляет процессом **Copper Mining** (Добыча меди)	Состояние агента	Разрешенный процесс

Окончание таблицы 4

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Определяющая состояние инструментальная связь	Процесс требует инструментального средства, находящегося в заданном состоянии	 Процесс **Copper Mining** (Добыча меди) требует от объекта **Drill** (Бурение) состояния **operational** (рабочее)	Состояние инструмента	Разрешенный процесс

9.5 Управляющие связи

9.5.1 Виды управляющих связей

В рамках парадигмы Event-Condition-Action (событие-условие-действие) (см. 8.2.1), лежащей в основе операционной OPM-семантики, ситуационная (event link), условная (condition link) и исключающая связи (exception link) должны характеризовать соответственно событие, условие и временное исключение. Эти три вида связей должны представлять управляющие связи (control links), которые должны существовать либо между объектом и процессом, либо между двумя процессами.

Ситуационная связь должна определять исходное событие и целевой процесс для активации наступления события, которое будет вызывать оценку предварительных условий процесса для их удовлетворения.

Удовлетворение предварительных условий позволяет выполнять процесс и делать процесс активным. Если предварительные условия для процесса не выполняются, то процесс не должен протекать. При этом вне зависимости от того, была оценка успешной или нет, данное событие должно пропускаться.

Если предварительные условия для процесса не выполняются, то процесс активации не будет происходить до тех пор, пока этот процесс не будет активирован другим событием. Управляющие связи определяют, будет процесс ожидать другого активирующего события или поток сигналов контроля исполнением обойдет данный процесс.

Примечание 1 — Для инициализации оценки предварительных условий все последующие события могут поступать из других источников.

Условная связь должна быть процедурной связью (procedural link) между исходным объектом-источником (или объектом-состоянием) и целевым процессом. Эта условная связь должна устанавливать способ обхода, который будет позволять системе управлять выполнением путем пропуска или обхода целевого процесса, если оценка выполнения его предварительных условий оказалась неудовлетворительной.

Примечание 2 — Без способа обхода в условной связи невозможность удовлетворения предварительных условий ограничивает процесс ожидания выполнения этих условий.

Для ситуационных/условных связей каждый вид входящих преобразующей и разрешающей связей, т. е. связи между объектом (или состоянием объекта) и процессом, должен иметь соответствующий вид ситуационной/условной связи.

Исключающей связью должна быть процедурная связь между процессом, который по какой-то причине невозможно успешно завершить или он занимает большее или меньшее время для завершения, чем ранее ожидалось, и процессом, который должен управлять ситуацией исключения.

Примечание 3 — Из-за невозможности успешного завершения процесса часто происходит увеличение или снижение сроков исполнения, исключительными могут стать другие ситуации. Кроме того, все несвязанные со временем исключения могут быть смоделированы с использованием диапазонов значений (подобный способ использования рассмотрен в разделе С.6).

Графически управляющий модификатор, который может представляться в виде аннотации рядом с входящей преобразующей или разрешающей связью, т. е. связью между объектом (или состоянием объекта) и процессом, будет обозначать соответствующую управляющую связь. Символ аннотации «е», означающий событие, должен обозначать ситуационную связь, и символ аннотации «с», обозначающий состояние, должен обозначать условную связь. Аннотация для управляющего модификатора для исключающей связи изображается в виде одного или двух коротких штрихов, пересекающих связь рядом с процессом управления исключениями.

9.5.2 Ситуационные связи

9.5.2.1 Преобразующие ситуационные связи

9.5.2.1.1 Потребительская ситуационная связь

Потребительской ситуационной связью (consumption event link) должна быть аннотированная потребительская связь между объектом и процессом, который инициируется операционным экземпляром объекта. Удовлетворение предварительных условий для процесса и последующего выполнения процесса должно приводить к потреблению экземпляра инициирующего объекта.

Графически стрелка, направленная от объекта к процессу с маленькой аннотирующей буквой «e» возле наконечника стрелки, относящейся к событию, должна обозначать потребительскую ситуационную связь.

Синтаксис определяющей состояние эффективной ситуационной связи должен выражаться следующим OPL-предложением: Объект **Object** (Объект) инициирует процесс **Process** (Процесс), который потребляет объект **Object**.

9.5.2.1.2 Воздействующая ситуационная связь

Воздействующей ситуационной связью (effect event link) должна быть аннотированная часть действующей связи между объектом и процессом, который инициирует создание оперативного экземпляра объекта. Удовлетворение предварительных условий для процесса и последующего его выполнения должно каким-либо образом оказывать влияние на инициирующий объект.

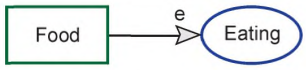
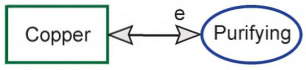
Графически двунаправленная стрелка между объектом и процессом с маленькой аннотирующей буквой «e» вблизи конца стрелки у процесса, обозначающей событие, должна обозначать действующую ситуационную связь.

Синтаксис определяющей состояние действующей ситуационной связи должен выражаться следующим OPL-предложением: Объект **Object** (Объект) инициирует процесс **Process** (Процесс), который взаимодействует с объектом **Object**.

9.5.2.1.3 Обзор преобразующих ситуационных связей

В таблице 5 приведены все преобразующие ситуационные связи.

Таблица 5 — Обзор преобразующих ситуационных связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Потребительская ситуационная связь	Объект инициирует процесс, который при его выполнении будет потреблять объект	 Объект **Food** (Пища) инициирует процесс **Eating** (Прием пищи), который будет потреблять объект **Food**	Инициализирующий объект типа *consumee*	Инициируемый процесс, который потребляет инициализирующий объект типа *consumee*
Воздействующая ситуационная связь	Объект инициирует процесс, который при его выполнении будет взаимодействовать с объектом	 Объект **Copper** (Медь) инициирует процесс **Purifying** (Рафинирование), который будет взаимодействовать с объектом **Copper**	Инициализирующий объект типа *affectee*	Инициируемый процесс, который воздействует на инициализирующий объект типа *affectee*
Примечание — Ситуационная связь — это связь между объектом и процессом, однако связь между процессом и объектом не является ситуационной связью.				

9.5.2.2 Разрешающие ситуационные связи

9.5.2.2.1 Агентская ситуационная связь

Агентской ситуационной связью (agent event link) должна быть аннотированная разрешающая связь (annotated enabling link) между объектом-агентом и процессом, который инициирует объект и обеспечивает действие.

Графически линия с зачерненным кружком на конце, напоминающим «черный леденец на палочке», проходящая от объекта-агента к процессу, который он инициировал и обеспечил его действие

(с маленькой аннотирующей буквой «e» вблизи конца линии около процесса, означающего событие), должна обозначать агентскую ситуационную связь (agent event link).

Синтаксис агентской ситуационной связи должен выражаться следующим OPL-предложением: Объект **Agent** (Агент) инициирует и управляет процессом **Process** (Процесс).

9.5.2.2.2 Инструментальная ситуационная связь

Инструментальной ситуационной связью (instrument event link) должна быть аннотированная разрешающая связь между инструментальным объектом и процессом, который объект инициировал и обеспечил его действие.

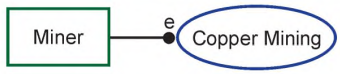
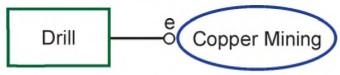
Графически линия с незаполненным кружком на конце, напоминающим «белый леденец на палочке», проходящая от инструментального объекта к процессу, который объект инициировал и обеспечил его действие (с маленькой аннотирующей буквой «e» ближе к концу линии около процесса, означающего событие), должна обозначать инструментальную ситуационную связь.

Синтаксис инструментальной ситуационной связи должен выражаться следующим OPL-предложением: Объект **Instrument** (Инструментальное средство) инициирует процесс **Process** (Процесс), который требует объекта **Instrument** (Инструментальное средство).

9.5.2.2.3 Обзор разрешающих ситуационных связей

В таблице 6 приведены все разрешающие ситуационные связи.

Таблица 6 — Обзор разрешающих ситуационных связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Агентская ситуационная связь	Агент-человек инициирует и позволяет действовать процессу. Агент должен существовать на протяжении всего процесса	 Объект **Miner** (Горняк) инициирует и управляет процессом **Copper Mining** (Добыча меди)	Инициализирующий агент	Инициализированный процесс
Инструментальная ситуационная связь	Объект инициирует процесс как инструментальное средство, поэтому он не изменяется, однако он необходим на протяжении всего процесса	 Объект **Drill** (Бурение) инициирует процесс **Copper Mining** (Добыча меди), который требует объекта **Drill**	Инициализирующее инструментальное средство	Инициализированный процесс

9.5.2.3 Определяющие состояние преобразующие ситуационные связи

9.5.2.3.1 Определяющая состояние потребительская ситуационная связь

Определяющей состояние потребительской ситуационной связью (state-specified consumption event link) должна быть аннотированная потребительская связь между процессом и заданным состоянием объекта, чей операционный экземпляр инициирует этот процесс. Удовлетворение предварительных условий для процесса, в том числе инициирующего объекта в указанном состоянии, и последующее выполнение этого процесса должны приводить к потреблению инициирующего объекта.

Графически стрелка, направленная от заданного состояния объекта к процессу, с маленькой аннотирующей буквой «e» (означающей событие) возле наконечника, должна обозначать определяющую состояние потребительскую ситуационную связь.

Синтаксис определяющей состояние потребительской ситуационной связи должен выражаться следующим OPL-предложением: Объект **Specified-state Object** (Объект в определенном состоянии) инициирует процесс **Process** (Процесс), который будет потреблять объект **Object** (Объект).

9.5.2.3.2 Определяющая вход/выход воздействующая ситуационная связь

Определяющей вход/выход воздействующей ситуационной связью (input-output-specified effect event link) должна быть аннотированная определяющая вход/выход воздействующая связь, которая будет инициировать воздействующий процесс, когда операционный экземпляр объекта войдет в заданное состояние исходного источника.

Графически определяющая вход/выход воздействующая ситуационная связь, изображаемая стрелкой с маленькой аннотирующей буквой «e» (обозначающей событие) вблизи наконечника стрелки (рядом с входной связью), должна означать этот вид связи.

Синтаксис определяющей вход/выход действующей ситуационной связи должен выражаться следующим OPL-предложением: Объект **Input-state Object** (Объект в исходном состоянии) инициирует процесс **Process** (Процесс), который изменяет состояние объекта **Object** (Объект) из состояния **input-state** (исходное состояние) на состояние **output-state** (конечное состояние).

9.5.2.3.3 Определяющая вход воздействующая ситуационная связь

Определяющей вход воздействующей ситуационной связью (input-specified effect event link) должна быть аннотированная определяющая вход воздействующая связь, которая будет инициировать воздействующий процесс, когда операционный экземпляр объекта войдет в заданное состояние исходного источника.

Графически определяющая вход воздействующая ситуационная связь, изображаемая стрелкой с маленькой аннотирующей буквой «e» (обозначающей событие) вблизи наконечника стрелки (рядом с входной связью), должна означать этот вид связи.

Синтаксис определяющей вход воздействующей ситуационной связи должен выражаться следующим OPL-предложением: Объект **Input-state Object** (Объект в исходном состоянии) инициирует процесс **Process** (Процесс), который изменяет состояние объекта **Object** (Объект) из состояния **input-state** (исходное состояние).

9.5.2.3.4 Определяющая выход воздействующая ситуационная связь

Определяющей выход воздействующей ситуационной связью (output-specified effect event link) должна быть аннотированная определяющая выход воздействующая связь, которая будет инициировать воздействующий процесс, когда возникнет операционный экземпляр объекта.

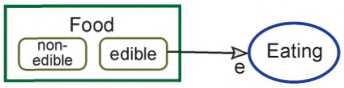
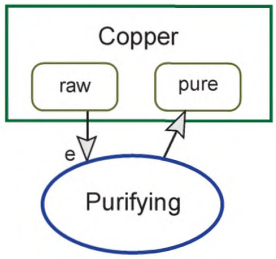
Графически определяющая выход воздействующая ситуационная связь, изображаемая стрелкой с маленькой аннотирующей буквой «e» (обозначающей событие) вблизи наконечника стрелки (рядом с входной связью), должна означать этот вид связи.

Синтаксис определяющей выход воздействующей ситуационной связи должен выражаться следующим OPL-предложением: Объект **Object** (Объект) в любом состоянии инициирует процесс **Process** (Процесс), который изменяет состояние объекта **Object** на состояние **destination-state** (целевое состояние).

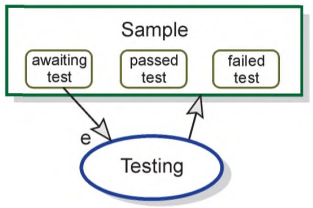
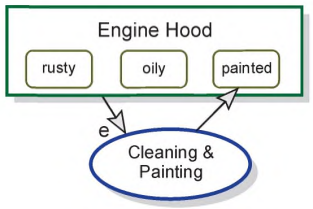
9.5.2.3.5 Обзор определяющих состояние преобразующих ситуационных связей

В таблице 7 приведены все определяющие состояние преобразующие ситуационные связи.

Таблица 7 — Обзор определяющих состояние преобразующих ситуационных связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Определяющая состояние потребительская ситуационная связь	Объект в заданном состоянии инициирует процесс и потребляется им	 Состояние **edible** (съедобное) объекта **Food** (Пища) инициирует процесс **Eating** (Прием пищи), который потребляет объект **Food**	Состояние объекта типа *consumee*	Инициализированный процесс
Определяющая вход/выход действующая ситуационная связь	Объект в заданном состоянии инициирует процесс и преобразуется им в конечное состояние	 Состояние **raw** (сырье) объекта **Copper** (Медь) инициирует процесс **Purifying** (Рафинирование), который изменяет состояние объекта **Copper** с состояния **raw** (сырье) на состояние **pure** (очищенное)	Состояние источника объекта типа *affectee*	Инициализированный процесс
			Инициализированный процесс	Целевое состояние объекта типа *affectee*

Окончание таблицы 7

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Пара определяющих вход действующих ситуационных связей	Объект в заданном состоянии инициирует процесс и преобразуется им в любое состояние	 Состояние **awaiting test** (ожидание испытаний) физического объекта **Sample** (Проба) инициирует физический процесс **Testing** (Испытание), который изменяет состояние объекта **Sample** с состояния **awaiting test**	Состояние источника объекта типа *affectee* Инициализированный процесс	Инициализированный процесс Объект типа *affectee*
Пара определяющих выход воздействующих ситуационных связей	Объект (в любом из его состояний) инициирует процесс и преобразуется им до выходного состояния	 Физический объект **Engine Hood** (Капот двигателя) инициирует физический процесс **Cleaning & Painting** (Очистка & Окраска), который изменяет состояние объекта **Engine Hood** на состояние **painted** (окрашенный)	Объект типа *affectee* Инициализированный процесс	Инициализированный процесс Целевое состояние объекта *affectee*

9.5.2.4 Определяющие состояние разрешающие ситуационные связи

9.5.2.4.1 Определяющая состояние агентская ситуационная связь

Определяющей состояние агентской ситуационной связью (state-specified agent event link) должна быть аннотированная определяющая состояние агентская связь (annotated state-specified agent link), которая будет инициализировать процесс, когда операционный экземпляр агента будет переходить в заданное состояние.

Графически определяющая состояние агентская ситуационная связь в виде линии с маленькой аннотирующей буквой «e» (означающей событие) на конце линии вблизи процесса должна обозначать эту связь.

Синтаксис определяющей состояние агентской ситуационной связи должен выражаться следующим OPL-предложением: Объект **Specified-state Agent** (Определяющий состояние агент) инициирует и управляет процессом **Processing** (Обработка).

9.5.2.4.2 Определяющая состояние инструментальная ситуационная связь

Определяющей состояние инструментальной ситуационной связью (state-specified instrument event link) должна быть аннотированная определяющая состояние инструментальная связь, которая инициирует процесс, когда операционный экземпляр инструмента переходит в заданное состояние.

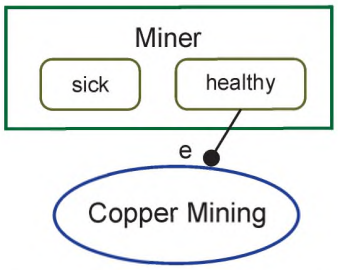
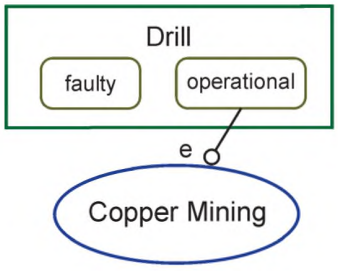
Графически определяющая состояние инструментальная ситуационная связь в виде линии с маленькой аннотирующей буквой «e» (означающей событие) на конце линии вблизи процесса, должна обозначать эту связь.

Синтаксис определяющей состояние инструментальной ситуационной связи должен выражаться следующим OPL-предложением: Объект **Specified-state Instrument** (Определяющее состояние инструментальное средство) инициирует процесс **Processing** (Обработка), который требует объект **Specified-state Instrument**.

9.5.2.4.3 Обзор определяющих состояние разрешающих ситуационных связей

В таблице 8 приведены все определяющие состояние разрешающие ситуационные связи (enabling event links).

Таблица 8 — Обзор определяющих состояние разрешающих ситуационных связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Определяющая состояние агентская ситуационная связь	Агент-человек в определенном состоянии инициирует процесс и действует как его агент. Агент должен находиться в определенном состоянии на протяжении всего процесса	 Состояние **healthy** (здоровый) объекта **Miner** (Горняк) инициирует и управляет процессом **Copper Mining** (Добыча меди)	Состояние объекта	Инициализированный процесс
Определяющая состояние инструментальная ситуационная связь	Объект в определенном состоянии инициирует процесс и является инструментальным средством его реализации. Инструментальному средству необходимо находиться в определенном состоянии на протяжении всего процесса	 Состояние **operational** (рабочее) объекта **Drill** (Бурение) инициирует процесс **Copper Mining** (Добыча меди), который требует состояния **operational** объекта **Drill**	Состояние инструмента	Инициализированный процесс

9.5.2.5 Инициализирующие связи

9.5.2.5.1 Процессуальная инициализация и инициализирующая связь

Процессуальной инициализацией (process invocation) должно быть событие, с помощью которого один процесс будет инициализировать другой процесс. Инициализирующей связью (invocation link) должна быть связь между процессом-источником и вызываемым им целевым процессом (процессом-получателем); последнее означает, что после завершения процесса-источника он немедленно будет инициализировать целевой процесс-получатель на другом конце инициализирующей связи.

Примечание 1 — Стандартный или ожидаемый поток сигналов контроля исполнения не будет инициализировать новый процесс, пока предыдущий процесс не будет успешно завершен. Разработчик модели должен сам заботиться о прерывании любого процесса. В разделе С.6 описано несколько способов управления прерыванием процесса из-за возникновения какого-либо сбоя (в особенности см. С.6.8).

Примечание 2 — Поскольку OPM-процесс выполняет преобразование, инициализирующая связь семантически подразумевает создание временного объекта (с помощью процесса, инициируемого источником), причем каждый последующий инициализированный процесс будет немедленно потребляться. В OPM-модели инициализирующая связь может заменять соответствующий кратковременный физический или информационный объект [например, объект **Record ID** (Идентификатор записи) в запросе], который создает исходный процесс для инициализации целевого процесса и сразу же будет потреблять временный объект.

Графически символ изломанной линии в виде молнии, исходящий от инициализирующего процесса к инициализированному (целевому) процессу и заканчивающемуся на нем, должен обозначать инициализирующую связь.

Синтаксис инициализирующей связи должен выражаться следующим OPL-предложением: Процесс **Invoking-process** (Инициализирующий процесс) инициирует процесс **Invoked-process** (Инициализированный процесс).

9.5.2.5.2 Самоинициализирующаяся связь

Самоинициализирующейся связью (self-invocation link) должна быть связь, сама по себе инициализирующая какой-либо процесс, причем после завершения этого процесса он сразу же будет инициализировать самого себя. Эта связь должна определять самоинициализацию.

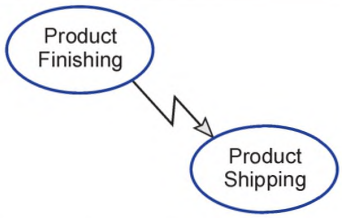
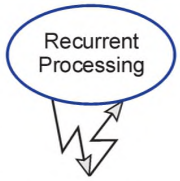
Графически пара самоинициализирующихся связей, возникающих в процессе и присоединяющих «голову процесса к его хвосту» перед возвращением к первоначальному процессу, должна обозначать эту связь.

Синтаксис самоинициализирующейся связи должен выражаться следующим OPL-предложением: Процесс **Invoking-process** (Инициализирующий процесс) инициирует самого себя.

9.5.2.5.3 Обзор инициализирующих связей

В таблице 9 приведены все инициализирующие связи.

Таблица 9 — Обзор инициализирующих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Инициализирующая связь	Сразу же после завершения инициализи-рованного процесса он будет инициализировать процесс, определяемый связью данного типа	 Процесс **Product Finishing** (Отделка изделий) инициирует процесс **Product Shipping** (Отправка изделий)	Инициализирующий процесс	Другой инициализированный процесс
Самоинициализирующаяся связь	Сразу же после завершения процесса он немедленно будет инициализировать самого себя	 Процесс **Recurrent Processing** (Рекуррентная обработка) инициирует самого себя	Инициализированный процесс	Тот же самый процесс

9.5.3 Условные связи

9.5.3.1 Основные условные преобразующие связи

9.5.3.1.1 Условная потребительская связь

Условной потребительской связью (condition consumption link) должна быть аннотированная потребительская связь (annotated consumption link) между объектом типа *consumee* и процессом. Если операционный экземпляр объекта типа *consumee* существует в момент инициализации процесса каким-либо событием, то наличие этого операционного экземпляра объекта типа *consumee* будет удовлетворять предварительным условиям для данного процесса (в отношении этого объекта). Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительно установленным условиям, то процесс запускается и потребляет этот экземпляр объекта типа *consumee*, однако если операционный экземпляр объекта типа *consumee* не существует в момент инициализации процесса событием, то процесс оценки предварительных условий потерпит неудачу (не выполнится), а поток сигналов управления исполнением будет обходить или пропускать процесс без определения его показателей.

Графически, стрелка, направленная от объекта типа *consumee* к процессу с маленькой аннотирующей буквой «с» (обозначающей состояние) около наконечника стрелки, должна обозначать условную потребительскую связь.

Синтаксис условной потребительской связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) выполняется тогда, когда существует объект **Object** (Объект); в этом случае объект **Object** потребляется; в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной потребительской связи должен выглядеть в виде следующего OPL-предложения: Если объект **Object** (Объект) существует, то процесс **Process** (Процесс) начинает выполняться и поглощает объект **Object**; в противном случае процесс **Process** обходится.

Примечание — Для получения подробных сведений относительно семантики понятия «пропуск» (skip) см. 14.2.2.4.2 и рисунок С.25.

9.5.3.1.2 Условная воздействующая связь

Условной воздействующей связью (condition effect link) должна быть аннотированная результирующая связь между объектом *affectee* и процессом. Если операционный экземпляр объекта типа *affectee* существует в момент инициализации процесса каким-либо событием, то наличие экземпляра объекта типа *affectee* будет удовлетворять предварительным условиям процесса в отношении данного объекта. Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительным условиям, то процесс будет запускаться и воздействовать на экземпляр объекта типа *affectee*, однако если операционный экземпляр объекта типа *affectee* не будет существовать в момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий потерпит неудачу (не выполнится), а поток сигналов управления исполнением будет обходить или пропускать процесс без определения его показателей.

Графически двунаправленная стрелка, указывающая одним концом на объект типа *affectee*, а другим — на воздействующий процесс, с маленькой аннотирующей буквой «с» (обозначающей состояние) около конца стрелки у процесса, должна обозначать условную воздействующую связь.

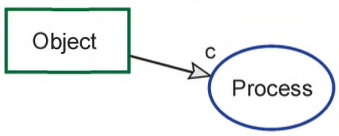
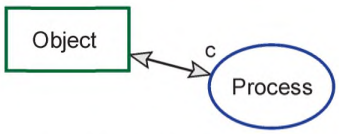
Синтаксис условной воздействующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) выполняется тогда, когда существует объект **Object** (Объект); в этом случае процесс **Process** будет воздействовать на объект **Object**; в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной воздействующей связи должен выглядеть в виде следующего OPL-предложения: Если объект **Object** (Объект) существует, то процесс **Process** (Процесс) возникает и воздействует на объект **Object**; в противном случае процесс **Process** обходится.

9.5.3.1.3 Обзор условных преобразующих связей

В таблице 10 приведены все условные преобразующие связи (condition transforming links).

Таблица 10 — Обзор условных преобразующих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Условная потребительская связь	Если операционный экземпляр объекта существует, а остальные предварительные условия удовлетворены, то процесс будет выполняться и потреблять экземпляр объекта; в противном случае сигналы контроля исполнением будут инициализировать последующий процесс	 Процесс **Process** (Процесс) будет протекать, если объект **Object** (Объект) существует; в этом случае процесс **Process** потребляет объект **Object**; в противном случае процесс **Process** пропускается	Объект с устанавливаемыми условиями	Условный процесс
Условная воздействующая связь	Если операционный экземпляр объекта существует, а остальные предварительные условия удовлетворены, то процесс будет выполняться и воздействовать на экземпляр объекта; в противном случае сигналы контроля исполнением будут инициализировать следующий процесс	 Процесс **Process** (Процесс) будет протекать, если объект **Object** (Объект) существует; в этом случае процесс **Process** будет взаимодействовать с объектом **Object**; в противном случае процесс **Process** пропускается	Объект с устанавливаемыми условиями	Условный процесс

9.5.3.2 Основные условные разрешающие связи

9.5.3.2.1 Условная агентская связь

Условной агентской связью (condition agent link) должна быть аннотированная агентская связь (annotated agent link) между агентом и процессом. Если операционный экземпляр агента существует в момент инициализации процесса событием, то наличие этого экземпляра агента будет удовлетворять предварительным условиям процесса (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительно установленным условиям, то процесс начинается и агент отрабатывает свои характеристики, однако если операционный экземпляр агента не существует в момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий терпит неудачу (не выполняется), а поток сигналов управления исполнением обходит или пропускает процесс без его выполнения.

Графически линия с зачерненным кружком на конце, напоминающим «черный леденец на палочке», проходящая от объекта-агента к процессу, который он разрешает, с маленькой аннотирующей буквой «c» (означающей состояние) ближе к концу линии у процесса, должна обозначать условную агентскую связь.

Синтаксис условной агентской связи должен выражаться следующим OPL-предложением: Объект **Agent** (Агент) управляет процессом **Process** (Процесс), если объект **Agent** существует; в противном случае этот процесс **Process** пропускается.

Альтернативный синтаксис условной агентской связи должен выглядеть в виде следующего OPL-предложения: Если объект **Agent** (Агент) существует, то объект **Agent** управляет процессом **Process** (Процесс); в противном случае этот процесс **Process** обходится.

9.5.3.2.2 Условная инструментальная связь

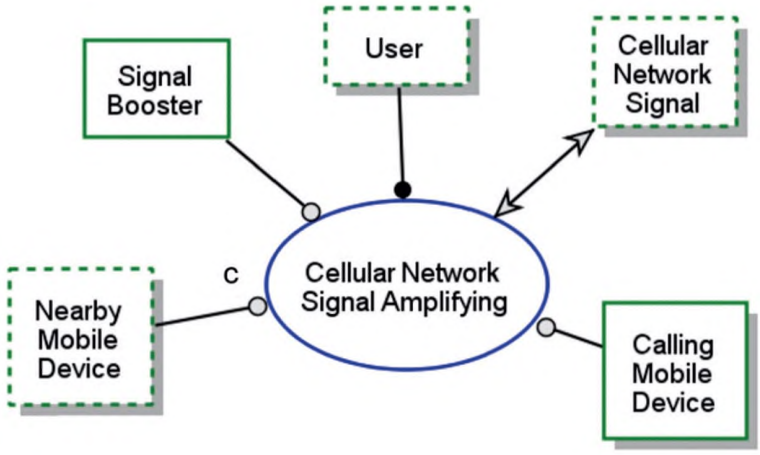
Условной инструментальной связью (condition instrument link) должна быть аннотированная инструментальная связь (annotated instrument link) между инструментальным средством и процессом. Если операционный экземпляр инструмента существует в момент инициализации процесса каким-либо событием, то наличие этого экземпляра инструмента будет удовлетворять предварительным условиям процесса (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительно установленным условиям, то процесс начинается, однако если операционный экземпляр инструмента не существует в момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий терпит неудачу (не выполняется), а поток сигналов управления исполнением обходит или пропускает процесс без его выполнения.

Графически линия с незаполненным кружком на конце, похожим на «белый леденец на палочке», проходящая от инструментального объекта до процесса, с маленькой аннотирующей буквой «c» (означающей состояние) ближе к концу линии у процесса, должна обозначать условную инструментальную связь.

Синтаксис условной инструментальной связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) выполняется, если объект **Instrument** (Инструментальное средство) существует; в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной агентской связи должен выглядеть в виде следующего OPL-предложения: Если объект **Instrument** (Инструментальное средство) существует, то процесс **Process** (Процесс) существует; в противном случае процесс **Process** обходится.

Пример — На рисунке 13 представлена OPD-диаграмма с условной инструментальной связью между связанным с внешней средой объектом Nearby Mobile Device (Ближайшее устройство мобильной связи) и процессом Cellular Network Signal Amplifying (Усилитель сигнала сотовой связи), которая возникает, только если существует объект Nearby Mobile Device; в противном случае процесс будет пропускаться из-за отсутствия на близлежащих пунктах усилительных устройств.



Процесс **Cellular Network Signal Amplifying** (Усилитель сигнала сотовой связи) протекает, если существует объект **Nearby Mobile Device** (Ближайшее устройство мобильной связи); в противном случае объект **Cellular Network Signal Amplifying** пропускается.

Signal Booster — Усилитель сигнала.

Cellular Network Signal — Сигнал мобильной связи.

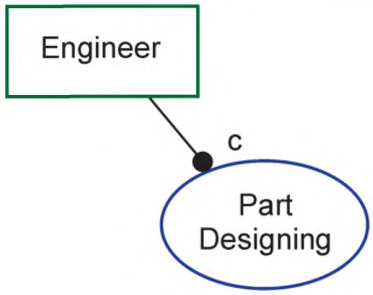
Cabling Mobile Device — Кабельное мобильное устройство.

Рисунок 13 — Диаграмма условной инструментальной связи (с частичным OPL-описанием)

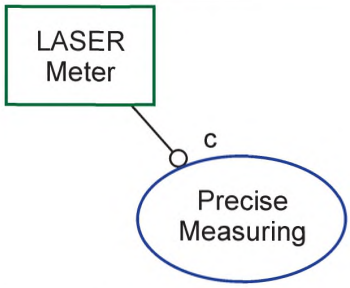
9.5.3.2.3 Обзор основных условных разрешающих связей

В таблице 11 приведены основные условные разрешающие связи (basic condition enabling links).

Таблица 11 — Обзор основных условных разрешающих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Агентская условная связь	Агент обеспечивает выполнение процесса, если этот агент существует; в противном случае этот процесс пропускается	 Объект **Engineer** (Инженер) управляет процессом **Part Designing** (Проектирование деталей), если объект **Engineer** существует; в противном случае процесс **Part Designing** пропускается	Объект с устанавливаемыми условиями	Условный процесс

Окончание таблицы 11

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Инструментальная условная связь	Инструментальное средство позволяет выполнять процесс, если инструментальное средство существует; в противном случае данный процесс пропускается	 Процесс **Precise Measuring** (Точные измерения) осуществляется при существовании объекта **LASER Meter** (Лазерный измеритель); в противном случае процесс **Precise Measuring** пропускается	Инструмент с устанавливаемыми условиями	Условный процесс

9.5.3.3 Условные устанавливающие состояние преобразующие связи

9.5.3.3.1 Условная устанавливающая состояние потребительская связь

Условной устанавливающей состояние потребительской связью (condition state-specified consumption link) должна быть аннотированная условная потребительская связь (annotated condition consumption link) между заданным состоянием объекта типа *consumee* и процессом. Если операционный экземпляр объекта типа *consumee* в указанном состоянии существует в момент инициализации процесса каким-либо событием, то наличие экземпляра объекта *consumee* будет удовлетворять предварительным условиям процесса (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительным условиям, то процесс запускается и потребляет экземпляр объекта *consumee*, однако если операционный экземпляр объекта типа *consumee* в указанном состоянии не существует в момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий терпит неудачу (не выполняется), а поток сигналов управления исполнением обходит или пропускает процесс без его выполнения.

Графически стрелка, направленная от указанного состояния объекта *consumee* к процессу с маленькой аннотирующей буквой «c» (обозначающей состояние) вблизи наконечника стрелки, должна обозначать условную устанавливающую состояние потребительскую связь.

Синтаксис условной устанавливающей состояние потребительской связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) выполняется, если объект **Object** (Объект) имеет состояние **specified-state** (заданное состояние); в этом случае объект **Object** потребляется; в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной устанавливающей состояние потребительской связи должен выглядеть в виде следующего OPL-предложения: Если состояние **specified-state** (заданное состояние) объекта **Object** (Объект) существует, то процесс **Process** (Процесс) существует и потребляет объект **Object**; в противном случае процесс **Process** обходится.

9.5.3.3.2 Условная определяющая вход/выход воздействующая связь

Условной определяющей вход/выход воздействующей связью (condition input-output-specified effect link) должна быть аннотированная определяющая вход/выход воздействующая связь (annotated input-output-specified effect link) между исходным состоянием источника и процессом. Если операционный экземпляр объекта типа *affectee* в указанном состоянии существует в момент инициализации процесса каким-либо событием, то наличие этого экземпляра объекта типа *affectee* будет удовлетворять предварительным условиям процесса (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительным условиям, то процесс запускается и воздействует на этот операционный экземпляр объекта, изменяя состояние этого экземпляра из указанного входного состояния в указанное выходное состояние, однако если операционный экземпляр объекта типа *affectee* в указанном состоянии не существует в момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий терпит неудачу (не выполняется), а поток сигналов управления исполнением будет обходить или пропускать процесс без его выполнения.

Графически линия для условной определяющей вход/выход воздействующей связи с маленькой аннотирующей буквой «c» (обозначающей состояние) вблизи наконечника стрелы у входной связи должна обозначать эту связь.

Синтаксис условной определяющей вход/выход воздействующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) выполняется, если объект **Object** (Объект) находится в состоянии **input-state** (исходное состояние); в этом случае процесс **Process** изменяет состояние объекта **Object** из состояния **input-state** на состояние **output-state** (выходное состояние); в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной определяющей вход/выход действующей связи должен выглядеть в виде следующего OPL-предложения: Если состояние **input-state** (исходное состояние) объекта **Object** (Объект) существует, то процесс **Process** (Процесс) изменяет состояние объекта **Object** с состояния **input-state** на состояние **output-state** (выходное состояние); в противном случае процесс **Process** будет обходиться.

9.5.3.3.3 Условная определяющая вход воздействующая связь

Условной определяющей вход воздействующей связью (condition input-specified effect link) должна быть аннотированная определяющая вход воздействующая связь (annotated input-specified effect link) между исходным состоянием источника и процессом. Если операционный экземпляр объекта типа *affectee* в указанном состоянии существует в момент инициализации процесса каким-либо событием, то наличие экземпляра объекта типа *affectee* будет удовлетворять предварительным условиям этого процесса (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительным условиям, то процесс запускается и воздействует на этот экземпляр объекта, изменяя его состояние из указанного входного состояния в целевое состояние. Этим состоянием должно быть либо состояние по умолчанию, либо при его отсутствии — состояние распределения вероятности для объекта, которое должно определять выходное целевое состояние данного объекта (см. 12.7). Тем не менее если операционный экземпляр объекта типа *affectee* в указанном состоянии не существует в момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий терпит неудачу (не выполняется), а поток сигналов управления исполнением обходит или пропускает процесс без его выполнения.

Графически линия условной определяющей вход воздействующей связи с маленькой аннотирующей буквой «c» (обозначающей состояние) вблизи наконечника стрелы у входной связи должна обозначать эту связь.

Синтаксис условной определяющей вход воздействующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) будет протекать, если объект **Object** (Объект) находится в состоянии **input-state** (исходное состояние); в этом случае процесс **Process** изменяет состояние объекта **Object** из состояния **input-state**; в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной определяющей вход воздействующей связи должен выглядеть в виде следующего OPL-предложения: Если состояние **input-state** (исходное состояние) объекта **Object** (Объект) существует, то процесс **Process** (Процесс) изменяет состояние объекта **Object** из состояния **input-state**; в противном случае процесс **Process** обходится.

9.5.3.3.4 Условная определяющая выход воздействующая связь

Условной определяющей выход воздействующей связью (condition output-specified effect link) должна быть аннотированная определяющая выход воздействующая связь (annotated output-specified effect link) между исходным объектом и процессом. Если операционный экземпляр объекта типа *affectee* в указанном состоянии существует в момент инициализации процесса каким-либо событием, то наличие этого экземпляра объекта типа *affectee* будет удовлетворять предварительным условиям для этого процесса (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов удовлетворяет предварительным условиям, то процесс будет запускаться и воздействовать на этот экземпляр объекта, изменяя его состояние на определенное выходное состояние. Тем не менее если операционный экземпляр объекта типа *affectee* не существует в момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий терпит неудачу (не выполняется), а поток сигналов управления исполнением обходит или пропускает процесс без его выполнения.

Графически линия для условной определяющей выход воздействующей связи с указанием ее состояния с помощью маленькой аннотирующей буквы «c» (обозначающей состояние) вблизи наконечника стрелы у входной связи должна обозначать эту связь.

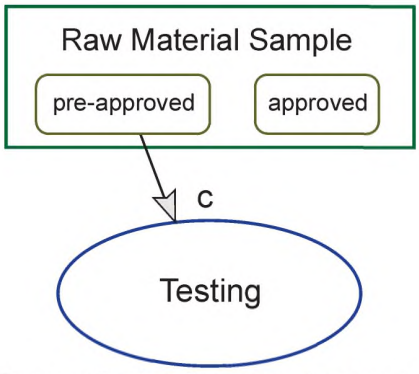
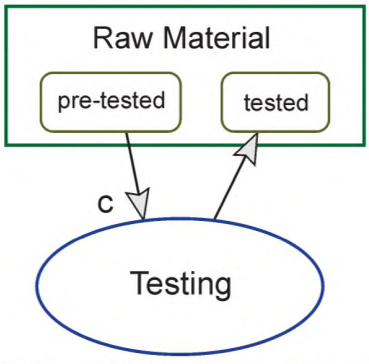
Синтаксис условной определяющей выход воздействующей связи должен выражаться следующим OPL-предложением: Процесс **Process** (Процесс) будет протекать, если объект **Object** (Объект) существует; в этом случае процесс **Process** будет изменять состояние объекта **Object** на состояние **output-state** (выходное состояние); в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной определяющей выход воздействующей связи должен выглядеть в виде следующего OPL-предложения: Если объект **Object** (Объект) существует, то процесс **Process** (Процесс) будет изменять состояние объекта **Object** на состояние **output-state** (выходное состояние); в противном случае процесс **Process** обходится.

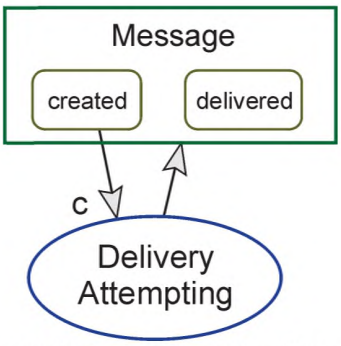
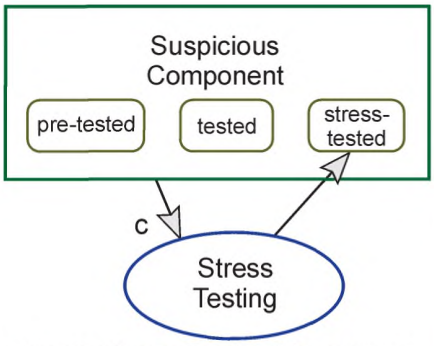
9.5.3.3.5 Обзор условных определяющих состояние преобразующих связей

В таблице 12 приведены все условные определяющие состояние преобразующие связи (condition state-specified transforming links).

Таблица 12 — Обзор условных определяющих состояние преобразующих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Условная определяющая состояние потребительская связь	Процесс будет выполняться, если объект находится в таком состоянии, из которого исходит связь, в противном случае этот процесс будет пропускаться	 Процесс **Testing** (Испытание) возникает, если объект **Raw Material Sample** (Проба необработанного материала) находится в состоянии **pre-approved** (заранее одобренный); в этом случае объект **Raw Material Sample** потребляется; в противном случае процесс **Testing** будет пропускаться	Конкретно-устанавливаемое состояние объекта	Условный процесс
Условная определяющая вход/выход воздействующая связь	Процесс будет выполняться, если объект находится в исходном состоянии (из которого исходит связь) и изменяет состояние объекта с его исходного состояния на его выходное состояние; в противном случае процесс будет пропускаться	 Процесс **Testing** (Испытание) происходит, если объект **Raw Material** (Необработанный материал) находится в состоянии **pre-tested** (предварительно испытанный); в этом случае процесс **Testing** изменяет состояние объекта **Raw Material** с состояния **pre-tested** на состояние **tested** (испытанный); в противном случае процесс **Testing** будет пропускаться	Конкретно-устанавливаемое исходное состояние объекта	Условный процесс

Окончание таблицы 12

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Условная определяющая вход воздействующая связь	Процесс будет выполняться, если объект находится в исходном состоянии (из которого исходит связь) и изменяет состояние объекта из его исходного состояния на его любое другое состояние; в противном случае процесс будет пропускаться	 Процесс **Delivery Attempting** (Попытка поставки) будет происходить, если объект **Message** (Сообщение) находится в состоянии **created** (созданное); в этом случае процесс **Delivery Attempting** будет изменять состояние объекта **Message** с состояния **created;** в противном случае процесс **Delivery Attempting** будет пропускаться	Конкретно-устанавливаемое исходное состояние объекта	Условный процесс
Условная определяющая выход воздействующая связь	Процесс будет выполняться, если объект существует и изменяет состояние объекта из его исходного состояния на его выходное состояние; в противном случае процесс будет пропускаться	 Процесс **Stress Testing** (Нагрузочное испытание) будет происходить, если существует объект **Suspicious Component** (Подозрительный компонент); в этом случае процесс **Stress Testing** будет изменять состояние объекта **Suspicious Component** на состояние **stress-tested** (испытанный на нагрузку); в противном случае процесс **Stress Testing** будет пропускаться	Объект с устанавливаемыми условиями	Условный процесс

9.5.3.4 Условные определяющие состояние разрешающие связи

9.5.3.4.1 Условная определяющая состояние агентская связь

Условной определяющей состояние агентской связью (condition state-specified agent link) должна быть аннотированная определяющая состояние агентская связь (annotated state-specified agent link) между заданным состоянием агента и процессом. Если операционный экземпляр агента в указанном состоянии будет существовать на момент инициализации процесса каким-либо событием, то наличие этого экземпляра агента должно удовлетворять предварительно установленным для этого процесса условиям (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов будет удовлетворять предварительным условиям, то процесс запустится, а сам агент будет заниматься выполнением операций, однако если операционный экземпляр агента в указанном состоянии не будет существовать на момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий потерпит неудачу (не выполнится), а поток сигналов управления исполнением обойдет или пропустит этот процесс без его выполнения.

Графически линия, означающая условную определяющую состояние агентскую связь, с маленькой аннотирующей буквой «с» (означающей состояние) ближе к концу линии у процесса, должна обозначать эту связь.

Синтаксис условной определяющей состояние агентской связи должен выражаться в виде следующего OPL-предложения: Объект **Agent** (Агент) будет заниматься выполнением процесса **Process** (Процесс), если объект **Agent** будет находиться в состоянии **specified-state** (заданное состояние); в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной определяющей состояние агентской связи должен выглядеть в виде следующего OPL-предложения: Если состояние **specified-state** (заданное состояние) объекта **Agent** (Агент) существует, то объект **Agent** будет заниматься выполнением процесса **Process** (Процесс); в противном случае процесс **Process** обходится.

9.5.3.4.2 Условная определяющая состояние инструментальная связь

Условной определяющей состояние инструментальной связью (condition state-specified instrument link) должна быть аннотированная определяющая состояние инструментальная связь (annotated state-specified instrument link) между заданным состоянием инструментального средства и процессом. Если операционный экземпляр инструментального средства в указанном состоянии будет существовать на момент инициализации процесса каким-либо событием, то наличие экземпляра инструментального средства должно удовлетворять предварительно установленным для этого процесса условиям (в отношении данного объекта). Если оценка всего набора предварительно обработанных объектов будет удовлетворять предварительным условиям, то процесс запустится, однако если операционный экземпляр инструментального средства в указанном состоянии не будет существовать на момент инициализации процесса каким-либо событием, то процесс оценки предварительных условий потерпит неудачу (не выполнится), а поток сигналов управления исполнением обойдет или пропустит этот процесс без его выполнения.

Графически линия, означающая условную определяющую состояние инструментальную связь, с маленькой аннотирующей буквой «с» (означающей состояние) ближе к концу линии у процесса, должна обозначать эту связь.

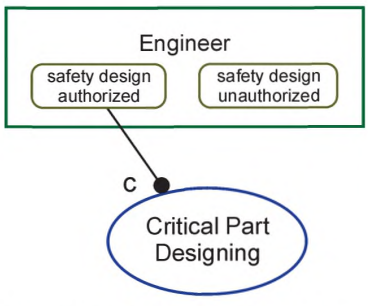
Синтаксис условной определяющей состояние инструментальной связи должен выражаться в виде следующего OPL-предложения: Процесс **Process** (Процесс) выполняется, если объект **Instrument** (Инструментальное средство) находится в состоянии **specified-state** (заданное состояние); в противном случае процесс **Process** пропускается.

Альтернативный синтаксис условной определяющей состояние инструментальной связи должен выглядеть в виде следующего OPL-предложения: Если объект **Instrument** (Инструментальное средство) находится в состоянии **specified-state** (заданное состояние), то процесс **Process** протекает; в противном случае процесс **Process** обходится.

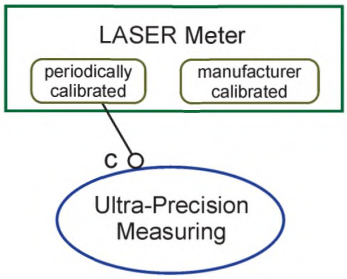
9.5.3.4.3 Обзор условных определяющих состояние разрешающих связей

В таблице 13 приведены все условные определяющие состояние разрешающие связи (condition state-specified enabling links).

Таблица 13 — Обзор условных определяющих состояние разрешающих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Условная определяющая состояние агентская связь	Агент допускает осуществление процесса, если первый находится в заданном состоянии; в противном случае этот процесс пропускается	 Объект **Engineer** (Инженер) занимается выполнением процесса **Critical**	Агент с устанавливаемыми условиями	Условный процесс

Окончание таблицы 13

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
		Part Designing (Разработка наименее долговечной детали), если объект **Engineer** находится в состоянии **safety design authorized** (уполномочен на проектирование с учетом требований безопасности); в противном случае процесс **Critical Part Designing** пропускается		
Условная определяющая состояние инструментальная связь	Инструмент допускает осуществление процесса, если первый находится в установленном состоянии; в противном случае этот процесс пропускается	 Процесс **Ultra-Precision Measuring** (Сверхточные измерения) осуществляется, если объект **LASER Meter** (Лазерный измеритель) находится в состоянии **periodically calibrated** (периодически откалиброван); в противном случае процесс **Ultra-Precision Measuring** пропускается	Инструмент с устанавливаемыми условиями	Условный процесс

9.5.4 Исключающие связи

9.5.4.1 Минимальная, ожидаемая и максимальная продолжительности процесса и их распределение

Любой процесс может иметь атрибут **Duration** (Продолжительность) со значением, выражаемым в единицах времени, причем атрибут **Duration** может определяться свойствами **Minimal Duration** (Минимальная продолжительность), **Expected Duration** (Ожидаемая продолжительность) и **Maximal Duration** (Максимальная продолжительность).

Свойства **Minimal Duration** (Минимальная продолжительность) и **Maximal Duration** (Максимальная продолжительность) должны характеризовать минимально и максимально допустимую продолжительности процесса до его завершения, а свойство **Expected Duration** (Ожидаемая продолжительность) должно быть статистическим средним значением продолжительности этого процесса.

Атрибут **Duration** (Продолжительность) может обладать дополнительным свойством **Duration Distribution** (Распределение продолжительностей) со значением, идентифицирующим связанные с этими продолжительностями именами и параметрами функции распределения вероятностей. Во время выполнения значение атрибута **Duration** определяется отдельно для каждого экземпляра процесса (т. е. при каждом появлении какого-либо процесса) путем выбора соответствующего значения из свойства **Duration Distribution**.

Примечание — Для обсуждения вопросов, связанных с продолжительностью процесса и системным временем выполнения, а также для рассмотрения примеров см. приложение D.

9.5.4.2 Связь, исключающая избыточную продолжительность процесса

Связь, исключающая избыточную продолжительность процесса (overtime exception link), должна связывать исходный процесс с целевым процессом, предназначенным для обработки избыточных (дополнительных) работ, чтобы указать, что если во время их выполнения характеристики экземпляра исходного процесса будут превышать значение **Maximal Duration** (Максимальная продолжительность), то событие будет инициализировать целевой процесс.

Графически единичный короткий штрих, наклоненный к линии, которая соединяет исходный и целевой процессы и расположенный рядом с целевым процессом, должен обозначать связь, исключающую избыточную продолжительность процесса.

Принимая во внимание то, что **max-duration** — это значение свойства **Maximal Duration** (Максимальная продолжительность), а **time-unit** — это допустимая единица измерения времени, синтаксис связи, исключающей избыточную продолжительность процесса, будет выглядеть следующим образом: Процесс **Overtime Handling Destination Process** (Целевой процесс обработки сверхурочных работ) должен осуществляться, если продолжительность процесса **Source Process** (Исходный процесс) превышает значение **max-duration time-units** (максимальная продолжительность в единицах времени).

9.5.4.3 Связь, исключающая недостаточную продолжительность процесса

Связь, исключающая недостаточную продолжительность процесса (undertime exception link), должна связывать исходный процесс с целевым процессом, предназначенным для обработки недовыполненных работ, чтобы указать, что если во время их выполнения характеристики экземпляра исходного процесса будут ниже значения **Minimal Duration** (Минимальная продолжительность), то это событие будет инициализировать целевой процесс назначения.

Графически два параллельных коротких штриха, наклоненных к линии, которая соединяет исходный и целевой процессы, и расположенных рядом с целевым процессом, должны обозначать связь, исключающую недостаточную продолжительность процесса.

Принимая во внимание то, что **min-duration** — это значение свойства **Minimal Duration** (Минимальная продолжительность), а **time-unit** — это допустимая единица измерения времени, синтаксис связи, исключающей недостаточную продолжительность процесса, будет выглядеть следующим образом: Процесс **Undertime Handling Destination Process** (Целевой процесс обработки недовыполненных работ) должен осуществляться, если продолжительность процесса **Source Process** (Исходный процесс) будет ниже значения **min-duration time-units** (минимальная продолжительность в единицах времени).

Примечание — Аналогично инициализирующей связи две исключающие временные связи являются процедурными связями, которые непосредственно соединяют два процесса (в отличие от большинства других процедурных связей, соединяющих объекты и процессы). По сути, существует промежуточный объект **Overtime Exception Message** (Сообщение об избыточной продолжительности процесса) или объект **Undertime Exception Message** (Сообщение о недостаточной продолжительности процесса), созданный с помощью OPM-механизма выполнения процесса, который реализует процесс, потерявший работоспособность до его окончания за отведенное время или преждевременно завершившийся, не выдержав требований к минимально отведенному времени, соответственно. Поскольку операционный OPM-механизм создает объекты и немедленно их потребляет, их описание не является необходимым в рамках модели.

10 Структурные связи

10.1 Виды структурных связей

Структурные связи (structural link) определяют статические, не зависящие от времени, долгосрочные взаимоотношения в системе, соединяющие два или несколько объектов/процессов, но не объекты и процессы [за исключением случая демонстрационной характеристической связи (см. 10.3.3)]. Двумя видами структурных связей должны быть тегированные структурные связи (tagged structural links) и фундаментальные структурные связи (fundamental structural links) типа «агрегация — является частью» (aggregation-participation), «представление-характеризация» (exhibition-characterization), «обобщение-специализация» (generalization-specialization) и «классификация-инстанцирование» (classification-instantiation).

10.2 Тегированная структурная связь

10.2.1 Однонаправленная тегированная структурная связь

Однонаправленная тегированная структурная связь (unidirectional tagged structural link) должна обладать заданной пользователем семантикой относительно характера взаимоотношений между двумя сущностями (thing). Содержательный тег в виде текстовой фразы должен выражать характер структурных взаимоотношений между связующими объектами или связующими процессами, причем он должен передавать это содержание при его введении в OPL-предложении.

Графически стрелка и аннотация тэга у основания стрелки должны обозначать однонаправленную тегированную структурную связь.

Синтаксис однонаправленной тегированной структурной связи должен выражаться следующим OPL-предложением: Объект **Source-thing** (Сущность-источник) связан тегом **tag** с объектом **Destination-thing** (Сущность-получатель).

Примечание — Поскольку тег представляет собой метку, которую присваивает модели ее разработчик, в OPL-предложении тег-фразу выделяют жирным шрифтом, чтобы отличить ее от других слов с неявной синтаксической конструкцией.

10.2.2 Однонаправленная структурная связь с нуль-тегом

Однонаправленной структурной связью с нуль-тегом (unidirectional null-tagged structural link) должна быть однонаправленная тегированная структурная связь без аннотации тега (unidirectional tagged structural link with no tag annotation), что должно означать использование однонаправленного тега по умолчанию (в виде тега типа «иметь отношение к»).

Синтаксис однонаправленной структурной связи с нуль-тегом должен выражаться следующим OPL-предложением: Объект **Source-thing** (Сущность-источник) имеет отношение к объекту **Destination-thing** (Сущность-получатель).

Примечание — Разработчик модели специфической системы (или множества систем) должен иметь возможность задания однонаправленного тега по умолчанию, который не будет выделяться жирным шрифтом.

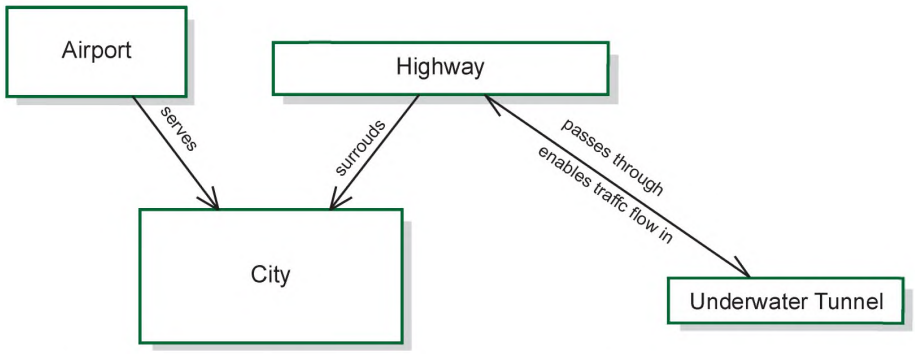
10.2.3 Двунаправленная тегированная структурная связь

Поскольку взаимосвязи между сущностями являются двунаправленными, каждая тегированная структурная связь должна обладать соответствующей тегированной структурной связью в противоположном направлении. Если теги в обоих направлениях обладают определенной содержательностью, а не просто являются инверсией друг друга, то они могут аннотироваться с помощью двух тегов с обеих сторон стрелки, символизирующей двунаправленную тегированную структурную связь.

Графически стрелка с наконечником в виде «гарпуна» на ее противоположных сторонах должна обозначать двунаправленную тегированную структурную связь (bidirectional tagged structural link), причем каждый тег должен располагаться со стороны стрелки с «гарпуном» и с краем, выходящим из наконечника стрелки (для однозначного определения направления, в котором используется каждая взаимосвязь).

Синтаксис полученной тегированной структурной связи должен выражаться парой OPL-предложений для раздельных однонаправленных тегированных структурных связей (по одному OPL-предложению для каждого направления).

Пример — На рисунке 14 представлены два вида тегированных структурных связей.



Физический объект **Airport** (Аэропорт) обслуживает физический объект **City** (Город).

Физический объект **Highway** (Автомагистраль) окружает физический объект **City** (Город).

Физический объект **Highway** (Автомагистраль) проходит через физический объект **Underwater Tunnel** (Подводный туннель).

Физический объект **Underwater Tunnel** (Подводный туннель) обеспечивает поток автотранспорта в физическом объекте **Highway** (Автомагистраль).

Рисунок 14 — Два вида тегированных структурных связей

10.2.4 Взаимная тегированная структурная связь

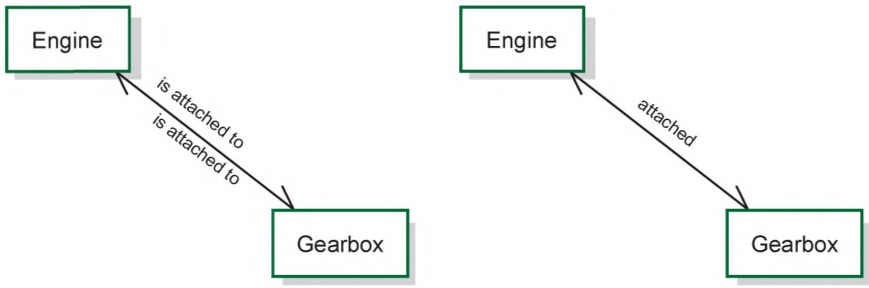
Взаимной тегированной структурной связью (reciprocal tagged structural link) должна быть двунаправленная тегированная структурная связь, имеющая только один тег (или вообще не имеющая ни одного тега). В любом случае принцип взаимности должен указывать на то, что тег двунаправленной

структурной связи имеет одну и ту же семантику для каждого направления связи. При отсутствии тега им по умолчанию должен быть тег типа «иметь отношение к» (are related).

Синтаксис взаимной тегированной структурной связи только с одним тегом должен выражаться следующим OPL-предложением: Объект **Source-thing** (Сущность-источник) и объект **Destination-thing** (Сущность-получатель) имеют тег **reciprocity-tag** (тег взаимности).

Синтаксис взаимной тегированной структурной связи без тега должен выражаться следующим OPL-предложением: Объект **Source-thing** (Сущность-источник) и объект **Destination-thing** (Сущность-получатель) являются взаимосвязанными.

Пример — На рисунке 15 справа показана взаимная структурная связь (reciprocal structure link), эквивалентная двунаправленной тегированной структурной связи (bidirectional tagged structure link), показанной слева, которая имеет тот же тег в каждом направлении.



Физический объект **Engine** (Двигатель) имеет тег **is attached to** (крепится к) для физического объекта **Gearbox** (Коробка передач).

Физический объект **Gearbox** (Коробка передач) имеет тег **is attached to** (крепится к) для физического объекта **Engine** (Двигатель).

Физические объекты **Engine** (Двигатель) и **Gearbox** (Коробка передач) имеют тег **attached** (скреплены).

Рисунок 15 — Диаграмма двунаправленной связи (слева) и эквивалентной ей взаимной тегированной структурной связи (справа)

Примечание — Как показано на рисунке 15, изменение форм глагола или существительного для двунаправленной тегированной структурной связи обычно необходимо для приведения в соответствие синтаксиса взаимной тегированной структурной связи.

10.3 Фундаментальные структурные взаимосвязи

10.3.1 Виды фундаментальных структурных взаимосвязей

Фундаментальные структурные взаимосвязи (fundamental structural relations) — это наиболее распространенные структурные взаимосвязи между OPM-сущностями, которые имеют особое значение для определения и понимания системы. Каждое из этих функциональных взаимоотношений должно уточнять и преобразовывать одну из исходных сущностей (source thing) типа *refineable* в совокупность из одной или нескольких конечных (целевых) сущностей (destination things) типа *refinees*.

Основными фундаментальными структурными взаимосвязями являются следующие:

- взаимосвязи типа «агрегация — является частью» (aggregation-participation), которые будут определять связи между целым и его частями;
- взаимосвязи типа «представление-характеризация» (exhibition-characterization), которые будут определять связи между представителем (exhibitor) — сущностью, которая представляет один или несколько признаков (атрибутов и/или операций), и сущностями, которые характеризуют этого представителя;
- взаимосвязи типа «обобщение-специализация» (generalization-specialization), которые будут определять связи между общей сущностью и ее конкретными реализациями;
- взаимосвязи типа «классификация-инстанцирование» (classification-instantiation), которые будут определять связи между классом сущностей и экземплярами сущностей типа *refinee* этого класса.

Взаимосвязи типа агрегация, представление, обобщение и классификация должны быть идентификаторами детализирующих взаимосвязей, т. е. идентификаторами, которые необходимо рассматривать

с точки зрения сущности типа *refineable*, а взаимосвязи типа является частью, характеризация, специализация и инстанцирование — должны быть соответствующими дополнительными идентификаторами взаимосвязей, т. е. идентификаторами, которые необходимо рассматривать с точки зрения объектов типа *refinees*.

За исключением взаимосвязи типа «представление-характеризация», целевые сущности объекта типа *refinee* должны иметь одно и то же значение устойчивости (Perseverance) как исходной сущности типа *refineable*, т. е. все объекты либо имеют статическое состояние Perseverance, либо динамическое состояние Perseverance.

Свертыванием (folding) объектов типа *refinees* должно быть сокрытие детализирующих сущностей типа *refineable*, а развертыванием (unfolding) этих сущностей для элементов типа *refinees* должно быть выражение сущностей типа *refineable* (см. 14.2.1.2).

Поскольку фундаментальные структурные взаимоотношения являются двунаправленными, связанный с ними OPL-раздел может давать OPL-предложения для каждого направления, однако из-за того, что одно из этих предложений всегда будет следствием другого, OPL-выражение для фундаментальной структурной взаимосвязи должно ограничиваться одним из двух возможных вариантов. Представление каждого вида этой взаимосвязи включает спецификацию OPL-предложения по умолчанию только для одного из двух возможных предложений (см. таблицу 14).

Совокупность объектов типа *refinees*, смоделированная для некоторой сущности типа *refineable* на определенной OPD-диаграмме, может быть полной или неполной, т. е. явно представлять графическое изображение, а соответствующий текст — явно выражать только те сущности, которые будут иметь отношение к OPD-диаграмме, на которой имеется структурная связь.

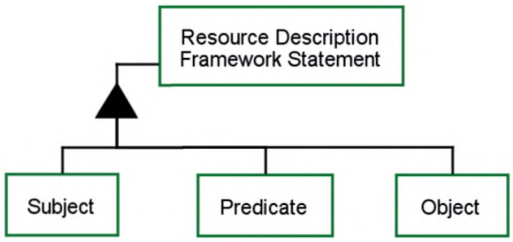
10.3.2 Реляционная связь типа «агрегация — является частью»

Фундаментальное структурное взаимоотношение типа «агрегация — является частью» (aggregation-participation) должно означать, что сущность типа *refineable* (whole) группирует одно или несколько объектов типа *refinees* (parts).

Графически зачерненный треугольник с вершиной, которая соединяется линией с объектом типа *whole*, и основанием, которое соединяется с объектами типа *parts*, должен обозначать условную связь типа «агрегация — является частью».

Синтаксис реляционной связи типа «агрегация — является частью» должен выражаться следующим OPL-предложением: Сущность типа **Whole-thing** состоит из частичных сущностей **Part-thing**$_1$, **Part-thing**$_2$, …, и **Part-thing**$_n$.

Пример 1 —



Объект **Resource Description Framework Statement** (Утверждение стандарта RDF [схема описания ресурсов]) состоит из объектов **Subject** (Субъект), **Predicate** (Предикат) и **Object** (Объект).

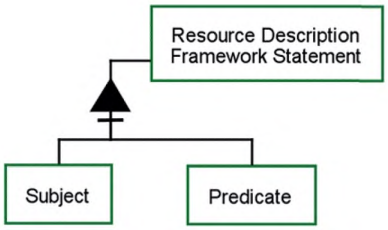
Рисунок 16 — Диаграмма для условной связи типа «агрегация — является частью»

Если представление совокупности частей при определенной степени детализации является неполным, то условная связь типа «агрегация — является частью» должна означать неполное представление с аннотацией.

Графически короткий горизонтальный штрих, пересекающий вертикальную линию под черным треугольником, должен обозначать условную связь типа «агрегация — является частью».

Синтаксис условной связи типа «агрегация — является частью», указывающий на частичную совокупность частей, в которой по меньшей мере одна часть отсутствует, должен выражаться следующим OPL-предложением: Сущность типа **Whole-thing** состоит из сущностей **of Part-thing**$_1$, **Part-thing**$_2$,… **Part-thing**$_k$ (Частичные сущности$_{1, 2... k}$) и по меньшей мере еще одной части.

Пример 2 — На рисунке 17 объект Object, изображенный на рисунке 16, отсутствует. Короткая горизонтальная черточка, пересекающая вертикальную линию под черным треугольником, означает недостающую сущность.



Объект **Resource Description Framework Statement** (Утверждение стандарта RDF [схема описания ресурсов]) состоит из объектов **Subject** (Субъект), **Predicate** (Предикат) и по меньшей мере еще одной части.

Рисунок 17 — Пример условной связи типа «агрегация — является частью» с частичным набором объектов типа *refine*

Пример 3 — В левой части рисунка 18 процесс Consuming (Потребление) потребляет объект Whole (Целый) вместе с объектами Part B (Часть B) и Part D (Часть D), тогда как объекты Part A (Часть A) и Part C (Часть C) потребляются обособленно. Справа на рисунке 18 представлен сокращенный вариант, в котором использована частичная агрегация, где показан процесс Consuming (Потребление), потребляющий объект Whole (Целый) и только объекты Part B (Часть B) и Part D (Часть D), тогда как другие части объекта Whole будут оставаться обособленными объектами.

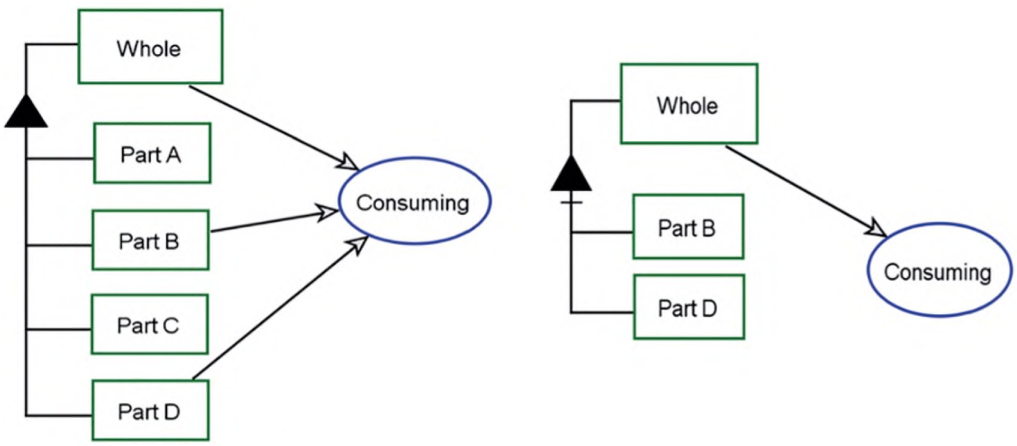


Рисунок 18 — Частичная агрегация потребления

Примечание — Инструментальное средство способно отслеживать набор объектов типа *refinees* для каждой сущности типа *refineable* и корректировать обозначения и соответствующие OPL-предложения (указанные ниже для каждой фундаментальной структурной реляционной связи) сразу же после того, как разработчик модели изменит набор объектов типа *refinees*.

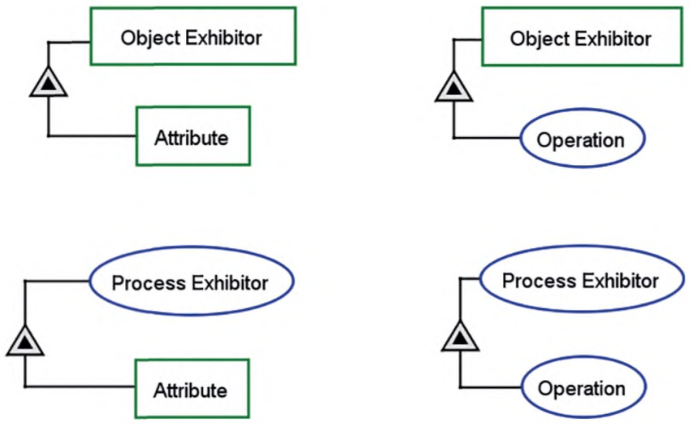
10.3.3 Связь типа «представление-характеризация»

10.3.3.1 Выражение условной связи типа «представление-характеризация»

Фундаментальная структурная условная связь типа «представление-характеризация» (fundamental structural relation exhibition-characterization) означает, что сущность типа *refineable* демонстрирует один или несколько признаков, которые характеризуют этого представителя (exhibitor) типа *refinees*.

Признак должен быть сущностью (thing). Атрибутом должен быть признак, который является объектом. Операцией должен быть признак, который является процессом. Представитель процесса и представитель объекта должны иметь по меньшей мере по одному признаку и могут иметь атрибуты и признаки объекта, операции и признаки процесса.

Взаимосвязь типа «представление — характеризация» может содержать четыре комбинации «представитель — признак» для объекта/процесса (см. рисунок 19).



Объект **Object Exhibitor** (Объект-представитель) представляет объект **Attribute** (Атрибут).

Процесс **Process Exhibitor** (Процесс-представитель) представляет объект **Attribute** (Атрибут).

Объект **Object Exhibitor** (Объект-представитель) представляет процесс **Operation** (Операция).

Процесс Process Exhibitor (Процесс-представитель) представляет процесс **Operation** (Операция).

Рисунок 19 — Четыре комбинации признаков взаимосвязи типа «представление-характеризация»

Графически маленький черный треугольник внутри большого незаполненного треугольника с вершиной, которая соединяется линией с представителем, и основанием треугольника, который соединяется с признаками, должны обозначать реляционную связь типа «представление-характеризация» (см. рисунок 19).

Синтаксис реляционной связи типа «представление-характеризация» для объекта-представителя с полным набором из n-атрибутов и m-операций, должен выражаться следующим OPL-предложением: Объект **Object-exhibitor** (Объект-представитель) представляет атрибуты **Attribute**$_1$, **Attribute**$_2$, … и **Attribute**$_n$, (Атрибуты$_{1,2... n}$) и операции **Operation**$_1$, **Operator**$_2$, … и **Operator**$_m$ (Операторы$_{1, 2,... m}$).

Синтаксис реляционной связи типа «представление-характеризация» с полным набором n-признаков операций и m-признаков атрибутов должен выражаться следующим OPL-предложением: Процесс **Process-exhibitor** (Процесс-представитель) представляет операции **Operation**$_1$, **Operation** $_2$, … и **Operation** $_n$ (Операции$_{1, 2,... n}$) и атрибуты **Attribute**$_1$, **Attribute**$_2$, … и **Attribute**$_m$ (Атрибуты$_{1,2... m}$).

Примечание 1 — В OPL-языке для взаимосвязи типа «представление-характеризация» для представителя объекта перечень атрибутов должен предшествовать перечню операций, тогда как для представителя процесса перечень операций должен предшествовать перечню атрибутов.

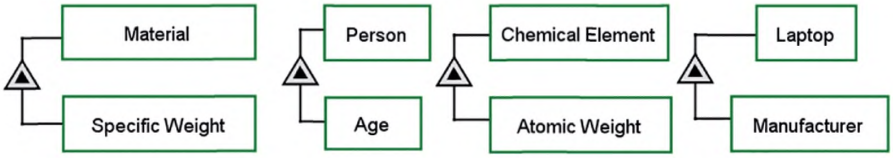
Если представление набора признаков с определенной степенью детализации является неполным, то реляционная связь типа «представление-характеризация» должна выражать неполное представление с аннотацией.

Графически короткая горизонтальная полоска, пересекающая вертикальную линию под большим незаполненным треугольником, означает неполную реляционную взаимосвязь типа «представление-характеризация».

Синтаксис реляционной взаимосвязи типа «представление-характеризация» для представителя объекта с неполным набором j-признаков атрибутов и k-признаков операций должен выражаться следующим OPL-предложением: Объект **Object-exhibitor-thing** (Объект-сущность представителя) представляет атрибуты **Attribute**$_1$, **Attribute**$_2$, …, **Attribute**$_j$ (Атрибуты$_{1,2... j}$) и по меньшей мере еще один атрибут и операции **Operation**$_1$, **Operation**$_2$, …, **Operation**$_k$ (Операции$_{1, 2,... n}$) и по меньшей мере еще одну операцию.

Синтаксис реляционной взаимосвязи типа «представление-характеризация» для представителя процесса с неполным набором из j-признаков операций и k-признаков атрибутов должен выражаться следующим OPL-предложением: Процесс **Process-exhibitor** (Процесс-представитель) представляет операции **Operation**$_1$, **Operation**$_2$, … , **Operation**$_j$ (Операции$_{1, 2... j}$) и по меньшей мере еще одну операцию, а также атрибуты **Attribute**$_1$, **Attribute**$_2$, …, **Attribute**$_k$ (Атрибуты$_{1,2... k}$) и по меньшей мере еще один атрибут.

Пример — На рисунках 20—23 приведены четыре комбинации взаимоотношений типа «представитель-признак» для объекта и процесса.



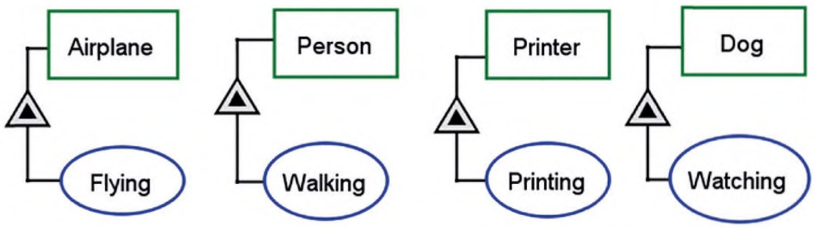
Объект **Material** (Материал) представляет объект **Specific Weight** (Удельный вес)

Объект **Person** (Человек) представляет объект **Age** (Возраст)

Объект **Chemical Element** (Химический элемент) представляет объект **Atomic Weight** (Атомный вес)

Объект **Laptop** (Ноутбук) представляет объект **Manufacturer** (Изготовитель)

Рисунок 20 — Примеры представления атрибутов объектов



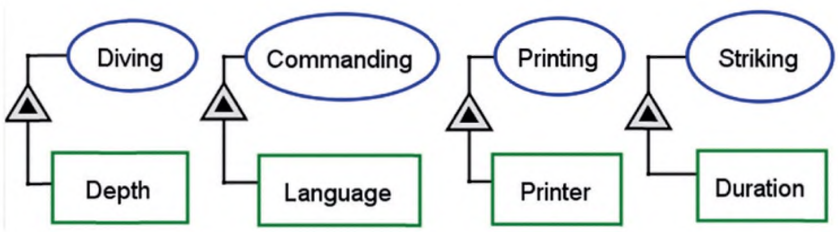
Объект **Airplane** (Самолет) представляет процесс **Flying** (Выполнение полета)

Объект **Person** (Человек) представляет процесс **Walking** (Ходьба)

Объект **Printer** (Принтер) представляет процесс **Printing** (Печать)

Объект **Dog** (Собака) представляет процесс **Watching** (Охрана)

Рисунок 21 — Примеры представления связей объекта с операциями



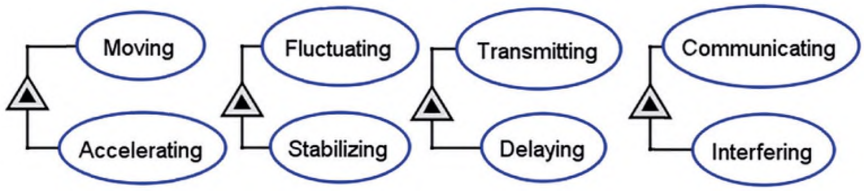
Процесс **Diving** (Ныряние) представляет объект **Depth** (Глубина)

Процесс **Commanding** (Командование) представляет объект **Language** (Язык)

Процесс **Printing** (Печать) представляет объект **Printer** (Принтер)

Процесс **Striking** (Зажигание) представляет объект **Duration** (Длительность)

Рисунок 22 — Примеры представления связей процесса с атрибутами



Процесс **Moving** (Перемещение) представляет процесс **Accelerating** (Ускорение)

Процесс **Fluctuating** (Флуктуирование) представляет процесс **Stabilizing** (Стабилизация)

Процесс **Transmitting** (Передача) представляет процесс **Delaying** (Задержка)

Процесс **Communicating** (Связь) представляет процесс **Interfering** (Создание помех)

Рисунок 23 — Примеры представления связей процессов с операциями

Примечание 2 — Инструментальное средство может отслеживать набор объектов типа *refinees* для каждой сущности типа *refineable* и корректировать символы и соответствующие OPL-предложения (указанные ниже для каждой фундаментальной структурной реляционной связи) сразу же после того, как разработчик модели изменит набор объектов типа *refinees*.

10.3.3.2 Состояние атрибута и признаки представителя

10.3.3.2.1 Атрибут состояния в качестве значения

Состояние атрибута, т. е. состояние объекта, который является атрибутом объекта типа *refinee*, должно быть значением этого атрибута. Статическая концептуальная модель должна идентифицировать все возможные значения атрибутов. Некоторые из них могут иметь диапазоны значений, тогда как для модели динамического операционного экземпляра следует указывать фактическое значение атрибута в момент его проверки (см. примеры 1 и 2 в 10.3.5.1).

10.3.3.2.2 Выражение взаимосвязи между представителем и признаком

При выражении признаков или значений атрибута модель должна идентифицировать их представителей. Для определения представителя признака в OPL-предложениях между признаком и его представителем должна иметься связка «of».

Синтаксис, идентифицирующий взаимосвязь между представителем и признаком, должен выражаться следующим OPL-предложением: Объект **Feature** (Признак) представителя **Exhibitor**...

Пример 1 — На рисунке 27 OPL-предложение с указанием наличия атрибута Specific Weight (Удельный вес) с помощью представителя Metal Powder Mixture (Металлическая порошкообразная смесь) выражается следующим образом: Атрибут Specific Weight в единицах г/см^3 представителя Metal Powder Mixture лежит в диапазоне значений от 7,545 до 7,537.

Пример 2 — На рисунке 25 OPL-предложение с указанием владения атрибута Travelling Medium (Среда перемещения) с помощью его представителя Ship (Корабль) выражается следующим образом: Атрибут Travelling Medium представителя Ship — это water surface (поверхность воды).

10.3.4 Связь типа «обобщение-специализация» и наследование

10.3.4.1 Реляционная связь типа «обобщение-специализация»

Фундаментальное структурное реляционное соотношение типа «обобщение-специализация» должно означать, что сущность типа *refineable* обобщает один или несколько объектов типа *refinees*, которые являются конкретными реализациями (специализациями) целого. Взаимосвязь типа «обобщение-специализация» связывает одну или несколько специализаций (конкретных реализаций) с одним и тем же свойством Perseverance (Устойчивость) как сущности типа *general*, так что эта сущность и все ее специализации являются процессами, или что эта сущность и все ее специализации являются объектами.

Графически незаполненный треугольник с вершиной, которая соединяется линией с сущностью типа *general*, и с основанием треугольника, который соединяется с конкретными реализациями, должны обозначать реляционную связь типа «обобщение-специализация» (см. рисунок 24).

Для полного набора из n-специализаций сущности типа *general*, которая является объектом, синтаксис реляционной связи типа «обобщение-специализация» должен выражаться следующим OPL-предложением: Объекты **Specialization-object$_1$**, **Specialization-object$_2$**, ... и **Specialization-object$_n$** являются объектами **General-object**.

Для полного набора из n-специализаций сущности типа *general*, которая является процессом, синтаксис реляционной связи типа «обобщение-специализация» выражается следующим OPL-предложением: Процесс **Specialization-process$_1$**, **Specialization-process$_2$**, ... и **Specialization-process$_n$** являются процессами **General-process**.

Если представление набора специализаций при определенной степени детализации является неполным, то реляционная связь типа «обобщение-специализация» должна означать неполное представление с аннотацией.

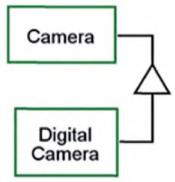
Графически короткая горизонтальная полоска, пересекающая вертикальную линию ниже незаполненного треугольника, должна обозначать неполную реляционную связь типа «обобщение-специализация».

Для неполного набора из k-специализаций сущности типа *general*, которая является объектом, синтаксис реляционной связи типа «обобщение-специализация» должен выражаться следующим OPL-предложением: Объект **Specialization-object$_1$**, **Specialization-object$_2$**, ... и **Specialization-object$_k$** и еще по меньшей мере одна из специализаций являются объектами **General-object**.

Для неполного набора k-специализаций сущности типа *general*, которая является процессом, синтаксис реляционной связи типа «обобщение-специализация» должен выражаться следующим

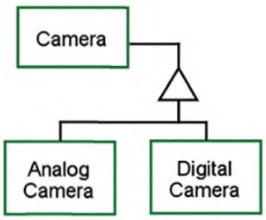
OPL-предложением: Процесс **Specialization-process$_1$**, **Specialization-process$_2$**, ... и **Specialization-process$_k$** и еще по меньшей мере одна из специализаций являются процессом **General-process**.

Пример — На рисунке 24 показаны одиночные и множественные специализации объектов и процессов.

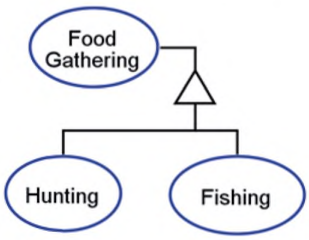


Объект **Digital Camera** (Цифровая камера) является специализацией объекта **Camera** (Камера)

Процесс **Hunting** (Охота) является специализацией процесса **Food Gathering** (Заготовка пищи)



Объекты **Analog Camera** (Аналоговая камера) и **Digital Camera** (Цифровая камера) являются специализацией объекта **Camera** (Камера)



Процессы **Hunting** (Охота) и **Fishing** (Рыболовство) являются специализацией процесса **Food Gathering** (Заготовка пищи)

Рисунок 24 — Одиночные и множественные специализации объектов и процессов

Примечание — Инструментальное средство может отслеживать множество объектов типа *refinees* для каждой сущности типа *refineable* и корректировать символ и соответствующие OPL-предложения для каждой фундаментальной структурной реляционной связи, поскольку разработчик модели изменяет набор объектов типа *refinees*.

10.3.4.2 Наследование через специализацию

Наследованием должно быть назначение OPM-элементов, сущностей и связей в направлении от общего к их специализациям.

Сущность типа *specialization* должна наследоваться из сущности типа *general* посредством связи типа «обобщение-специализация», каждая со следующими четырьмя видами существующих наследуемых элементов:

- все части сущности типа *general* — из ее связи типа «обобщение-специализация»;
- все признаки сущности типа *general* — из ее связи типа «представление-характеризация»;
- все тегированные структурные связи, с которыми связана сущность типа *general*;
- все процедурные связи, с которыми связана сущность типа *general*.

OPM-методология дает возможность выполнения нескольких наследований, позволяя сущности наследовать из нескольких сущностей типа *general* каждого из объектов типа *refinees* — следующих четырех наследуемых элементов [участники (participants), признаки (features), тегированные структурные связи (tagged structural links) и процедурные связи (procedural links)], которые существуют для данной сущности типа *general*.

Разработчик модели может переопределять любого из участников сущности типа *general*, которые по умолчанию были унаследованы специализацией, путем определения любого участника, унаследованного из сущности типа *general*, со специализацией этого участника под другим именем и другим набором состояний (см. 10.3.4.3).

Примечание — Если реляционная связь типа «представление-характеризация» существует, то во время выполнения экземпляр специализированной сущности не будет существовать (в отсутствие экземпляра более общей сущности, которая специализируется и из которой он наследует каждый из четырех видов наследуемых элементов).

Для создания сущности типа *general* из одной или нескольких возможных специализаций, наследуемые элементы, общие для каждой из этих специализаций, необходимо перемещать в обощенную сущность. Обработка наследуемых элементов должна осуществляться следующим образом:

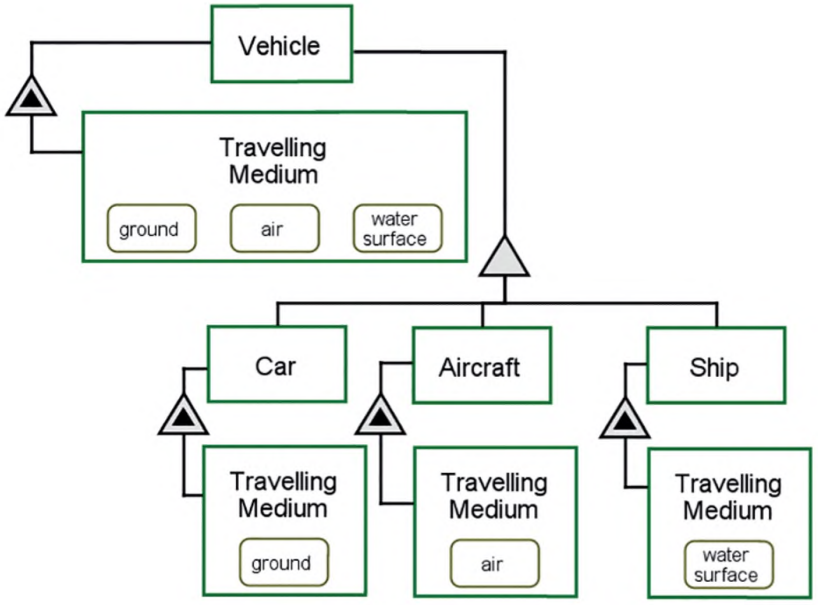
- ассоциации всех общих признаков и общих участников специализаций в одну вновь созданную сущность типа *general*;
- соединение новой сущности типа *general* (с использованием реляционной связи типа «представление-характеризация») со специализациями;
- удаление из специализаций всех общих признаков и общих участников, которые специализации в настоящее время наследуют из новой сущности типа *general*;
- перенос любых общих тегированных структурных связей и любого общего графа процедурной связи, который соединяется со всеми специализациями всех специализированных сущностей в обобщенную сущность.

10.3.4.3 Ограничение специализаций посредством дискриминирующего атрибута

Возможные значения атрибута, наследуемые из сущности типа *general*, могут ограничивать допустимые значения специализаций. Унаследованный атрибут с различными значениями, которые ограничивают различные значения соответствующих характеристик специализации, должны быть дискримирующим атрибутом.

Примечание — Специализация наследует признаки и возможные значения атрибутов обобщенной сущности. Разработка сущности типа *general* посредством детализации позволяет более точно оценивать наследуемые атрибута, в том числе спецификации на значение атрибута, приемлемого для определения характеристик специализации, посредством уточнения взаимосвязи типа «представление-характеризация», которые она наследует (см. также 10.4.1).

Пример 1 — На рисунке 25 приведена OPD-диаграмма, на которой объект Vehicle (Транспортное средство) представляет атрибут (объект) Travelling Medium (Среда перемещения) с состояниями (значениями) ground (земля), air (воздух) и water surface (поверхность воды). Атрибут Travelling Medium является дискриминирующим атрибутом объекта Vehicle, поскольку он ограничивает его специализации значениями среды перемещения. Объект Vehicle имеет специализации (объекты) Car (Автомобиль), Aircraft (Самолет) и Ship (Корабль) с соответствующими значениями атрибута Travelling Medium — ground, air и water surface.



Объект **Vehicle** (Транспортное средство) представляет атрибут **Travelling Medium** (Среда перемещения).

Значениями атрибута **Travelling Medium** (Среда перемещения) для объекта **Vehicle** (Транспортное средство) могут быть **ground** (земля), **air** (воздух) и **water surface** (поверхность воды).

Объекты **Car** (Автомобиль), **Aircraft** (Самолет) и **Ship** (Корабль) являются объектами **Vehicles** (Транспортные средства).

Атрибут **Travelling Medium** (Среда перемещения) для объекта **Car** (Автомобиль) имеет значение **ground** (земля).

Атрибут **Travelling Medium** (Среда перемещения) для объекта **Aircraft** (Самолет) имеет значение **air** (воздух).

Атрибут **Travelling Medium** (Среда перемещения) для объекта **Ship** (Корабль) имеет значение **water surface** (поверхность воды)

Рисунок 25 — Диаграмма для дискриминирующего атрибута **Travelling Medium** и его специализаций

Сущность типа *general* может иметь несколько дискриминирующих атрибутов. Максимальное число специализаций с несколькими дискриминирующими атрибутами должно быть равно прямому произведению числа возможных значений для каждого дискриминирующего атрибута, причем определенное сочетание значений атрибута может оказаться неприемлемым.

Пример 2 — Расширением рисунка 25 другим атрибутом объекта Vehicle (Транспортное средство) может быть сущность Purpose (Назначение) с двумя значениями — civilian (гражданское) и military (военное). Основываясь на этих двух значениях, могут быть предусмотрены две специализации объекта Vehicle: civilian Vehicle (гражданское транспортное средство) и military Vehicle (военное транспортное средство). Из-за множественного наследования его результатом будет матрица наследования, где число наиболее детальных специализаций будет 3 × 2 = 6 следующих значений: civilian Car (гражданский автомобиль), civilian Aircraft (гражданский самолет), civilian Ship (гражданский корабль), military Car (военный автомобиль), military Aircraft (военный самолет) и military Ship (военный корабль).

10.3.5 Связь типа «классификация-инстанцирование»

10.3.5.1 Реляционная связь типа «классификация-инстанцирование»

Фундаментальная структурная реляционная связь типа «классификация-инстанцирование» должна означать, что сущность типа *refineable* (класс), классифицирует один или несколько объектов типа *refinees* (экземпляры класса). Классификация, которая связана с классом объектов или классом процессов, представляет собой исходный шаблон для сущности, соединяющейся с одной или несколькими сущностями назначения, которые являются экземплярами исходного шаблона сущности, т. е. качеств (характерных особенностей) шаблона, определяющих получение точных значений для создания экземпляра сущности. Эта взаимосвязь дает разработчику модели точный механизм выражения отношений между классом и его экземплярами, что позволяет предоставлять нужные значения.

Примечание 1 — Использование термина «экземпляр» при рассмотрении элементов совокупности экземпляров концептуального класса называют «экземплярами объекта типа *refinee*», чтобы отличать их от «операционных экземпляров» операционной модели. Для каждого экземпляра объекта типа *refinee* может существовать один или несколько операционных экземпляров.

Примечание 2 — Все OPM-сущности, выраженные в концептуальной модели, представляют собой класс шаблонов для экземпляров этой сущности, которая предназначена для реализации в процессе оценки модели или ее эксплуатации. Путем создания сущности в концептуальной модели разработчик модели должен предполагать, что по меньшей мере один из операционных экземпляров этой сущности (или специализация этой сущности) может существовать в какой-то момент времени работы системы.

Если класс шаблона включает в себя связь типа «представление-характеризация», определяющую атрибут объекта типа *refinee* с допустимым диапазоном значений, то соответствующее значение атрибута для каждого операционного экземпляра объекта типа *refinee* этого класса должно находиться в пределах диапазона указанных значений признака атрибута класса (class attribute feature).

Графически маленький черный кружок внутри незаполненного большого треугольника с вершиной, которая соединяется линией с сущностью-классом, и с основанием треугольника, который соединяется с экземплярами сущностей, должен обозначать реляционную связь типа «классификация-инстанцирование».

Синтаксис реляционной связи типа «классификация-инстанцирование» между классом объекта и его единственным экземпляром должен выражаться следующим OPL-предложением: Объект **Instance-object** (Объект-экземпляр) является экземпляром объекта **Class-object** (Объект-класс).

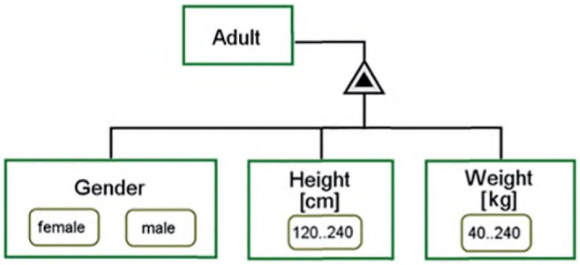
Синтаксис реляционной связи типа «классификация-инстанцирование» между классом процесса и его единственным экземпляром должен выражаться следующим OPL-предложением: Процесс **Instance-process** (Процесс-экземпляр) является экземпляром процесса **Class-process** (Процесс-класс).

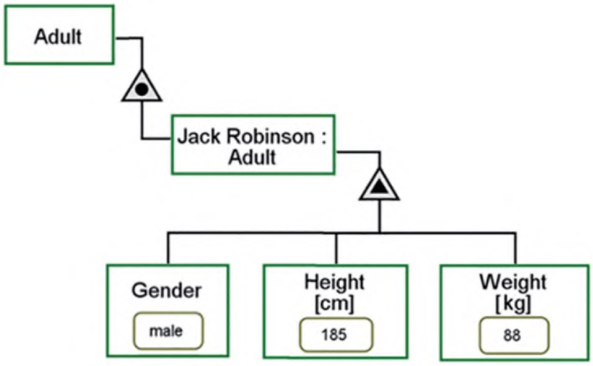
Синтаксис реляционной связи типа «классификация-инстанцирование» между классом процесса и его n-экземплярами должен выражаться следующим OPL-предложением: Объекты **Instance-object$_1$**, **Instance-object$_2$** ... и **Instance-object$_n$** являются экземплярами объекта **Class-object** (Объект-класс).

Синтаксис реляционной связи типа «классификация-инстанцирование» между классом процесса и его n-экземплярами должен выражаться следующим OPL-предложением: Процессы **Instance-process$_1$**, **Instance-process$_2$** ... и **Instance-process$_n$** являются экземплярами процесса **Class-process** (Процесс-класс).

Примечание 3 — Поскольку число экземпляров любого класса может быть априори неизвестно и может изменяться в процессе работы системы, для связи типа «классификация-инстанцирование» не существует никакого различия между полным и неполным наборами сущностей назначения.

Пример 1 — На рисунке 26 объект Adult (Взрослый) является классом с тремя атрибутами: Gender (Пол) с двумя возможными значениями (состояниями) female (женский) и male (мужской); Height, cm (Высота, см) с возможными значениями 120..240; Weight, kg (Вес, кг) с возможными значениями 40..240. Объект Jack Robinson (Джек Робинсон) является экземпляром объекта Adult со значениями для объектов Height (см), равным 185, и Weight (кг), равным 88, соответственно.





Класс **Adult** (Взрослый) представляется атрибутами **Gender** (Пол), **Height** (Высота, см) и **Weight** (Вес, кг).

Состоянием атрибута **Gender** (Пол) для класса **Adult** (Взрослый) может быть состояние **female** (женщина) или **male** (мужчина).

Состояние атрибута **Height** (Высота, см) для класса **Adult** (Взрослый) лежит в диапазоне **120..240.**

Состояние атрибута **Weight** (Вес, кг) для объекта **Adult** (Взрослый) лежит в диапазоне **40..240**.

Объект **Jack Robinson** (Джек Робинсон) является экземпляром класса **Adult** (Взрослый).

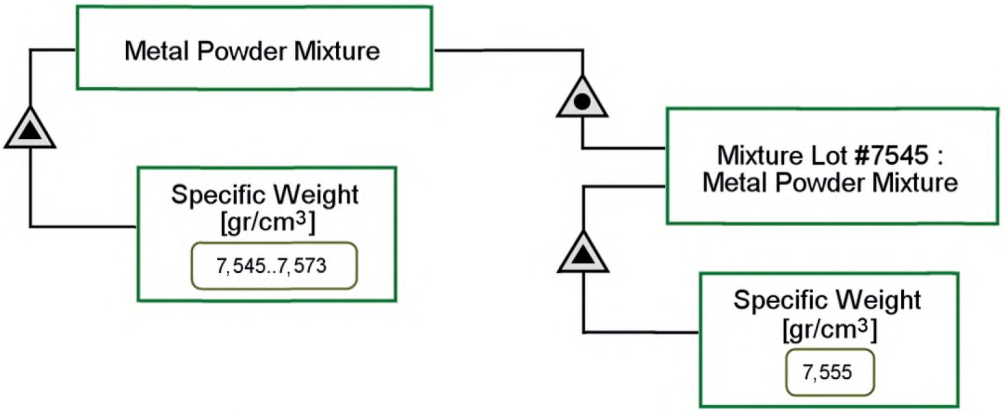
Атрибут **Gender** (Пол), связанный с объектом **Jack Robinson** (Джек Робинсон) имеет состояние **male** (мужчина).

Атрибут **Height** (Высота, см), связанный с объектом **Jack Robinson** (Джек Робинсон) имеет состояние **185.**

Атрибут **Weight** (Вес, кг), связанный с объектом **Jack Robinson** (Джек Робинсон) имеет состояние **88**.

Рисунок 26 — Диаграмма связи типа «классификация-инстанцирование» с диапазоном допустимых значений (класс и экземпляр, слева направо соответственно)

Пример 2 — OPD-диаграмма в левой части рисунка 27 — это концептуальная модель объекта Metal Powder Mixture (Смесь порошков металла), показывающая, что значение его атрибута Specific Weight (Удельный вес) может лежать в диапазоне от 7,545 до 7,573 г/см^3. Рисунок 27 представляет собой операционную экземплярную (динамическую) модель Metal Powder Mixture-Instance (Экземпляр смеси порошков металла), показывающую, что значение атрибута Specific Weight, равное 7,555 г/см ^{3}, лежит в диапазоне допустимых значений.



Объект **Metal Powder Mixture** (Смесь порошков металла) представляется атрибутом **Specific Weight, г/см^3** (Удельный вес).

Значение атрибута **Specific Weight, г/см^3** (Удельный вес) объекта **Metal Powder Mixture** (Смесь порошков металла) лежит в диапазоне от **7,545** до **7,573**.

Объект **Mixture Lot #7545** (Партия смеси № 7545) является экземпляром объекта **Metal Powder Mixture** (Смесь порошков металла).

Атрибут **Specific Weight, г/см^3** (Удельный вес) объекта **Mixture Lot #7545** (Партия смеси № 7545) равен **7,555**.

Рисунок 27 — Состояние атрибута как значения: взаимосвязь между концептуальной и операционной моделями

Примечание 4 — OPL-предложение «Объект Mixture Lot #7545 (Партия смеси № 7545) представляет атрибут Specific Weight, г/см^3 (Удельный вес)» не представлено на рисунке 27, поскольку это предложение подразумевается четко выраженным фактом, что «Объект Mixture Lot #7545 является экземпляром объекта Metal Powder Mixture», и поэтому объект Mixture Lot #7545 наследует этот атрибут от объекта Metal Powder Mixture (Смесь порошков металла).

10.3.5.2 Экземпляры класса объектов и класса процессов

Класс объектов и класс процессов должны быть двумя различными видами классов. Экземпляр класса должен быть воплощением идентифицируемого экземпляра класса с тем же идентификатором классификации.

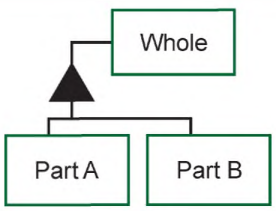
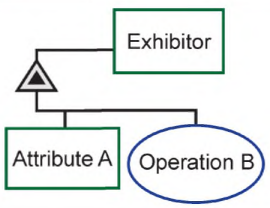
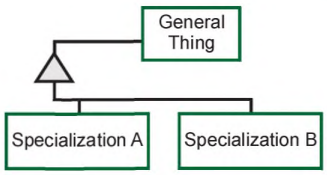
Один объект типа *refinee* должен быть экземпляром объекта, тогда как шаблон объекта, которому все экземпляры принадлежат, должен быть классом сущностей типа *refineable*.

Класс процессов должен быть шаблоном происходящего (последовательностью подпроцессов), который включает в себя классы объектов, которые являются элементами наборов предварительно и апостериорно обработанных объектов. Проявлением процесса, который соответствует этому шаблону и включает в себя конкретные экземпляры объектов в их наборах предварительно и апостериорно обработанных объектов, должен быть экземпляр процесса, поэтому экземпляром процесса должно быть конкретное проявление класса процесса, к которому принадлежит данный экземпляр. Любой экземпляр процесса должен иметь связанный с ним определенный набор предварительно и апостериорно обработанных экземпляров объекта.

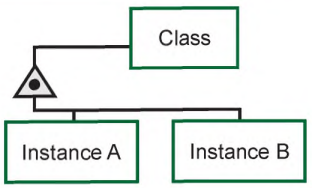
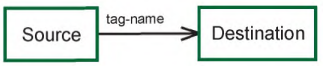
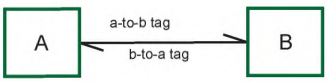
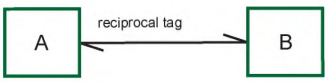
Примечание — Сильной стороной концепции класса процесса является то, что она позволяет моделировать процесс как шаблон или протокол некоторого преобразования, которому подвергается класс объектов. Это преобразование не включает в себя ни пространственно-временную структуру, ни конкретный набор экземпляров объекта, с помощью которых могут объединяться экземпляры процесса.

10.3.6 Обзор структурных реляционных и тегированных структурных связей

Таблица 14 — Обзор структурных реляционных (structural relations) и тегированных структурных (tagged structural) связей

Структурная реляционная связь (Structural Relation) Прямая-Обратная (Forward-Reverse) (refineable-to-refinee) (жирным шрифтом выделено сокращенное название)	OPD-символ	OPL-предложение	
		Прямое (refineable-to-refinee)	Обратное (refinee-to-refineable)
Связь типа «агрегация — является частью»		Объект типа **Whole** состоит из объекта **Part A** (Часть А) и объекта **Part B** (Часть В)	—
Связь типа «представление-характеризация»		Объект **Exhibitor** (Представитель) представляется объектом **Attribute A** (Атрибут А) и процессом **Operation B** (Операция В)	—
Связь типа «обобщение-специализация»		—	Объект **Specialization A** (Специализация А) и объект **Specialization B** (Специализация В) являются объектом типа **General Thing** (Общая сущность)

Окончание таблицы 14

Структурная реляционная связь (Structural Relation) Прямая-Обратная (Forward-Reverse) (refineable-to-refinee) (жирным шрифтом выделено сокращенное название)	OPD-символ	OPL-предложение	
		Прямое (refineable-to-refinee)	Обратное (refinee-to-refineable)
Связь типа «классификация-инстанцирование»		—	Объект **Instance A** (Экземпляр А) и объект **Instance B** (Экземпляр В) являются экземплярами класса **Class** (Класс)
Однонаправленная тегированная связь [однонаправленная нулевая тегированная связь]		Объект **Source** (Источник) с помощью тега **tag-name** связывается с объектом **Destination** (Получатель). [Объект **Source** связывается с объектом **Destination**]	
Двунаправленная тегированная связь		Объект **A** с помощью тега **a-to-b tag** связывается с объектом **B.** Объект **B** с помощью тега **b-to-a tag** связывается с объектом **A**	
Взаимная тегированная связь [взаимная нулевая тегированная связь]		Объекты **A** и **B** с помощью тега **reciprocal tag** (взаимно-тегированный) связываются друг с другом. [Объекты **A** и **B** с помощью тега **reciprocal tag** (взаимно-тегированный) становятся взаимосвязанными]	

10.4 Определяющая состояние характеристическая реляционная связь

10.4.1 Определяющая состояние характеристическая реляционная связь

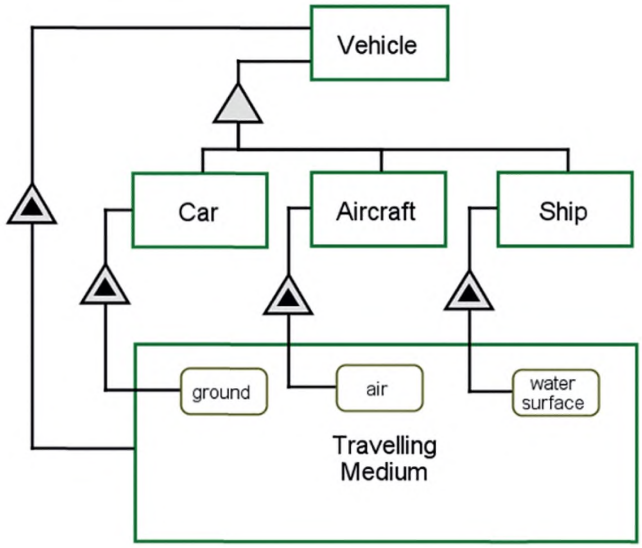
Определяющей состояние характеристической реляционной связью (state-specified characterization relation link) должна быть реляционная связь типа «представление-характеризация» между специализированным объектом, который представляет значение атрибута, и дискриминирующим атрибутом его обобщения, означающим, что специализированный объект должен обладать только тем значением атрибута, которое он унаследовал.

Графически определяющая состояние характеристическая реляционная связь в виде треугольника с вершиной, которая соединяется со специализированным объектом, а основание треугольника — со значением, представляющим состояние, должна обозначать эту связь.

Примечание — Хотя в этом нет особой необходимости, OPD-диаграмма будет более понятной, если связь типа «представление-характеризация» для объекта типа *general* с дискриминирующим атрибутом будет также изображаться на этой же OPD-диаграмме (см. рисунок 28).

Синтаксис определяющей состояние характеристической реляционной связи должен выражаться следующим OPL-предложением: Объект **Specialized-object** (Специализированный объект) представляется атрибутом **value-name** (значение-имя) для объекта **Attribute-Name** (Атрибут-имя).

Пример — С использованием определяющей состояние характеристической реляционной связи OPD-диаграмма, приведенная на рисунке 28, становится значительно более компактной, чем ее OPD-эквивалент на рисунке 25. Здесь дискриминирующий атрибут Travelling Medium (Среда перемещения) объекта Vehicle (Транспортное средство) с объектами Car (Автомобиль), Aircraft (Самолет) и Ship (Корабль) являются специализациями объекта Vehicle, которые появляются только один раз (в противоположность четырем случаям на рисунке 25). В модели объекты Car, Aircraft и Ship являются специализациями объекта Vehicle, соединяющим каждую специализацию (с помощью определяющей состояние характеристической реляционной связи) с соответствующими состояниями атрибута Travelling Medium — ground (земля), air (воздух) и water surface (поверхность воды), соответственно.



Объект **Vehicle** (Транспортное средство) представляется атрибутом **Travelling Medium** (Среда перемещения).

Состояниями атрибута **Travelling Medium** (Среда перемещения) объекта **Vehicle** (Транспортное средство) могут быть **ground** (земля), **air** (воздух) и **water surface** (поверхность воды).

Объекты **Car** (Автомобиль), **Aircraft** (Самолет) и **Ship** (Корабль) являются объектами **Vehicles** (Транспортные средства).

Объект **Car** представляется значением **ground** (земля) атрибута **Travelling Medium** (Транспортное средство).

Объект **Aircraft** представляется значением **air** (воздух) атрибута **Travelling Medium** (Транспортное средство).

Объект **Ship** представляется значением **water surface** (поверхность воды) атрибута **Travelling Medium** (Транспортное средство).

Рисунок 28 — Пример определяющей состояние характеристической реляционной связи

10.4.2 Определяющие состояние тегированные структурные реляционные связи

10.4.2.1 Определяющие состояние тегированные структурные связи

Определяющей состояние тегированной структурной связью (state-specified tagged structural link) должна быть тегированная структурная связь между состоянием объекта (или значением атрибута) и другим объектом (его состоянием или значением атрибута), обозначающим взаимосвязь между этими двумя сущностями и тегом, выражающим семантику этой взаимосвязи. В случае нулевого тега, т. е. в случае отсутствия явной спецификации тега, соответствующий OPL-синтаксис должен по умолчанию использовать нулевой тег (см. 10.2.2).

Должны существовать три вида определяющих состояние тегированных структурных связей: определяющая состояние тегированная структурная связь для источника (source state-specified tagged structural link), определяющая состояние тегированная структурная связь для получателя (destination state-specified tagged structural link) и определяющая состояние тегированная структурная связь для источника/получателя (source-and-destination state-specified tagged structural link). Каждый из указанных видов связи должен содержать однонаправленную, двунаправленную и взаимно-направленную тегированные структурные связи, что приводит к возникновению семи видов определяющих состояние тегированных структурных реляционных связей и соответствующих OPL-предложений (см. таблицу 15).

10.4.2.2 Однонаправленная определяющая состояние тегированная структурная связь для источника

Однонаправленной определяющей состояние тегированной структурной связью для источника (unidirectional source state-specified tagged structural link) должна быть однонаправленная тегированная структурная связь между заданным состоянием объекта-источника и целевым объектом (объектом-получателем) без спецификации состояния.

Графически стрелка с наконечником, соединяющая состояние объекта-источника с объектом-получателем, с аннотацией имени тега возле основания стрелки, должна обозначать однонаправленную определяющую состояние тегированную структурную связь для источника.

Синтаксис однонаправленной определяющей состояние тегированной структурной связи для источника должен выражаться следующим OPL-предложением: **Specified-state source-object tag-name Destination-object**.

Примечание — Нуль-тег по умолчанию использует имя тега типа «относится к», не выделенное жирным шрифтом (если он не будет изменен разработчиком модели).

10.4.2.3 Однонаправленная определяющая состояние тегированная структурная связь для получателя

Однонаправленной определяющей состояние тегированной структурной связью (unidirectional destination state-specified tagged structural link) для получателя должна быть однонаправленная тегированная структурная связь между состоянием объекта-источника без спецификации состояния и заданным состоянием целевого объекта (объекта-получателя).

Графически стрелка с наконечником, соединяющая объект-источник с заданным состоянием объекта-получателя, с аннотацией имени тега возле основания стрелки, должна обозначать однонаправленную определяющую состояние тегированную структурную связь для получателя.

Синтаксис однонаправленной определяющей состояние тегированной структурной связи для получателя должен выражаться следующим OPL-предложением: **Source-object tag-name specified-state Destination-object**.

Примечание — Нуль-тег по умолчанию использует имя тега типа «относится к», не выделенное жирным шрифтом (если он не будет изменен разработчиком модели).

10.4.2.4 Однонаправленная определяющая состояние тегированная структурная связь для источника и получателя

Однонаправленной определяющей состояние тегированной структурной связью (unidirectional source-and-destination state-specified tagged structural link) для источника и получателя должна быть двунаправленная тегированная структурная связь от определенного состояния объекта-источника к определенному состоянию объекта-получателя.

Графически стрелка с наконечником, соединяющая заданное состояние объекта-источника с заданным состоянием объекта-получателя, с аннотацией имени тега возле основания стрелки, должна обозначать этот тип связи.

Синтаксис однонаправленной определяющей состояние тегированной структурной связи для источника и получателя должен выражаться следующим OPL-предложением: **Source-specified-state source** (Объект-источник с определенным исходным состоянием) и объект **destination-specified-state Destination-object** (Объект-получатель с определенным конечным состоянием) имеют тег **object tag-name**.

Примечание — Нулевой тег по умолчанию использует имя тега типа «относится к», не выделенное жирным шрифтом (если он не будет изменен разработчиком модели).

10.4.2.5 Двунаправленная определяющая состояние тегированная структурная связь для источника или получателя

Двунаправленной определяющей состояние тегированной структурной связью для источника или получателя (bidirectional source-or-destination state-specified tagged structural link) должна быть двунаправленная тегированная структурная связь между заданным состоянием либо объекта-источника, либо объекта-получателя (но не с ними обоими вместе).

Графически стрелка с наконечниками в виде «гарпунов», соединяющая противоположными концами объект (или его состояние) и состояние объекта (или объект), соответственно, должна обозначать двунаправленную определяющую состояние тегированную структурную связь для источника или получателя. Каждое имя тега должно располагаться вдоль основания стрелки с наконечниками в виде «гарпунов», однозначно определяя направление, по которому действует каждая из связей.

Синтаксис двунаправленной определяющей состояние тегированной структурной связи для источника или получателя должен выражаться OPL-предложениями, соответствующими каждому направлению (с соответствующими спецификациями состояния.

10.4.2.6 Двунаправленная определяющая состояние тегированная структурная связь для источника и получателя

Двунаправленной определяющей состояние тегированной структурной связью для источника и получателя (bidirectional source-and-destination state-specified tagged structural link) должна быть двуна-

правленная тегированная структурная связь между заданным объектом-источником и объектом-получателем.

Графически линия с наконечниками в виде «гарпунов», соединяющая противоположными концами заданное состояние одного объекта с заданным состоянием другого объекта, должна обозначать двунаправленную определяющую состояние тегированную структурную связь для источника и получателя. Каждое имя тега должно располагаться вдоль основания стрелки с наконечниками в виде «гарпунов», однозначно определяя направление, по которому действует каждая из связей.

Синтаксис двунаправленной определяющей состояние тегированной структурной связи для источника и получателя должен выражаться OPL-предложениями, соответствующими каждому направлению (с соответствующими спецификациями состояния и именами тегов).

10.4.2.7 Взаимная определяющая состояние тегированная структурная связь для источника или получателя

Взаимной определяющей состояние тегированной структурной связью для источника или получателя (reciprocal source-or-destination tagged structural link) должна быть двунаправленная тегированная структурная связь для источника или получателя для одного из рассматриваемых объектов (но не обоих объектов вместе) и только одним тегом взаимности (reciprocity-tag) или без тега. В любом случае взаимность должна указывать, что тег взаимной определяющей состояние тегированной структурной связи для источника или получателя имеет одну и ту же семантику для каждого направления взаимосвязи. При отсутствии тега им по умолчанию должен быть тег «are related (связанный)».

Графически линия с наконечниками в виде «гарпунов», соединяющая противоположными концами заданное состояние одного объекта с другим объектом (без спецификации состояния) и обозначающая только имя тега, располагаемого вдоль основания стрелки, должна обозначать взаимную определяющую состояние тегированную структурную связью для источника или получателя.

Синтаксис взаимной определяющей состояние тегированной структурной связи для источника или получателя лишь с одним тегом должен выражаться одним из следующих OPL-предложений: Объект **Source-specified-state Source-object** (Объект-источник с определенным исходным состоянием) и объект **Destination-object** (Объект-получатель) имеют тег **reciprocity-tag**; или объект **Source-object** (Объект-источник) и объект **destination-specified-state Destination-object** (Объект-получатель с определенным конечным состоянием) имеют тег **reciprocity-tag**.

10.4.2.8 Взаимная определяющая состояние тегированная структурная связь для источника и получателя

Взаимной определяющей состояние тегированной структурной связью для источника и получателя (reciprocal source-and-destination tagged structural link) должна быть двунаправленная тегированная структурная связь для источника и получателя с определенным состоянием обоих участвующих объектов лишь с одним взаимным тегом или вообще без тега. В любом случае взаимность должна указывать, что тег этой связи имеет ту же семантику в обоих направлениях связи. При отсутствии тега им по умолчанию должен быть тег «are related» (связанный).

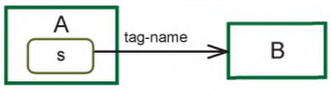
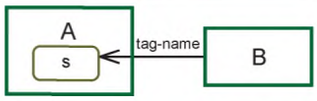
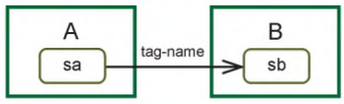
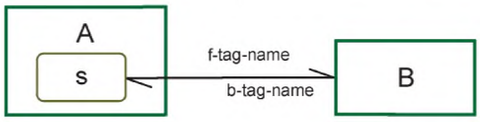
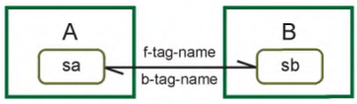
Графически линия с наконечниками в виде «гарпунов», соединяющая противоположными концами заданное состояние одного объекта с заданным состоянием другого объекта и обозначающая только имя тега, располагаемого вдоль основания стрелки, должна обозначать взаимную определяющую состояние тегированную структурную связью для источника и получателя.

Синтаксис взаимной определяющей состояние тегированной структурной связи для источника и получателя лишь с одним тегом должен выражаться одним из следующих OPL-предложений: Объект **Source-specified-state Source-object** (Объект-источник с определенным исходным состоянием) и объект **destination-specified-state Destination-object** (Объект-получатель с определенным конечным состоянием) имеют тег **reciprocity-tag**.

Синтаксис взаимной определяющей состояние тегированной структурной связи для источника и получателя без тега должен выражаться одним из следующих OPL-предложений: Объект **Source-specified-state Source-object** (Объект-источник с определенным исходным состоянием) и объект **destination-specified-state Destination-object** (Объект-получатель с определенным конечным состоянием) являются связанными.

10.4.2.9 Обзор определяющих состояние структурных реляционных связей

Таблица 15 — Обзор определяющих состояние структурных реляционных отношений и связей (state-specified structural relations and links)

Направленность	Источник/Получатель		
	Зависящий от состояния источник	Зависящий от состояния получатель	Зависящие от состояния источник и получатель
Однонаправленная связь	 Связь состояния **s** объекта **A** с объектом **B** имеет тег **tag-name**	 Связь объекта **B** с состоянием **s** объекта **A** имеет тег **tag-name**	 Связь состояния **sa** объекта **A** с состоянием **sb** объекта **B** имеет тег **tag-name**
Двунаправленная связь	 Связь состояния **s** объекта **A** с объектом **B** имеет тег **f-tag-name B.** Связь объекта **B** с состоянием **s** объекта **A** имеет тег **b-tag-name**		 Связь состояния **sa** объекта **A** с состоянием **sb** объекта **B** имеет тег **f-tag-name.** Связь состояния **sb** объекта **B** имеет тег **b-tag-name**
Взаимная связь	 Взаимосвязь объекта **B** и состояния **s** объекта **A** имеет тег **recip-tag-name**		 Взаимосвязь состояния **sa** объекта **A** и состояния **sb** объекта **B** имеет тег **recip-tag-name**

Пример 1 — На OPD-диаграмме, приведенной на рисунке 29, объект Keeper (Владелец чека) является атрибутом объекта Check (Чек) со значениями payer (плательщик), payee (получатель платежа) и bank (банк). Каждое из этих значений также является объектом со своими собственными правами в модели. Три однонаправленные определяющие состояние структурные связи с нуль-тегами соединяют каждое из этих значений с соответствующими физическими объектами. Отметим отсутствие требований к тому, чтобы имена состояний были аналогичны именам соответствующих объектов (например, имен состояния financial institution (финансово-кредитное учреждение) и объекта Bank (Банк).

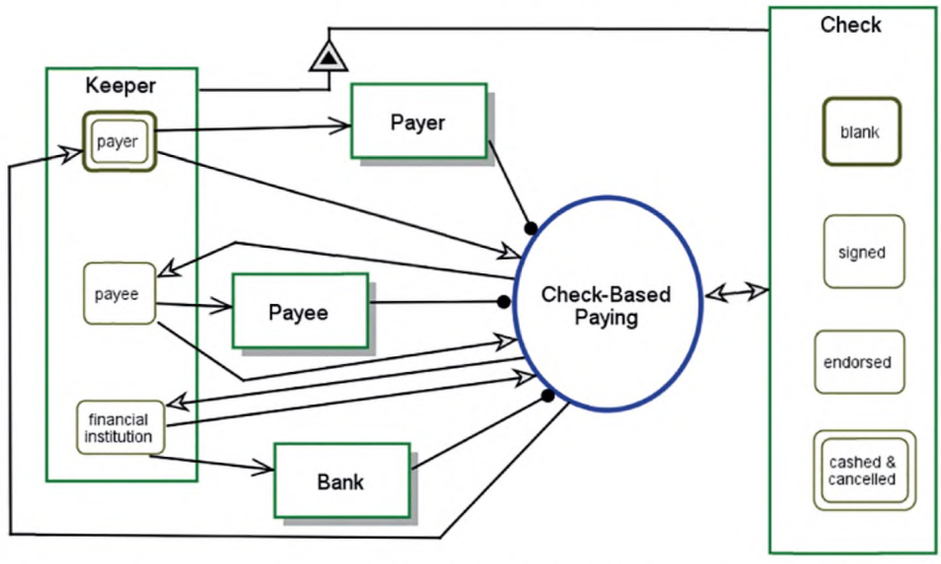


Рисунок 29, лист 1 — Связь значений атрибутов с объектами с использованием определяемой состоянием структурной связи

Объект **Check** (Чек) может иметь состояние **blank** (неподписанный), **signed** (подписанный), **endorsed** (индоссированный) или **cashed & cancelled** (инкассированный & аннулированный).

Объект **Check** (Чек) представляется объектом **Keeper** (Владелец чека).

Объект **Keeper** (Владелец чека) может иметь состояние **payer** (плательщик), **payee** (получатель платежа) или **financial institution** (финансово-кредитное учреждение).

Состояние **payer** (плательщик) объекта **Keeper** (Владелец чека) связано с физическим объектом **Payer** (Плательщик).

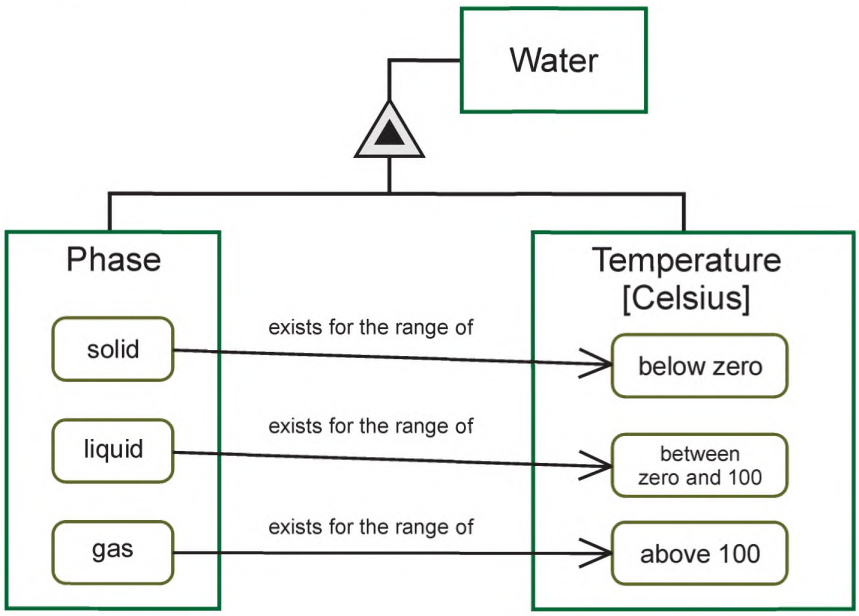
Состояние **payee** (получатель платежа) объекта **Keeper** (Владелец чека) связано с физическим объектом **Payee** (Получатель платежа).

Состояние **financial institution** (финансово-кредитное учреждение) объекта **Keeper** (Владелец чека) связано с физическим объектом **Bank** (Банк).

Check-Based Paying — Процесс финансовых расчетов на основе чеков.

Рисунок 29, лист 2

Пример 2 — На OPD-диаграмме рисунка 30 каждое из трех значений (состояний) объекта Phase (Фазовое состояние) для объекта Water (Вода) связывается с соответствующими состояниями (диапазонами) объекта Temperature [Celsius] (Температура по шкале Цельсия) с помощью трех определяющих состояние тегированных структурных связей для источника и получателя, имеющих тег «exists for the range of» (существует для диапазона ...).



Объект **Water** (Вода) представляется объектами **Phase** (Фазовое состояние) и **Temperature [Celsius]** (Температура по шкале Цельсия).

Объект **Phase** (Фазовое состояние) для объекта **Water** (Вода) может находиться в состоянии **solid** (твердое), **liquid** (жидкое) или **gas** (газообразное).

Объект **Temperature [Celsius]** (Температура по шкале Цельсия) для объекта **Water** (Вода) может находиться в диапазоне **below zero** (ниже нуля), **between zero and 100** (между 0 и 100) или **above 100** (выше 100).

Состояние **solid** (твердое) объекта **Phase** (Фазовое состояние) существует для диапазона **below zero** (ниже нуля) объекта **Temperature [Celsius]** (Температура по шкале Цельсия).

Состояние **liquid** (жидкое) объекта **Phase** (Фазовое состояние) существует для диапазона **between zero and 100** (между 0 и 100) объекта **Temperature [Celsius]** (Температура по шкале Цельсия).

Состояние **gas** (газообразное) объекта **Phase** (Фазовое состояние) существует для диапазона **above 100** (выше 100) объекта **Temperature [Celsius]** (Температура по шкале Цельсия).

Рисунок 30 — Диаграмма определяющей состояние тегированной структурной связи для источника и получателя

11 Мощности отношений (связей)

11.1 Кратность объектов в структурных и процедурных связях

Кратность объекта должна относиться к требованиям или характеристикам ограничений (которую иногда называют «ограничением участия») на количество или число объектных операционных экзем-

пляров, относящихся к определенной связи. При отсутствии спецификации на кратность объекта каждый конец связи должен определять лишь один объектный операционный экземпляр. Спецификации на кратность объекта могут появляться в следующих ситуациях:

a) при определении нескольких объектных операционных экземпляров источника или получателя для тегированной структурной связи любого вида;

b) при определении объекта-участника с несколькими оперативными экземплярами для связи типа «агрегация — является частью», где различные характеристики участия могут присоединяться к каждой из частей целого;

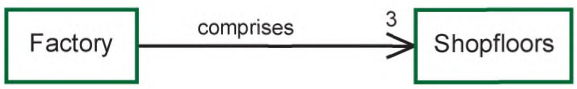
c) при определении объекта с несколькими оперативными экземплярами в процедурной связи.

Характеристикой кратности объекта могут быть целые числа или символы параметров, разделяемые целочисленными значениями при прогоне модели, которые могут включать в себя арифметические выражения. Эта характеристика также может содержать в себе диапазон значений (или набор диапазонов значений).

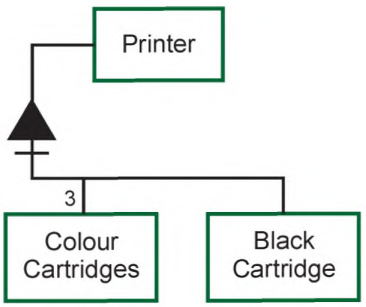
Графически целое число, диапазон целочисленных значений, символ параметра, диапазон символов параметров или набор целых чисел или символов параметров, любой из которых может появляться в виде аннотаций ближе к концу линии, к которой он относится, должны обозначать кратность объекта.

Синтаксис OPL-предложения для объекта с кратностью должен содержать перед именем объекта значение его кратности, причем это имя следует указывать во множественном числе (если возможно определение нескольких операционных экземпляров). Ниже приведены примеры некоторых из многих видов использования объектов кратности на OPL-предложениях.

Пример — На рисунке 31 показаны: слева — OPD-диаграмма ограничения на участие (на стороне получателя) для однонаправленной тегированной структурной связи, а справа — OPD-диаграмма ограничения на участие (на стороне получателя) одного из двух объектов для связи типа «агрегация — является частью».



Объект **Factory** (Фабрика) связан с 3 объектами **Shopfloors** (Производственное помещение) и имеет тег **comprises** (содержит).



Объект **Printer** (Принтер) содержит 3 объекта **Colour Cartridges** (Цветные картриджи), объект **Black Cartridge** (Черный картридж) и другие части.

Рисунок 31 — Примеры кратности объектов

Кратностью объекта может быть параметр, диапазон параметров или совокупность из двух или более диапазонов чисел и/или параметров, разделенных запятой. Диапазон следует указывать как q_{min} .. q_{max} и он должен быть «закрытым», т. е. включать граничные значения q_{min} и q_{max}. В OPL-синтаксисе выражение для символа диапазона «..» должно означать «to» («к»), а выражение запятой, разделяющей два смежных диапазона, должно означать «or» («или»).

Спецификация кратности объекта может существовать в качестве опционального параметра, используя при этом символ диапазона, символ звездочки и знака вопроса следующим образом:

- «0..1» должно означать ноль или один, используя для этого аннотацию в виде знака вопроса (?) около объекта, к которому он относится, с OPL-синтаксисом типа «an optional» («на выбор») непосредственно перед объектом;

- «0 .. *» должно означать ноль или большее значение, используя для этого аннотацию в виде звездочки (*) около объекта, к которому он относится, с OPL-синтаксисом типа «an optional» («на выбор») непосредственно перед объектом, и

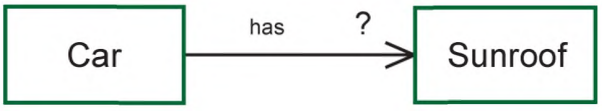
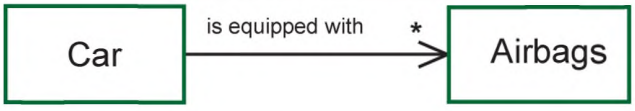
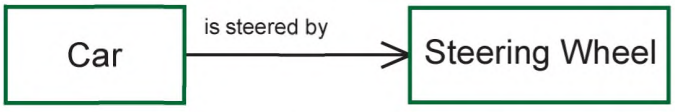
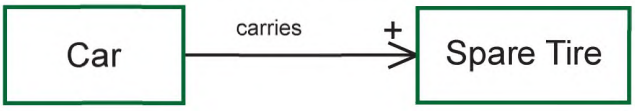
- «1 .. *» должно означать ноль или большее значение, используя для этого аннотацию в виде символа (+) около объекта, к которому он относится, с OPL-синтаксисом типа «at least one» («по крайней мере один») непосредственно перед объектом.

Примечание 1 — Символ диапазон «..» имеет два применения в терминах кратности: одно — в качестве разделителя между двумя граничными значениями, например, q_{min} .. q_{max}, с интерпретацией «до», и другое — в качестве разделителя между дополнительными значениями, например, «0 .. *», с интерпретацией «или».

Примечание 2 — При определении ограничений мощности отношений (связей) необходимо быть внимательными и применять ограничение только к рассматриваемому объекту, а не свойству этого объекта. Если объект имеет единицу измерений, то кратность должна относиться к числу отдельных единиц этой меры, например, 32 единицы объекта Water (Вода) (в мл).

В таблице 16 приведен обзор функциональных возможностей связей.

Таблица 16 — Обзор функциональных возможностей связей

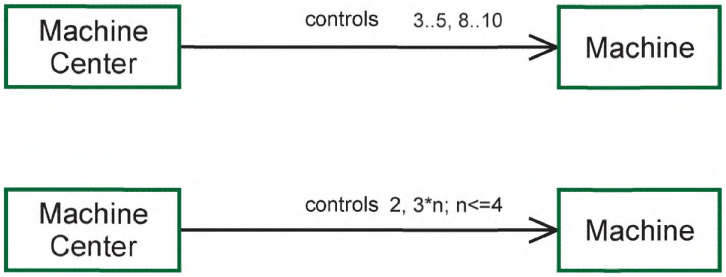
Верхняя и нижняя границы q_{min} .. q_{max}	Символ ограничения участия и OPL-фраза	Пример OPD-диаграммы и соответствующее OPL-предложение
0..1	? «на выбор»	 Объект **Car** (Автомобиль) связан с дополнительным объектом **Sunroof** (Люк в крыше) и имеет атрибут **has** (иметь)
0..*	* «на выбор» (от нуля ко многим)	 Объект **Car** (Автомобиль) связан с дополнительным объектом **Airbags** (Подушки безопасности) и имеет атрибут **is equipped with** (оснащен…)
1..1	(один к одному)	 Объект **Car** (Автомобиль) связан с объектом **Steering Wheel** (Руль управления) и имеет атрибут **is steered by** (управляется с помощью…)
1..*	+ по крайней мере один	 Объект **Car** (Автомобиль) связан по крайней мере с одним объектом **Spare Tire** (Запасная шина) и имеет атрибут **carries** (перевозит)

11.2 Выражения для кратности объекта и ограничений

Кратностью объекта может быть арифметическое выражение, в котором следует использовать символы оператора «+», «–», «*», «/», «(«и»)» с их обычной семантикой, а также обычные текстовые соотношения в соответствующих OPL-предложениях.

Целое или арифметическое выражение могут ограничивать кратность объекта. Графически выражения ограничений должны записываться после запятой, отделяющей их от выражения, которое они ограничивают, и они должны использовать символы равенства/неравенства типа «=», «<», «>», «<=», и «> =», фигурные скобки «{» и «}» для заключения в них набора элементов, а также оператор принадлежности «in» («в») (элемент, ∈), все операторы — с их обычной семантикой. Соответствующее OPL-предложение должно содержать ограничительную фразу жирным шрифтом после того объекта, к которому применяется ограничение в форме «where constraint» («где ограничение есть…»).

Пример 1 — На рисунке 32 приведены примеры кратности объекта для диапазонов и параметров.



Объект **Machine Center** (Обрабатывающий центр) связан с 3—5 или 8—10 объектами **Machine** (Станок) и имеет тег **controls** с кратностью **3..5, 8..10** (Управление 3—5, 8—10 станками).

Объект **Machine Center** (Обрабатывающий центр) связан с 2, 3*n объектами **Machine** (Станок) и имеет тег **controls** с кратностью **2**, **3*n; n<=4** (Управление 2, 3*n станками; где n<=4).

Рисунок 32 — Примеры кратности объекта с диапазонами и параметрами

Пример 2 — На рисунке 33 представлены модели системы (процесса) Blade Replacing (Замена лопаток турбины), в которой физический объект Jet Engine (Реактивный двигатель) имеет b-объектов Installed Blades (Установленные лопатки). От 2 до 4 (из общего числа k) физических объектов Aviation Engine Mechanics (Механиков по авиационным двигателям) выполняют процесс Blade Replacing (Замена турбинных лопаток), для чего они используют k-физических объектов Blade Fastening Tools (Инструмент для монтажа лопаток). Кроме того, процесс Blade Replacing (Замена турбинных лопаток) выполняют 1—2 связанных с внешней средой объекта Aerospace Engineers (Авиационные инженеры). Этот процесс дает b-связанных с внешней средой объектов Dismantled Blades (Демонтированные лопатки), которые подвергаются процессу внешней среды Blade Inspecting (Контроль лопаток), который дает a-проконтролированных объектов Blade (Лопатка) в состоянии inspected (проконтролированный) (причем a не может превышать b). Этот процесс потребляет в общем b-объектов Blades (Лопатки), из которых i-объектов находятся в состоянии inspected (проконтролированный), а (b—i)-объектов — в состоянии new (новый). Любое число объектов Blades в состоянии new может быть получено с помощью связанного с внешней средой процесса Purchasing (Закупка).

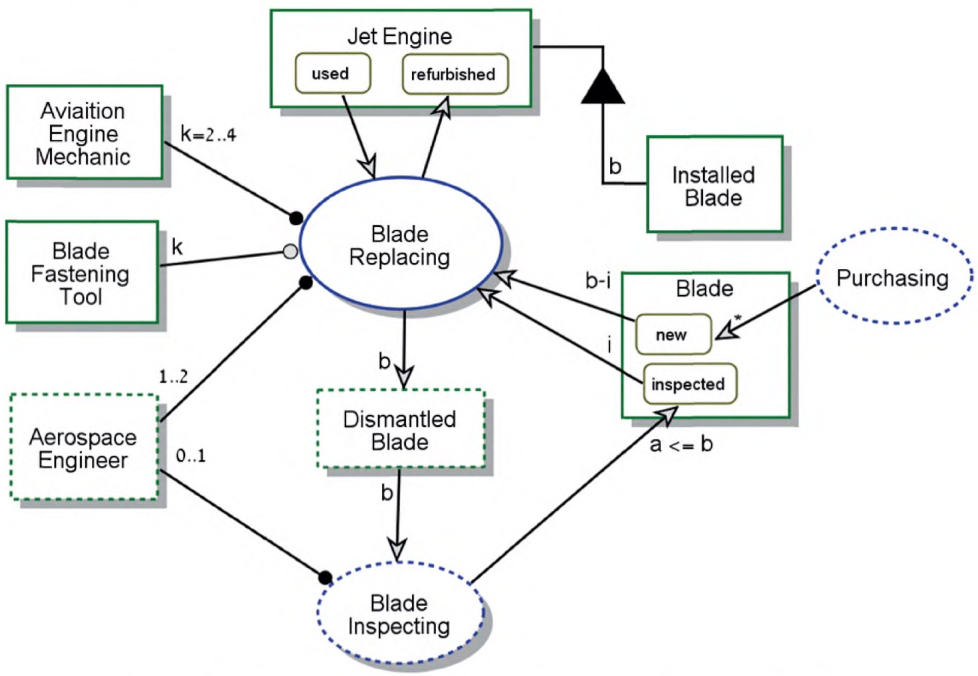


Рисунок 33, лист 1 — Диаграмма, иллюстрирующая кратность объекта: арифметические выражения и пример ограничений

k=2..4 физических объектов **Aviation Engine Mechanics** (Механиков по авиационным двигателям) выполняют процесс **Blade Replacing** (Замена турбинных лопаток).

Физический объект **Jet Engine** (Реактивный двигатель) может находиться в состоянии **used** (использованный) или **refurbished** (отремонтированный).

Физический объект **Jet Engine** (Реактивный двигатель) содержит **b**-физических объектов **Installed Blades** (Установленные лопатки).

1—2 связанных с внешней средой объектов **Aerospace Engineers** (Авиационные инженеры) выполняют процесс **Blade Replacing** (Замена турбинных лопаток).

Дополнительный объект **Aerospace Engineer** (Авиационный инженер) выполняет процесс **Blade Replacing** (Замена турбинных лопаток).

Физический объект **Blade** (Лопатка) может находиться в состоянии **inspected** (проконтролированный) или **new** (новый).

Процесс **Blade Replacing** (Замена турбинных лопаток) требует **k**-физических объектов **Blade Fastening Tools** (Инструмент для монтажа лопаток).

Процесс **Blade Replacing** (Замена турбинных лопаток) изменяет состояние физического объекта **Jet Engine** (Реактивный двигатель) с **used** (использованный) на **refurbished** (отремонтированный).

Процесс **Blade Replacing** (Замена турбинных лопаток) потребляет **i**-физических объектов **Blades** (Лопатки) в состоянии **inspected** (проконтролированный) и (**b—i**)-физических объектов **Blades** (Лопатки) в состоянии **new** (новый).

Процесс **Blade Replacing** (Замена турбинных лопаток) дает **b**-объектов внешней среды **Dismantled Blades** (Демонтированные лопатки).

Процесс внешней среды **Blade Inspecting** (Контроль лопаток) потребляет **b**-объектов **Dismantled Blades** (Демонтированные лопатки).

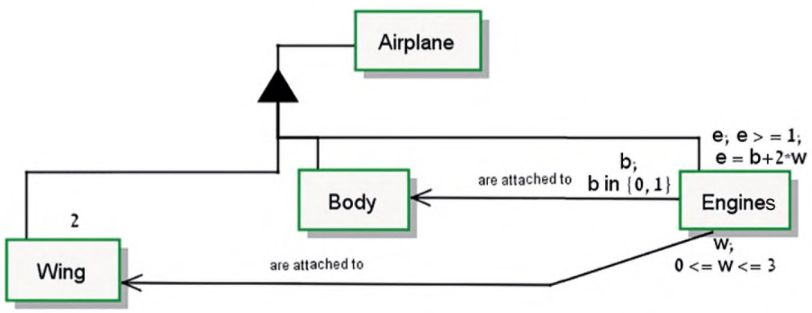
Процесс внешней среды **Blade Inspecting** дает (**a <= b**)-физических объектов **Blades** (Лопатки) в состоянии **inspected** (проконтролированный).

Внешний процесс **Purchasing** дает много физических объектов **Blades** (Лопатки) в состоянии **new** (новый).

Рисунок 33, лист 2

Если параметр кратности объекта имеет несколько ограничений, то они должны выражаться в виде разделенного точкой с запятой перечня ограничений, следующего после параметра. Любое ограничение может включать в себя любой параметр кратности объекта, появляющийся в модели. Имена параметров должны быть уникальными в рамках всей модели системы.

Пример 3 — На рисунке 34 показан способ определения параметризованных ограничений на участие на OPD-диаграмме и соответствующие OPL-предложения.



Физический объект **Airplane** состоит из физического объекта **Body** (Корпус), 2 физических объектов **Wings** (Крылья) и **e**-физических объектов **Engines** (Двигатели), где **e >= 1, e = b+2*w.**

b-физических объектов **Engines** (Двигатели) связаны с физическим объектом **Body** (Корпус) (где **b** в диапазоне **{0, 1}**), и имеют тег **are attached to** (прикреплены к ...).

w-физических объектов **Engines** (Двигатели) связаны с физическим объектом **Wing** (Крыло) (где **0 <= w <= 3**) и имеют тег **are attached to** (прикреплены к ...).

Рисунок 34 — Пример кратных параметризованных ограничений

Примечание 1 — Связь типа «агрегация — является частью» является единственной фундаментальной структурной взаимосвязью, для которой применимы ограничения по участию.

Примечание 2 — В выражении кратности процессов не используются ограничения по участию; выражение последовательного повторения одного и того же процесса использует рекуррентный процесс со счетчиком числа итераций. Параллельные синхронные или асинхронные процессы в пределах масштабированного процесса предусматривают использование других итерационных методов.

11.3 Значение атрибута и кратные ограничения

Выражение кратности объекта для структурных и процедурных связей допускает использование целых значений или символов параметров (в целочисленных значениях). Наоборот, значения, связанные с атрибутами объектов или процессов, могут быть целочисленными, действительными числами или символами параметров, которые могут быть целочисленными или действительными числами, а также символьными строками и перечислимыми значениями.

Примечание 1 — Реальные значения, содержащиеся в выражении, используют единицы измерения, которые связаны с конкретным объектом.

Графически помеченный прямоугольник с закругленными углами, находящийся внутри области атрибута, к которому он принадлежит, должен содержать значение атрибута или диапазон значений (целых чисел, вещественных чисел или символьных строк), соответствующих имени метки. В OPL-тексте значение атрибута должно выделяться жирным шрифтом без выделения заглавными буквами.

Синтаксис для объекта со значением атрибута должен выражаться в виде следующего OPL-предложения: Атрибутом **Attribute** (Атрибут) объекта **Object** (Объект) является значение **value** (единственное значение).

Синтаксис для объекта с диапазоном значений атрибута должен выражаться в виде следующего OPL-предложения: Атрибутом **Attribute** (Атрибут) объекта **Object** (Объект) является **value-range** (диапазон значений).

Примечание 2 — Диапазон значений атрибута имеет такую же экспрессивность, которая применима для кратности объекта, за исключением опциональности.

Структурная или процедурная связь, соединяющая с атрибутом, который имеет действительное численное значение, может определять ограничительное отношение, которое может отличаться от кратности объекта.

Графически ограничение значения атрибута является численной аннотацией — с помощью целого или действительного значения или символа параметра вблизи окончания атрибута для связи и расположенного вдоль направления этой связи.

12 Логические операторы AND, XOR и OR

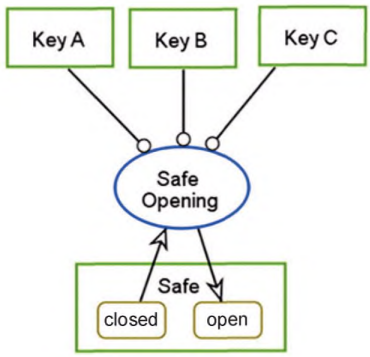
12.1 Логические процедурные связи типа AND

Группа из двух или более процедурных связей одного и того же вида, которые возникают из одного и того же процесса (или же происходят из него), и одного и того же процесса должна иметь семантику логического оператора «И» («AND»).

Графически связи с семантикой логического оператора AND не соприкасаются друг с другом на рабочем контуре.

Синтаксис связи с семантикой логического оператора AND должен выражаться фразой с использованием обозначения «and» в одном OPL-предложении, а не в виде отдельных предложений для каждой связи.

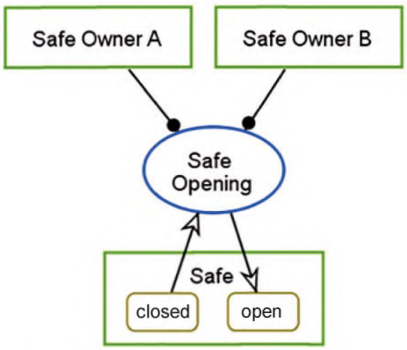
Пример 1 — Как показано в правой части рисунка 35 процесс Safe Opening (Открытие сейфа) требует объектов Safe Owner A (Владелец сейфа A) и Safe Owner B (Владелец сейфа B). В левой части рисунка 35 показано, что открытие объекта Safe (Сейф) требует наличия всех трех ключей.



Объект **Safe** (Сейф) может быть в состоянии **closed** (закрытый) или **open** (открытый).

Объект **Safe** в состоянии **open** (открытый) требует наличия объектов **Key A** (Ключ А), **Key B** (Ключ В) и **Key C** (Ключ С).

Состояние **open** (открытый) объекта **Safe** (Сейф) изменяет его состояние с **closed** (закрытое) на **open** (открытое).



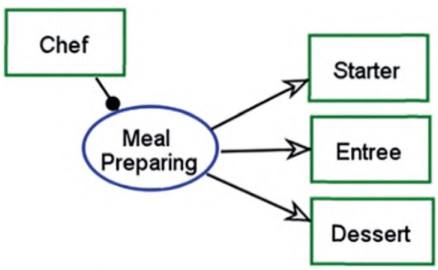
Объект **Safe** (Сейф) может быть в состоянии **closed** (закрытый) или **open** (открытый).

Объекты **Safe Owner A** (Владелец сейфа А) и **Safe Owner B** (Владелец сейфа В) управляет процессом **Safe Opening** (Открытие сейфа).

Процесс **Safe Opening** (Открытие сейфа) изменяет состояние объекта **Safe** (Сейф) с **closed** (закрытое) на **open** (открытое).

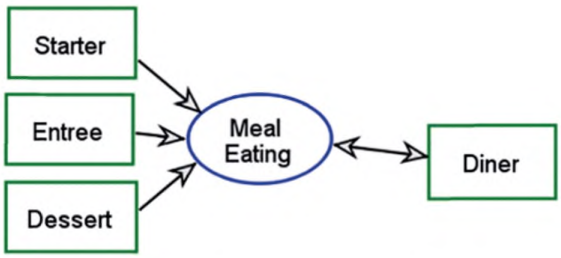
Рисунок 35 — Диаграмма логического оператора «AND» для агентских и инструментальных связей

Пример 2 — Как показано в левой части рисунка 36 процесс Meal Preparing (Приготовление пищи) дает все три блюда, а в правой части этого рисунка — процесс Meal Preparing потребляет все три блюда.



Объект **Chef** (Повар) управляет процессом **Meal Preparing** (Приготовление пищи).

Процесс **Meal Preparing** (Приготовление пищи) дает объекты **Starter** (Закуска), **Entree** (Основное блюдо) и **Dessert** (Десерт).

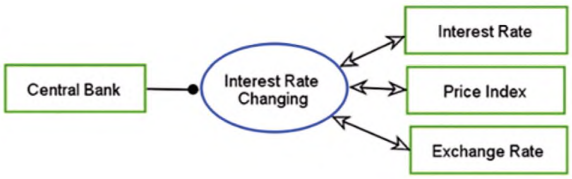


Процесс **Meal Eating** (Употребление пищи) взаимодействует с объектом **Diner** (Ресторан).

Процесс **Meal Eating** (Употребление пищи) потребляет объекты **Starter** (Закуска), **Entree** (Основное блюдо) и **Dessert** (Десерт).

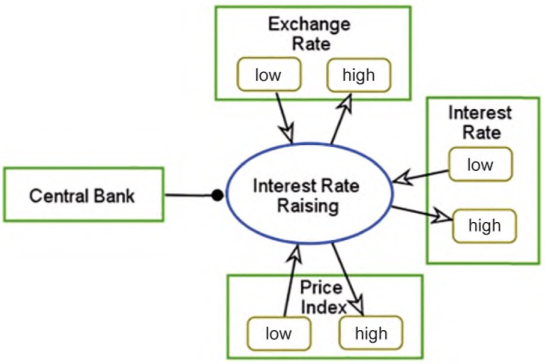
Рисунок 36 — Логическая операция «AND» для результирующих и потребительских связей

Пример 3 — На OPD-диаграмме, приведенной в левой части рисунка 37, процесс Interest Rate Changing (Изменение размера процентной ставки) взаимодействует с тремя объектами — Exchange Rate (Валютный курс), Price Index (Индекс цен) и Interest Rate (Размер процентной ставки). На OPD-диаграмме, приведенной в правой части рисунка 37 все три воздействия процесса Interest Rate Raising (Повышение размера процентной ставки) на объекты Exchange Rate (Валютный курс), Price Index (Индекс цен) и Interest Rate (Размер процентной ставки) являются определенно выраженными посредством трех пар определяющих вход/выход действующих связей.



Объект **Central Bank** (Центральный банк) управляет процессом **Interest Rate Changing** (Изменение размера процентной ставки).

Процесс **Interest Rate Changing** (Изменение размера процентной ставки) взаимодействует с объектами **Exchange Rate** (Валютный курс), **Price Index** (Индекс цен) и **Interest Rate** (Размер процентной ставки).



Объект **Central Bank** (Центральный банк) управляет процессом **Interest Rate Changing** (Изменение размера процентной ставки).

Объект **Interest Rate** (Валютный курс) может иметь состояние **high** (высокий) или **low** (низкий).

Объект **Price Index** (Индекс цен) может иметь состояние **high** (высокий) или **low** (низкий).

Объект **Exchange Rate** (Валютный курс) может иметь состояние **high** (высокий) или **low** (низкий).

Рисунок 37 — Логическая операция «AND» для воздействующей связи и пар входной/выходной связи

Примечание — Для ознакомления с влиянием меток путей на синтаксис логической операции «AND» см. раздел 13.

12.2 Логические процедурные связи типа XOR и OR

Группа из двух или более процедурных связей одного и того же вида, которые исходят из общей точки (или приходят к общей точке) на одном и том же объекте или процессе, должна быть представлена в качестве «веера» связей, конец которого должен следовать семантике логического оператора «Исключающее ИЛИ» («XOR») или логического оператора «ИЛИ» («OR») и быть сходящимся концом связи. Конечная связь, которая не является общей для всех связей, должна быть расходящимся концом связи.

Применение оператора «XOR» должно означать, что одна из сущностей на расходящемся конце так называемой «веерной» связи существует или возникает. Если на расходящемся конце связи имеются объекты, то по меньшей мере один из них будет действовать. Если на расходящемся конце связи имеются процессы, то только один из них будет выполняться.

Примечание — Подобное использование оператора XOR в OPM-методологии отличается от интерпретации некоторых бинарных XOR-операторов, где выход равен 1 — для нечетного числа входов и 0 — для четного числа входов.

Графически пунктирная дуга поперек веерных связей с центральным узлом дуги на сходящейся конечной точке контакта, должна обозначать действие XOR-оператора.

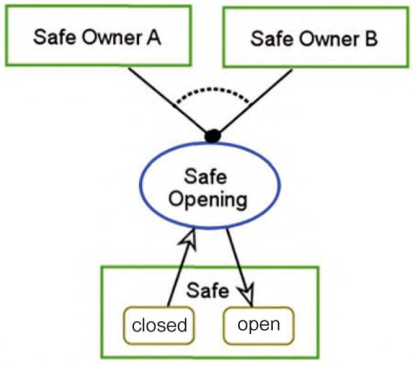
Синтаксис веерной связи n-сущностей с семантикой XOR-оператора должен выражаться одиночным OPL-предложением, содержащим фразу вида: Лишь одна из сущностей **Thing$_1$**, **Thing$_2$**, ... и **Thing$_n$**...

Применение оператора OR должно означать, что по меньшей мере одна из двух (или более) сущностей на расходящемся конце веерной связи существует или возникает. Если на расходящемся конце веерной связи имеются объекты, то по крайней мере один из них может действовать. Если на расходящемся конце веерной связи имеются процессы, то по крайней мере один из них может выполняться.

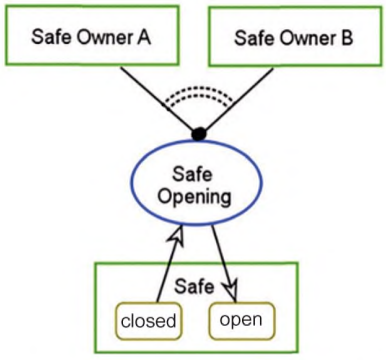
Графически две концентрические пунктирные дуги, пересекающие «веерные» связи с центральным узлом на сходящейся конечной точке контакта, должны обозначать действие логического оператора «OR».

Синтаксис веерной связи n-сущностей с семантикой логического оператора «XOR» должен выражаться одиночным OPL-предложением, содержащим фразу вида: По меньшей мере одна из сущностей **Thing$_1$**, **Thing$_2$**, ... и **Thing$_n$**...

Пример — На OPD-диаграмме, приведенной в правой части рисунка 38 и использующей XOR-оператор, только один из объектов Safe Owner A (Владелец сейфа A) или Safe Owner B (Владелец сейфа B) должен представляться, чтобы выполнялся процесс Safe Opening (Открытие сейфа). На OPD-диаграмме, приведенной в левой части рисунка и использующей OR-оператор, по меньшей мере один из объектов Safe Owner A и Safe Owner B должен быть представлен, чтобы процесс Safe Opening выполнялся. Веерная связь на обеих OPD-диаграммах является сходящейся и состоит из двух агентских связей.



Лишь только один из объектов **Safe Owner A** (Владелец сейфа A) и **Safe Owner B** (Владелец сейфа B) управляет процессом **Safe Opening** (Открытие сейфа).
Состояние **closed** (закрытый) объекта **Safe** (Сейф).



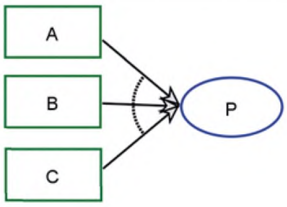
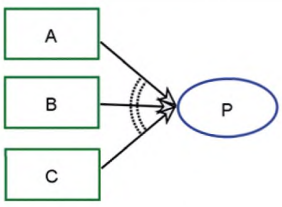
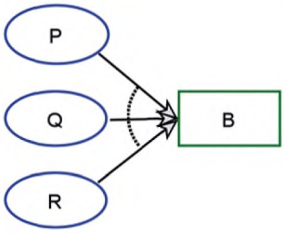
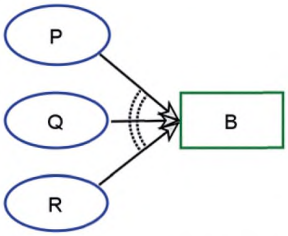
По меньшей мере один из объектов **Safe Owner A** (Владелец сейфа A) и **Safe Owner B** (Владелец сейфа B) управляет процессом **Safe Opening** (Открытие сейфа).
Состояние **closed** (закрытый) объекта **Safe** (Сейф).

Рисунок 38 — Примеры агентской связи для логического оператора OR (слева) и логического оператора XOR (справа)

12.3 Расходящиеся и сходящиеся XOR- и OR-связи

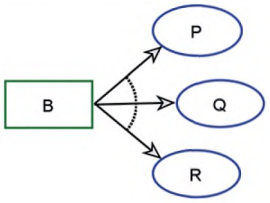
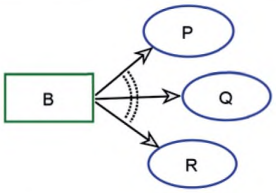
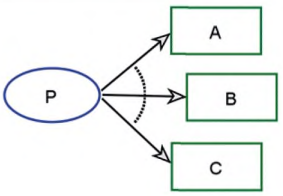
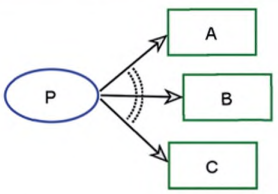
В таблице 17 показано, что если исходными сущностями являются объекты, а сущностью получателя является процесс, то потребительская веерная связь будет сходящейся, тогда как если исходными сущностями являются процессы, а сущностью получателя является объект, то результирующая веерная связь также будет сходящейся.

Таблица 17 — Обзор потребительских и результирующих сходящихся XOR- и OR-связей

	Логический оператор XOR	Логический оператор OR
Сходящаяся потребительская веерная связь	 Процесс **P** потребляет только один из объектов **A, B** или **C**	 Процесс **P** потребляет по меньшей мере один из объектов **A, B** или **C**
Сходящаяся результирующая веерная связь	 Только один из процессов **P, Q** или **R** производит объект **B**	 По меньшей мере один из процессов **P, Q** или **R** производит объект **B**

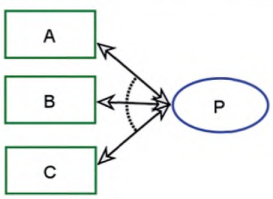
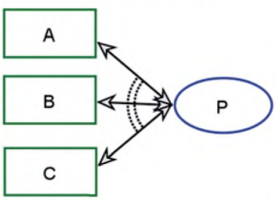
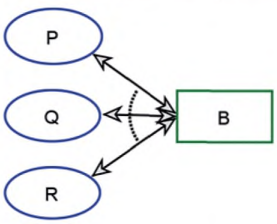
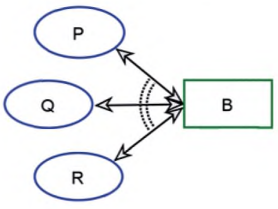
В таблице 18 показано, что если исходной сущностью является объект, а сущностями получателя являются процессы, то потребительская веерная связь будет расходящейся, а если исходная сущность является процессом, а сущностями получателя являются объекты, то результирующая веерная связь также будет расходящейся.

Таблица 18 — Обзор потребительских и результирующих расходящихся XOR- и OR-связей

	Логический оператор XOR	Логический оператор OR
Расходящаяся потребительская веерная связь	 Только один из процессов **P, Q** или **R** потребляет объект **B**	 По меньшей мере один из процессов **P, Q** или **R** потребляет объект **B**
Расходящаяся результирующая веерная связь	 Процесс **P** порождает только один из объектов **A, B** или **C**	 Процесс **P** порождает по меньшей мере один из объектов **A, B** или **C**

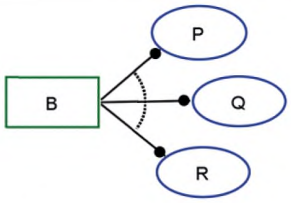
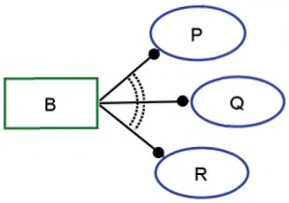
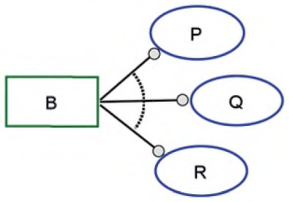
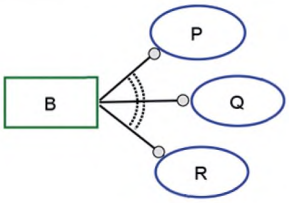
Поскольку воздействующая связь (effect link) является двунаправленной, сущности, связанные воздействующей веерной связью, являются одновременно и источником, и получателем, аннулируют определения сходящейся и расходящейся веерных связей. Вместо этого, как видно из таблицы 19, должны возникать расхождения относительно множества объектов или множества процессов, которые соединяются этой веерной связью.

Таблица 19 — Обзор воздействующих веерных XOR- и OR-связей

	Логический оператор XOR	Логический оператор OR
Многообъектная воздействующая веерная связь	 Процесс **P** взаимодействует только с одним из объектов **A**, **B** или **C**	 Процесс **P** взаимодействует по меньшей мере с одним из объектов **A**, **B** или **C**
Многопроцессная воздействующая веерная связь	 Только один из процессов **P**, **Q** или **R** взаимодействует с объектом **B**	 По крайней мере один из процессов **P**, **Q** или **R** взаимодействует с объектом **B**

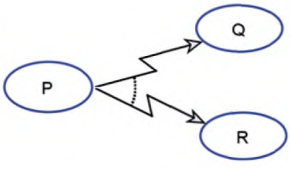
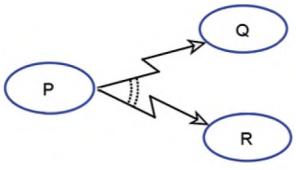
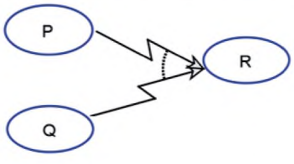
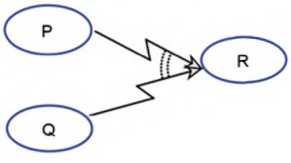
Поскольку *enabler* — это объект (см. таблица 20), и агентская, и инструментальная веерные связи должны быть расходящимися, с несколькими процессами в качестве целевых.

Таблица 20 — Обзор агентских и инструментальных веерных связей

	Логический оператор XOR	Логический оператор OR
Агентская веерная связь	 Объект **B** имеет дело только с одним из процессов **P**, **Q** или **R**	 Объект **B** имеет дело по меньшей мере с одним из процессов **P**, **Q** или **R**
Инструментальная веерная связь	 Только один из процессов **P**, **Q** или **R** требует объекта **B**	 По меньшей части один из процессов **P**, **Q** или **R** требует объект **B**

Инициализирующие веерные связи (invocation link) могут быть расходящимися или сходящимися для логических операторов XOR и OR (см. таблица 21).

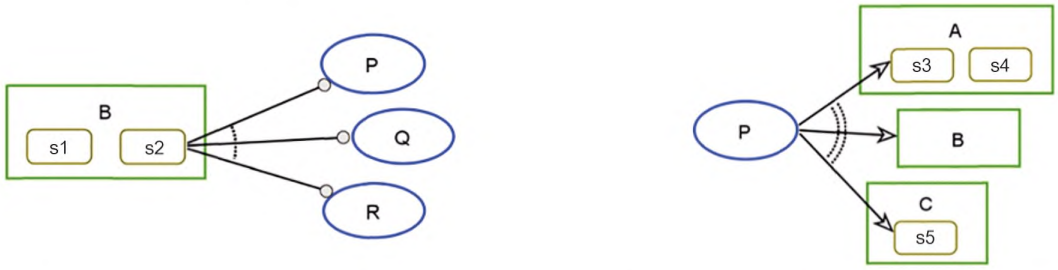
Таблица 21 — Обзор инициализирующих веерных связей

	Логический оператор XOR	Логический оператор OR
Расходящаяся веерная инициализирующая связь	 Процесс **P** инициирует только один из процессов **Q** или **R**	 Процесс **P** инициирует по меньшей мере один из процессов **Q** или **R**
Сходящаяся веерная инициализирующая связь	 Только один из процессов **P** или **Q** инициирует процесс **R**	 По меньшей мере один из процессов **P** или **Q** инициирует процесс **R**

12.4 Определяющие состояние веерные XOR- и OR-связи

Каждая из веерных связей (см. 12.3) должна иметь соответствующий определяющий состояние вариант, при котором источник и получатель могут быть специфическими состояниями объекта или объектами без спецификации состояния. Могут иметь место различные сочетания определяющих состояние связей и связей без состояний как источников и получателей веерных связей.

Пример — В левой части рисунка 39 приведена определяющая состояние веерная инструментальная XOR-связь, а в правой части этого рисунка — смешанная результирующая веерная OR-связь, где связи для объектов A и C (но не для объекта B) являются определяющими состояние.



Только один из процессов **P, Q** или **R** требует состояния **s2** объекта **B**.

Процесс **P** создает состояние **s3** объекта **A,** состояние **s5** объекта **C** и объект **B**.

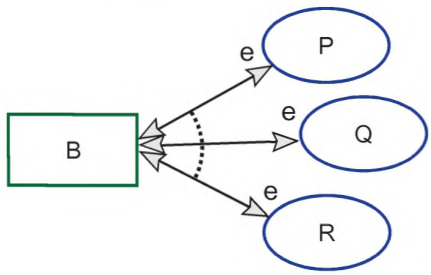
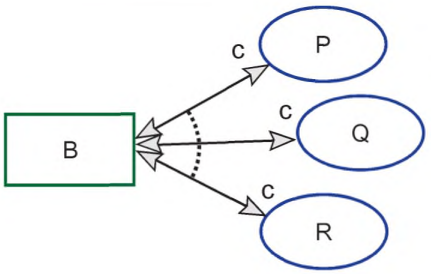
Рисунок 39 — Примеры определяющих состояние XOR- и OR-связей

12.5 Модифицирующие управление веерные связи

Каждая из веерных XOR-потребительских, результирующих, воздействующих и разрешающих связей должна иметь соответствующую модифицирующую управление веерную связь: ситуационную веерную связь (event link fan) и условную веерную связь (condition link fan).

В таблице 22 представлены ситуационные и условные воздействующие веерные связи, рассматриваемые как представители основного варианта (не определяющего состояние) модифицированных веерных связей.

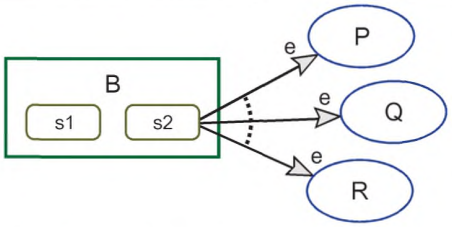
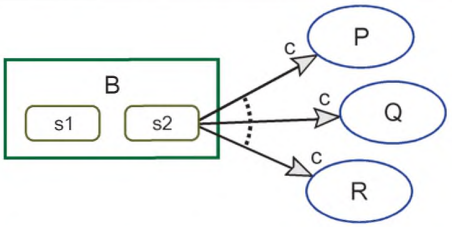
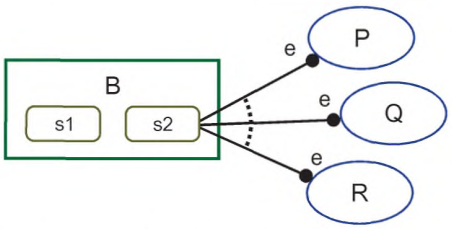
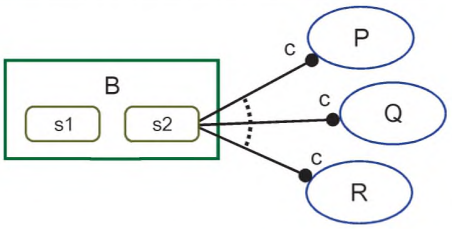
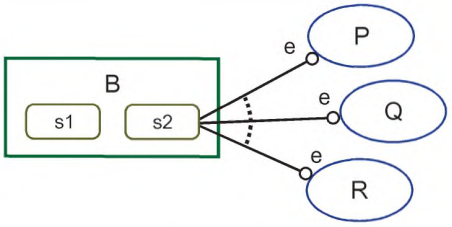
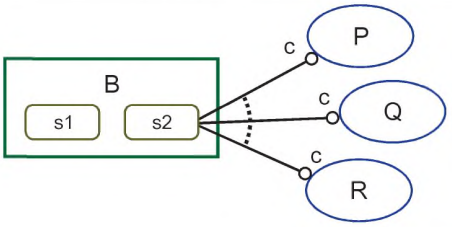
Таблица 22 — Ситуационные и условные воздействующие веерные связи

Ситуационные связи	Условные связи
	
Объект **B** инициирует только один из процессов **P**, **Q** или **R;** только в этом случае возникающий процесс будет воздействовать на объект **B**	Только один из процессов **P**, **Q** или **R** возникает, если объект **B** существует; только в этом случае возникающий процесс воздействует на объект **B;** в противном случае эти процессы должны пропускаться

12.6 Определяющие состояние и модифицирующие управление веерные связи

Каждая из модифицирующих управление веерных связей (control-modified link fans), за исключением модифицирующей управление воздействующей веерной связи (control-modified effect link fan), должна иметь соответствующую определяющую состояние и модифицирующую управление веерную связь (state-specified control-modified link fan). Поскольку эти определяющие состояние версии связей являются более сложными, чем соответствующие не определяющие состояние связи, в таблице 23 представлены OPD-диаграммы и OPL-синтаксис определяющих состояние и соответствующих не определяющих состояние версий связей.

Таблица 23 — Определяющие состояние и не имеющие состояния веерные связи с модифицированным управлением

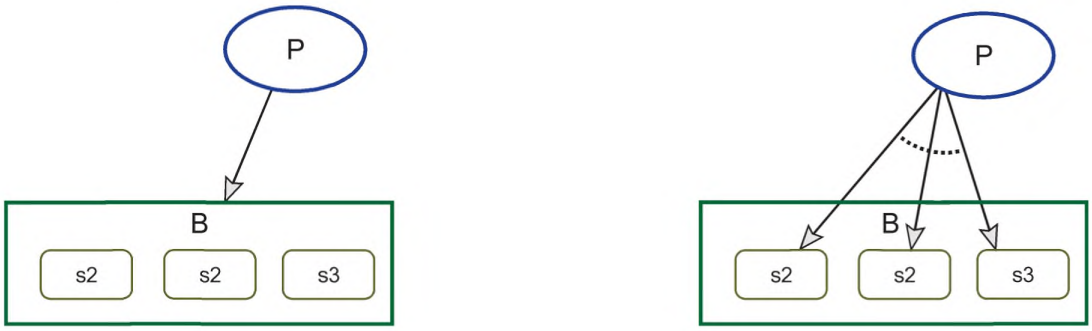
	Модификатор управления событиями	Модификатор управления состояниями
Потребительская веерная связь	 Состояние **s2** объекта **B** инициирует только один из процессов **P**, **Q** или **R**, который будет потреблять объект **B**. *В случае отсутствия определенного внутреннего состояния:* Объект **B** инициирует только один из процессов **P**, **Q** или **R**, который потребляет объект **B**	 Только один из процессов **P, Q** или **R** возникает, если объект **B** находится в состоянии **s2;** только в этом случае возникающий процесс будет потреблять объект **B;** в противном случае эти процессы будут пропускаться. *В случае отсутствия определенного внутреннего состояния:* Только один из процессов **P, Q** или **R** возникает, если объект **B** существует; только в этом случае возникающий процесс будет потреблять объект **B;** в противном случае эти процессы будут пропускаться
Агентская веерная связь	 Состояние **s2** объекта **B** инициирует и управляет только процессом **P**, **Q** или **R.** *В случае отсутствия определенного внутреннего состояния:* Объект **B** определяет и управляет только одним процессом **P**, **Q** или **R**	 Объект **B** управляет только одним состоянием **P**, **Q** или **R,** если объект **B** находится в состоянии **s2;** в противном случае эти процессы будут пропускаться. *В случае отсутствия определенного внутреннего состояния:* Объект **B** управляет только одним из процессов **P**, **Q** или **R,** если объект существует **B;** в противном случае эти процессы будут пропускаться
Инструментальная веерная связь	 Состояние **s2** объекта **B** инициирует только один из процессов **P**, **Q** или **R**, которые требуются состоянием **s2** объекта **B.** *В случае отсутствия определенного внутреннего состояния:* Объект **B** инициирует только один из процессов **P**, **Q** или **R**, который требует объект **B**	 Только один из процессов **P**, **Q** или **R** необходим, если объект **B** находится в состоянии **s2;** в противном случае эти процессы будут пропускаться. *В случае отсутствия определенного внутреннего состояния:* Только один из процессов **P**, **Q** или **R** требует, чтобы объект **B** существовал; в противном случае эти процессы будут пропускаться

Каждая веерная XOR-связь, представленная в таблицах 22—23, должна иметь собственный OR-эквивалент (обозначаемый двойной пунктирной дугой), с соответствующим OPL-предложением, в котором служебная фраза «по крайней мере» (at least) заменяется на фразу «в точности» (exactly).

12.7 Вероятности связей и вероятностные веерные связи

Процесс **P** с результирующей связью, которая создает структурированный объект **B** с n-состояниями $\mathbf{s_1}$—$\mathbf{s_n}$ (без спецификации конкретного состояния) должен означать, что вероятность формирования объекта **B** в любом конкретном состоянии должна составлять 1/n. В этом случае единственная результирующая связь с объектом должна заменять результирующую веерную связь в каждом ее состоянии.

Пример 1 — На левой OPD-диаграмме (см. рисунок 40) результирующая связь между процессом P и объектом B, имеющим три состояния, будет означать, что процесс P может создавать каждое состояние объекта B с равной вероятностью Pr = 1/3. Правая OPD-диаграмма на рисунке 40 иллюстрирует более сложный способ выражения той же самой ситуации.



Объект **B** может находиться в состоянии **s1, s2** или **s3.**
Процесс **P** создает объект **B.**

Объект **B** может находиться в состоянии **s1, s2** или **s3.**
Процесс **P** создает состояние **s1**, **s2** или **s3** объекта **B.**

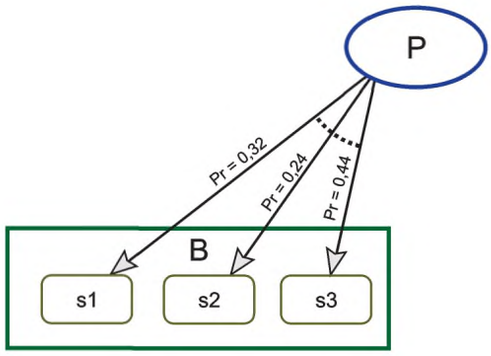
Рисунок 40 — Эквивалентность результирующей связи и совокупности определяющих состояние результирующих XOR-связей

В общем случае вероятности осуществления конкретной связи в веерной связи не равны. Вероятность связи может быть значением свойства, присвоенного связи в расходящейся веерной XOR-связи, которое определяет вероятность осуществления конкретной связи в веерной связи. Вероятностная веерная связь должна быть веерной связью с аннотациями на каждой связи с указанием значений вероятности, причем сумма всех вероятностей должна быть в точности равна 1.

Графически вдоль каждой веерной связи с вероятностными свойствами должны находиться аннотации в виде Pr = p, где p — это численное значение вероятности или параметр, которые должны обозначать вероятность выполнения системой выбора и осуществления конкретной веерной связи.

Соответствующим OPL-предложением должно быть предложение для расходящейся веерной XOR-связи (без вероятности для связей), исключая фразу «точно один из ...» (exactly one of...) и включая фразу «... с вероятностью p» (...with probability p) — после каждого имеющегося имени сущности (с аннотацией вероятности «Pr = p»).

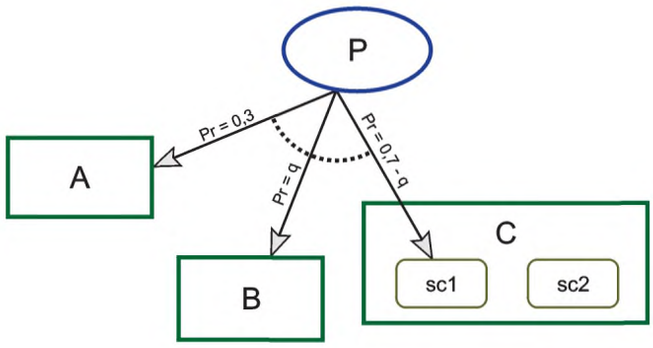
Пример 2 — На рисунке 41 представлены два примера создания вероятностных определяющих состояние объектов и их детерминированных аналогов. На OPD-диаграмме слева показан процесс P, который может создавать объект B в трех возможных состояниях — s1, s2 или s3 с соответствующими им вероятностями 0,32, 0,24 и 0,44, которые указаны вдоль каждой из результирующих веерных связей. На OPD-диаграмме справа показан процесс P, который может создавать один из объектов A, B или C в состоянии sc1 с вероятностями, указанными вдоль каждой из результирующих веерных связей.



Процесс **P** создает объект **B** в состоянии **s1** с вероятностью **0,32**, объект **B** в состоянии **s2** с вероятностью **0,24** или объект **B** в состоянии **s3** с вероятностью **0,44**.

Аналогичный детерминированный случай:

Процесс **P** создает объект **B** только в состоянии **s1**, **s2** или **s3**.



Процесс **P** создает объект **A** с вероятностью **0,3**, объект **B** с вероятностью **q** или объект **C** в состоянии **sc1** с вероятностью **0,7-q**.

Аналогичный детерминированный случай:

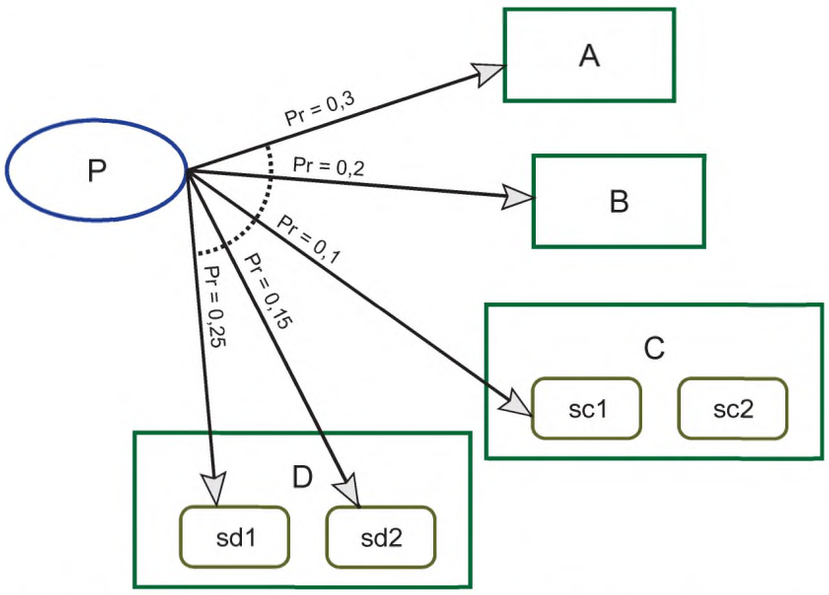
Процесс **P** создает только объект **A**, **B** или **C** в состоянии **sc1**.

Рисунок 41 — Примеры создания вероятностного определяющего состояние объекта

Для процесса **P** с результирующей связью, создающего структурированный объект с состояниями $\mathbf{s_1}$ — $\mathbf{s_n}$ и с начальным состоянием $\mathbf{s_i}$, процесс будет создавать объект **B** в состоянии $\mathbf{s_i}$ с вероятностью 1,0, однако если объект **B** имеет m-начальных состояний (m < n), то процесс **P** будет создавать объект **B** в одном из начальных состояний с вероятностью 1/m.

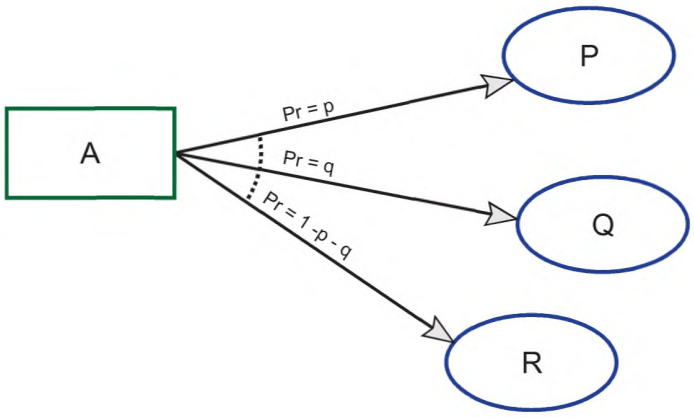
Для вероятностной веерной результирующей связи любой из объектов типа *resultees* может находиться в указанном состоянии (или не в нем). Для всех веерных связей, содержащих процедурные связи других видов (в том числе с модификаторами событий и состояний), где целями веерных связей являются процессы, источник может быть объектом или его заданным состоянием.

Пример 3 — OPD-диаграмма на верхней части рисунка 42 показывает вероятностную результирующую связь, с помощью которой процесс P создает объект A, B или C в состоянии sc1 или объект D — в состоянии sd1 или sd2. OPD-диаграмма в средней части рисунка 42 показывает вероятностную потребительскую веерную связь, с помощью которой объект A потребляется, с определенными вероятностями для процесса P, Q или R. OPD-диаграмма в нижней части рисунка 42 представляет дополнительный факт того, что объекту A необходимо находиться в состоянии s2.

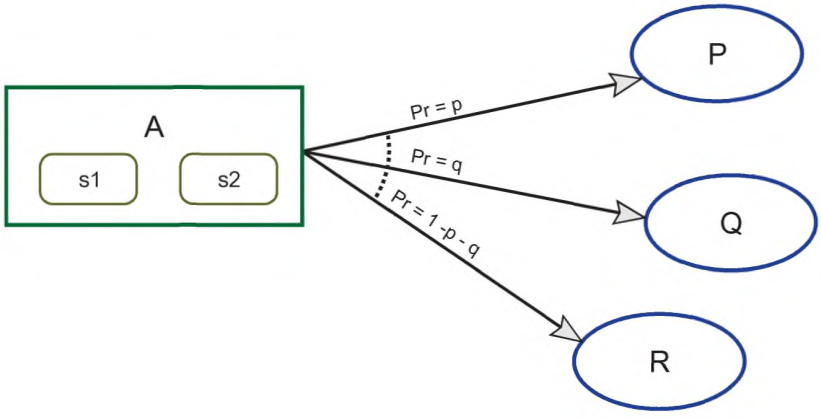


Процесс **P** создает объект **A** с вероятностью **0,3**, объект **B** — с вероятностью **0,2**, состояние **sc1** объекта **C** — с вероятностью **0,1**, состояние **sd1** объекта **D** — с вероятностью **0,25**, состояние **sd2** объекта **D** — с вероятностью **0,15**.

Рисунок 42, лист 1 — Объекты, имеющие или не имеющие определенных состояний, как источники и получатели вероятностной веерной связи



Объект **A** потребляется процессом **P** с вероятностью **p**, процессом **Q** — с вероятностью **q** или процессом **R** — с вероятностью **1-p-q**.



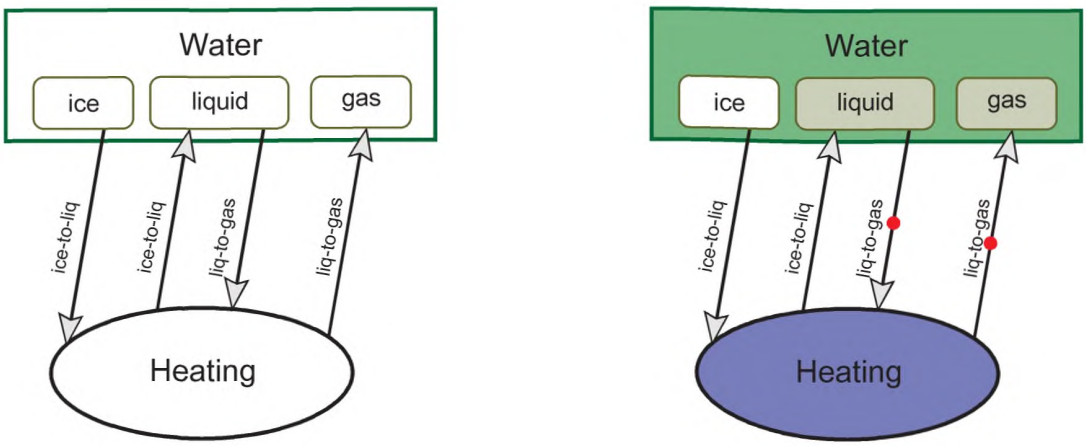
Объект **A** в состоянии **s2** потребляется процессом **P** с вероятностью **p**, процессом **Q** — с вероятностью **q** или процессом **R** — с вероятностью **1-p-q**.

Рисунок 42, лист 2

13 Путь выполнения и метки путей

Метка пути должна быть свойством связи и соответствующей аннотацией, согласующей пару процедурных связей. Если предварительные условия для процесса содержат связи с метками путей, а набор апостериорно обработанных объектов имеет более чем одну возможность для объекта назначения, то последний набор должен получаться с помощью связи с той же меткой пути, что и используемой для набора предварительно обработанных объектов.

Пример 1 — На рисунке 43 имеются две выходные связи: одна от процесса Heating (Нагрев) до состояния liquid (жидкость) объекта Water (Вода), а другая — до состояния gas (газ). При переходе процесса Heating из состояния ice (лед) остается неясным, будет ли конечным результатом состояние liquid или gas. Метки пути вдоль процедурных связей позволяют решать дилемму относительно однозначного определения соответствующей связи на выходе процесса, как это было показано путем динамического моделирования (см. левую диаграмму).



Объект **Water** (Вода) может находиться в состоянии **ice** (лед), **liquid** (жидкость) или **gas** (газ).

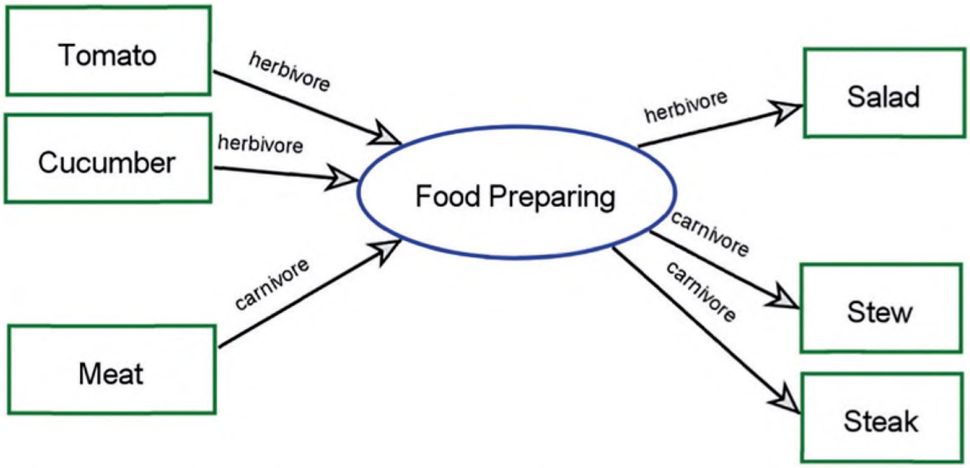
Следуя пути **ice-to-liq** (лед-жидкость), процесс **Heating** (Нагрев) изменяет состояние объекта **Water (Вода)** из состояния **ice** (лед) на состояние **liquid** (жидкость).

Следуя пути **liq-to-gas** (жидкость-газ), процесс **Heating** (Нагрев) изменяет состояние объекта **Water** (Вода) из состояния **liquid** (жидкость) на состояние **gas** (газ).

Рисунок 43 — Пути выполнения и метки путей

Примечание — Метка пути — это метка на процедурной связи, которая устраняет неопределенность (возникающую из-за множественности исходящих процедурных связей) путем определения связи с такой же меткой, что и у инициализирующего процесса.

Пример 2 — Рисунок 44 иллюстрирует использование меток путей на потребительских и воздействующих связях с последующим OPL-разделом.



Следуя пути **carnivore** (мясоедство), процесс **Food Preparing** (Приготовление пищи) потребляет объект **Meat** (Мясо).

Следуя пути **herbivore** (вегетарианство), процесс **Food Preparing** (Приготовление пищи) потребляет объекты **Cucumber** (Огурец) и **Tomato** (Помидор).

Следуя пути **carnivore** (мясоедство), процесс **Food Preparing** (Приготовление пищи) создает объекты **Stew** (Жаркое) и **Steak** (Стейк).

Следуя пути **herbivore** (вегетарианство), процесс **Food Preparing** (Приготовление пищи) создает объект **Salad** (Салат).

Рисунок 44 — Метки путей, иллюстрирующие потребительские или результирующие связи

14 Управление контекстом с использованием OPM-методологии

14.1 Разработка структурной диаграммы

Определение целей системы, области ее применения и функций с точки зрения границ, заинтересованных сторон, предпосылок и постусловий является основанием для определения целесообразности появления в модели других элементов, в том числе сущностей внешней среды.

Структурная диаграмма (SD) должна быть OPD-диаграммой, моделирующей:

- поведение заинтересованных сторон, в частности, бенефициаров;
- процесс передачи функциональной ценности, которую бенефициар ожидает получить;
- другие сущности внешней среды и системные сущности, необходимые для создания соответствующего краткого OPL-раздела.

Соответствующий OPL-раздел должен обеспечивать ситуационный контекст, необходимый для функционирования системы.

Способ выражения функциональной ценности может быть:

- явным, путем идентификации состояний исходного источника и получателя для бенефициара или начальных и конечных значений одного или нескольких из его атрибутов, или
- неявным, путем указания того, что бенефициар зависит от функции системы.

Структурная диаграмма должна содержать только основные, наиболее важные сущности, т. е. те сущности, которые необходимы для понимания функционирования и контекста системы. Разработчик модели должен использовать OPM-методы уточнения для постепенного выявления деталей, которые связаны с сущностями, которые составляют контент структурной диаграммы (SD).

Пример — На объекте Manufacturing Facility (Промышленное предприятие) объект Beneficiary (Бенефициар) разработал и развернул объект Preventive Maintenance System (Система профилактического технического обслуживания). Системная функция Preventive Maintenance Executing (Выполнение профилактического технического обслуживания) изменяет состояние атрибута Downtime (Простой) объекта Manufacturing Facility из состояния «high» (высокое) на состояние «low» (низкое). Подобное изменение добавляет функциональную ценность объекту Manufacturing Facility, поскольку оно будет обладать большим временем исправного состояния для изготовления продукции и повышенным объемом продаж/доходов за счет инвестиций в разработку и функционирование объекта Preventive Maintenance System.

14.2 Достижение надлежащего понимания модели

14.2.1 OPM-методы конкретизации-обобщения

OPM-методология обеспечивает методы конкретизации и обобщения (abstracting and refining), которые предназначены для управления выражением ясности и полноты модели. Эти методы позволяют определять контекстуализированные сегменты модели в виде отдельных, но взаимосвязанных OPD-диаграмм, которые в своей совокупности представляют модель функциональной ценности системы обеспечения. Эти методы должны обеспечивать представление и просмотр моделируемой системы, а также элементов, содержащихся в ней в различных контекстах, которые взаимосвязаны с общими объектами, процессами и отношениями. Набор четко определенных и совместимых между собой OPD-диаграмм должен полностью определять всю систему (до определенной степени детализации) и обеспечивать всестороннее представление этой системы с соответствующим текстовым описанием модели на OPD-диаграмме.

OPM-методами конкретизации-обобщения (refinement-abstraction) должны быть следующие три пары методов: отображение/маскирование состояний (state expression and suppression), разворачивание/сворачивание (unfolding and folding), детализация/обобщение (in-zooming and out-zooming).

14.2.1.1 Отображение/маскирование состояний

Слишком подробное отображение состояний объекта на OPD-диаграмме может приводить к диаграмме, которая будет слишком насыщенной или загроможденной, что сделает ее трудно воспринимаемой или читаемой.

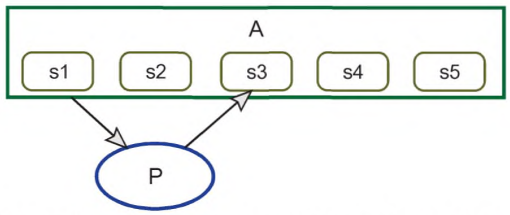
OPM-методология должна давать возможность маскирования (сокрытия) части состояний, которые отображаются на конкретной OPD-диаграмме, если эти состояния не требуются для понимания контекста данной диаграммы.

Обратная операция для операции маскирования состояний — это операция их отображения, которая дает информацию о возможных состояниях объекта. OPL-язык, соответствующий OPD-диаграмме, выражает состояние объектов в соответствии с этой диаграммой.

В OPM-методологии разработчик модели может маскировать любое подмножество состояний, однако полный набор объектных состояний для объекта должен быть равен сумме состояний того же объекта, проявляющихся на всех OPD-диаграммах полной OPM-модели.

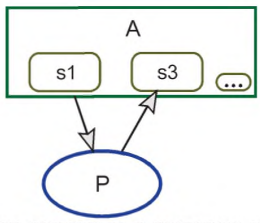
Графически аннотация, указывающая, что объект представляет собственное подмножество (т. е. по меньшей мере одно, но не все) своих состояний, должна быть в виде небольшого символа маскирования, приводимого вблизи нижнего правого угла объекта в виде маленького эллипса с многоточием. Этот символ будет означать существование одного или нескольких состояний, чье отображение скрывается. Текстуальная эквивалентность символа маскирования состояния должна быть зарезервирована за фразой типа «или другие состояния» (or other states).

Пример — На рисунке 45 слева показан структурированный объект со всеми отображенными состояниями, а также аналогичный объект с частью маскированных состояний (справа).



Объект **A** может находиться в состоянии **s1, s2, s3, s4** или **s5**.

Процесс **P** изменяет состояние объекта **A** из состояния **s1** на состояние **s3**.



Объект **A** может находиться в состоянии **s1, s3** или в других состояниях.

Процесс **P** изменяет состояние объекта **A** из состояния **s1** на состояние **s3**.

Рисунок 45 — Диаграмма объекта с внутренним состоянием со всеми отображаемыми состояниями (слева) и тот же объект с частично отображаемыми состояниями (справа)

14.2.1.2 Разворачивание/сворачивание

Разворачиванием должен быть метод конкретизации, проработки или декомпозиции, позволяющий выявлять набор сущностей типа *refineable*, которые связаны с разворачиваемыми сущностями типа *refineable*. Результатом этого должно стать создание иерархического древа, корнем которого должна быть разворачиваемая сущность, а связями с этим корнем должны быть сущности, которые представляют собой детализацию развернутой сущности.

С другой стороны, сворачиванием должен быть метод обобщения или композиции, который следует применять к развернутому иерархическому древу. Оно должно обеспечивать маскирование набора развернутых сущностей, оставляя при этом только корень древа.

К каждой из четырех основных структурных реляционных связей (structural relation links) могут применяться операции разворачивания/сворачивания для получения следующих четырех пар операций:

- разворачивание группы (aggregation unfolding) — раскрытие части целого, и сворачивание частей — сокрытие части целого;
- разворачивание представителей (exhibition unfolding) — раскрытие свойств представителя, и сворачивание характеристик — маскирование свойств представителя;
- разворачивание обобщений (generalization unfolding) — раскрытие специализаций общего, и сворачивание специализаций — маскирование специализаций общего;
- разворачивание классификации (classification unfolding) — раскрытие экземпляров класса, и инстанцирование-сворачивание — маскирование экземпляров класса.

Разворачивание на диаграмме должно происходить тогда, когда сущность *refineable* и ее объекты *refinees* оказываются развернутыми на той же OPD-диаграмме. Поскольку при разворачивании используются основные структурные связи, разворачивание на диаграмме графически, синтаксически и семантически эквивалентно использованию основных структурных связей.

Разворачивание на новой диаграмме должно происходить тогда, когда сущность *refineable* и ее объекты типа *refinees* оказываются развернутыми на новой OPD-диаграмме.

Графически сущность *refineable* должна изображаться фигурой с утолщенным контуром, как и на обобщенной OPD-диаграмме, на которой эта сущность оказывается свернутой без объектов типа *refinees*, а в новом более детальном OPD-контексте, в котором сущность *refineable* проявляется в развернутом виде и соединяется со своими объектами типа *refinees*, одной или несколькими фундаментальными структурными связями.

Соответствующее OPL-предложение для новой OPD-диаграммы, на которой сущность *refineable* имеет n-объектов типа *refinees*, должно иметь следующий вид: Сущность **Refineable** развертывается в сущности **Refinee$_1$**, **Refinee$_2$**, ... и **Refinee$_n$**.

Примечание 1 — Разворачивание может быть более точно определено как разворачивание части (part-unfolding), разворачивание признака (feature-unfolding), разворачивание специализации (specialization-unfolding) и разворачивание экземпляра (instance-unfolding) (см. А.4.7.2).

Разработчик модели может принимать решение о том, использовать операцию разворачивания на существующей диаграмме или на новой диаграмме, принимая во внимание компромисс между определенной скученностью элементов на существующей OPD-диаграмме и необходимостью создания новой OPD-диаграммы для отображения объектов типа *refinees* и соответствующих связей между ними.

Примечание 2 — Разворачивание часто выполняют в виде сочетания разворачивания существующей и новой диаграмм для представления детализирующих или декомпозирующих ситуаций.

Примечание 3 — Частичное разворачивание можно изображать таким же образом, как и частичную фундаментальную структурную реляционную связь.

Для удовлетворения конкретного контекстного соответствия для OPD-диаграммы разработчик модели может выбирать, какие объекты типа *refinees* будут проявляться в развернутом виде. После бимодального представления OPM-методологии, OPL-язык, соответствующий OPD-диаграмме, должен выражать только те объекты типа *refinees*, которые приводятся на данной OPD-диаграмме.

Примечание 4 — Частичное сворачивание эквивалентно частичному разворачиванию, где совокупность множеств отображаемых и скрытых объектов типа *refinee* взаимно дополняют друг друга.

Примечание 5 — Разворачивание позволяет выявлять более детальные структурные, а не поведенческие детали, т. е. никакой передачи контроля исполнения при этом не происходит (см. 14.2.2). Тем не менее иерархические зависимости, включая процедурные связи, могут приводить к изменениям поведения, которые связаны с использованием развернутой сущности.

14.2.1.3 Детализация/обобщение

Детализацией внутрь (in-zooming) должно быть особого рода разворачивание, которое сочетает в себе связи типа «агрегация — является частью» (aggregation-participation) и «представление-характеризация» (exhibition-characterization) с дополнительной семантикой. В случае процессов детализация позволяет моделировать подпроцессы, их временной порядок, их взаимодействие с объектами и передачу контроля исполнения с учетом контекста. В случае объектов детализация позволяет создавать отдельный контекст для моделирования пространственного или логического порядка элементов объектов.

Графически для существующей детализирующей внутрь (in-diagram) и новой (new-diagram) диаграмм процессов эллипс для сущности типа *refineable* должен иметь увеличенный размер для размещения в нем символов объектов типа *refinees* и связей между ними, находящихся в пределах детализированного контекста. В случае новой детализированной диаграммы фигура для сущности типа *refineable* должна иметь утолщенный контур, как и на обобщенной OPD-диаграмме, на которой сущность типа *refineable* проявляется без объектов типа *refinees*, и в новом детализированном OPD-контексте, где сущность типа *refineable* проявляется в окружении объектов-подпроцессов типа *refinees* и сопутствующих объектов.

Соответствующее OPL-предложение для детализированного процесса должно иметь следующий вид: Процесс **Process** (Процесс) распадается на подпроцессы **Subprocess A**, **Subprocess B** и **Subprocess C** в указанной последовательности.

Примечание 1 — Детализацию можно более точно определять путем указания обобщенного OPD-имени и более детального OPD-имени (см. А.4.7.4).

Контекст детализированного процесса должен включать подпроцессы, которые являются частями этого процесса, и, возможно, внутренние объекты, которые являются атрибутами детализированного процесса. Контекстуальной областью применения детализированного процесса должны быть сущность типа *refineable*, его подпроцессы, атрибуты и связи, изображенные на OPD-диаграмме.

График выполнения в контексте детализированного процесса должен проходить от верхней части ее расширенного эллипса — символа процесса к нижней части этого же эллипса. Этот график должен изображать последовательность инициализаций подпроцессов. Вертикальное расположение верхней

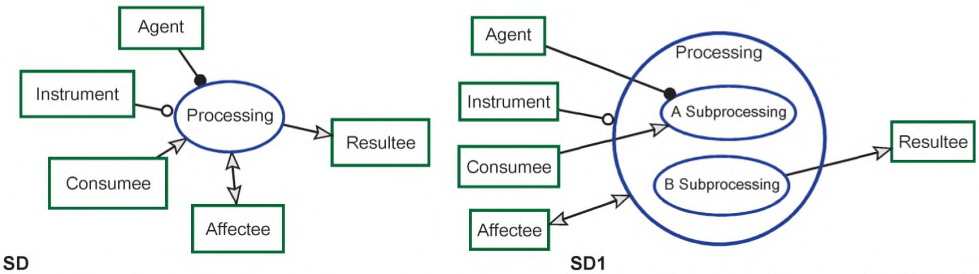
точки эллипсов — символов подпроцесса внутри внешнего эллипса — символа процесса должно указывать номинальную последовательность выполнения подпроцессов в контексте основного процесса.

Аналогично детализации процесса детализация объекта должна раскрывать составные объекты как части детализированного объекта и, возможно, временных процессов, которые являются операциями с детализированными объектами в области контекста детализированного объекта. В отличие от детализированного процесса детализация объекта не приводит к передаче контроля исполнения. Последствием детализации новой диаграммы объекта является смещение контекста от объекта (как части большего OPD-контекста) в направлении к объекту (как полному OPD-контексту), в котором составные части объекта раскрываются и пространственно или логически упорядочиваются.

Графически размеры прямоугольника детализированного объекта будут увеличиваться для размещения в нем символов для объектов типа *refinees* и связей между ними. Расположение прямоугольников объектов в контексте детализированного объекта должно указывать пространственное расположение или логическое упорядочение объектов. Это позволяет упорядочивать нумерацию данных, например, для векторов или матриц.

Соответствующее OPL-предложение для детализированного объекта должно иметь следующий вид: Объект **Object** (Объект) распадается на субобъекты **Subobject A**, **Subobject B** и **Subobject C** в указанном порядке.

Пример 1 — На рисунке 46 слева изображена структурная диаграмма (SD) обобщенного процесса Processing (Обработка), а справа — структурная диаграмма (SD1) процесса Processing после детализации, показывающая его два подпроцесса.



SD

Объект **Agent** (Агент) управляет процессом **Processing** (Обработка).

Процесс **Processing** (Обработка) требует объекта **Instrument** (Инструментальное средство).

Процесс **Processing** (Обработка) потребляет объект **Consumee**.

Процесс **Processing** (Обработка) взаимодействует с объектом **Affectee**.

Процесс **Processing** (Обработка) создает объект **Resultee**.

SD1

Процесс **Processing** (Обработка) требует объекта **Instrument** (Инструментальное средство).

Процесс **Processing** (Обработка) взаимодействует с объектом **Affectee**.

Процесс **Processing** (Обработка) распадается на подпроцессы **A Subprocessing** (Подпроцесс А) и **B Subprocessing** (Подпроцесс В) в указанной последовательности.

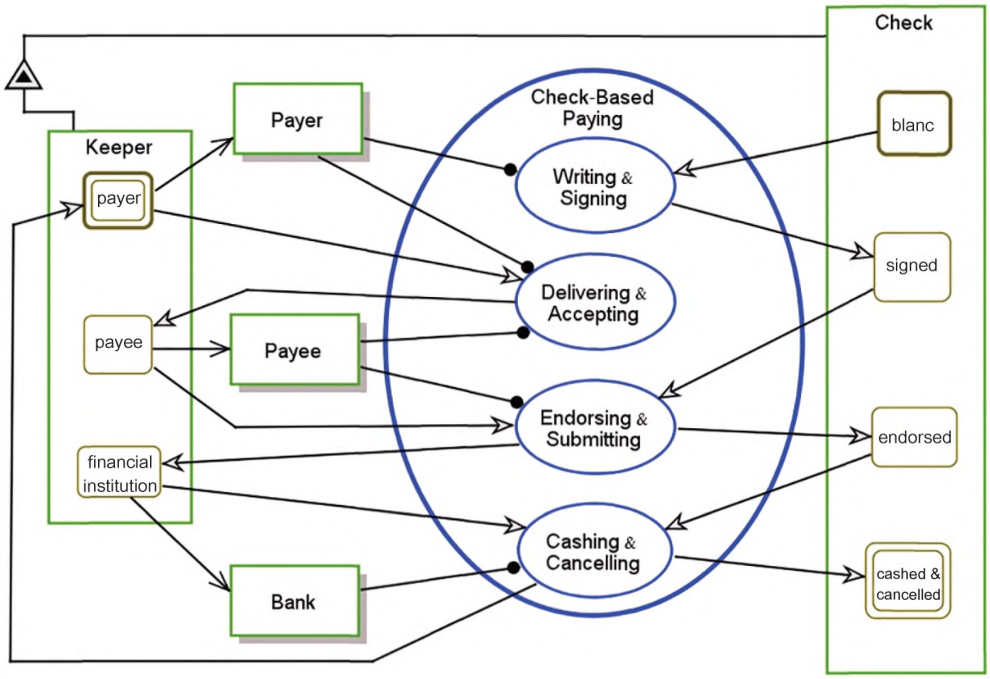
Объект **Agent** (Агент) управляет подпроцессом **A Subprocessing** (Подпроцесс А).

Подпроцесс **A Subprocessing** (Подпроцесс А) потребляет объект **Consumee**.

Подпроцесс **B Subprocessing** (Подпроцесс В) создает объект **Resultee**.

Рисунок 46 — Пример новой детализированной диаграммы

Пример 2 — На рисунке 47 изображена диаграмма процесса Check-Based Paying (Расчеты по чекам) (см. рисунок 29) с детализацией для раскрытия последовательности подпроцессов и распределения связей между процессом и его подпроцессами.



Объект **Check** (Чек) представляется объектом **Keeper** (Владелец чека).

Объект **Check** (Чек) может находиться в состоянии **blank** (неподписанный), **signed** (подписанный), **endorsed** (подтвержденный) или **cashed & cancelled** (инкассированный & аннулированный).

Состояние **blank** (неподписанный) объекта **Check** (Чек) является **initial** (исходным).

Состояние **cashed & cancelled** (инкассированный & аннулированный) объекта **Check** (Чек) является **final** (конечным).

Объект **Keeper** (Владелец чека) может находиться в состоянии **payer** (плательщик), **payee** (получатель платежа) или **financial institution** (финансово-кредитное учреждение).

Состояние **payer** (плательщик) объекта **Keeper** (Владелец чека) является **initial** (исходным) и **final** (конечным).

Состояние **payer** (плательщик) объекта **Keeper** (Владелец чека) связано с физическим объектом **Payer** (Плательщик).

Состояние **payee** (получатель платежа) объекта **Keeper** (Владелец чека) связано с физическим объектом **Payee** (Получатель платежа).

Состояние **financial institution** (финансово-кредитное учреждение) объекта **Keeper** (Владелец чека) связано с физическим объектом **Bank** (Банк).

Процесс **Check-Based Paying** (Расчеты по чекам) распадается на процессы **Writing & Signing** (Прочтение & Подписание), **Delivering & Accepting** (Выдача & Акцептование), **Endorsing & Submitting** (Утверждение & Предъявление) и **Cashing & Cancelling** (Получение наличными & Погашение), в указанной последовательности.

Физический объект **Payer** (Плательщик) управляет процессами **Writing & Signing** (Прочтение & Подписание) и **Delivering & Accepting** (Выдача & Акцептование).

Физический объект **Payer** (Плательщик) управляет процессами **Delivering & Accepting** (Выдача & Акцептование) и **Endorsing & Submitting** (Утверждение & Предъявление).

Физический объект **Bank** (Банк) управляет процессом **Cashing & Cancelling** (Получение наличными & Погашение).

Процесс **Writing & Signing** (Прочтение & Подписание) изменяет состояние объекта **Check** (Чек) из состояния **blank** (неподписанный) на состояние **signed** (подписанный).

Процесс **Delivering & Accepting** (Выдача & Акцептование) изменяет состояние объекта **Keeper** (Владелец чека) из состояния **payer** (плательщик) на состояние **payee** (получатель платежа).

Процесс **Endorsing & Submitting** (Утверждение & Предъявление) изменяет состояние объекта **Check** (Чек) из состояния **signed** (подписанный) на состояние **endorsed** (подтвержденный), и состояние объекта **Keeper** (Владелец чека) из состояния **payee** (получатель платежа) на состояние **financial institution** (финансово-кредитное учреждение).

Процесс **Cashing & Cancelling** (Получение наличными & Погашение) изменяет состояние объекта **Check** (Чек) из состояния **endorsed** (подтвержденный) на состояние **cashed & cancelled** (полученное наличными & погашенное) и объекта **Keeper** (Владелец чека) из состояния **financial institution** (финансово-кредитное учреждение) на состояние **payer** (плательщик).

Рисунок 47 — Диаграмма детализированного процесса Check-Based Paying (Расчеты по чекам), представленного последовательными подпроцессами

Примечание 2 — Детализация характеризует поведение процесса, который является результатом действия структурных и процедурных связей, указывающих динамическую передачу контроля исполнения между OPD-диаграммами. Операционный контекст исполнения переходит от процесса к детализированной OPD-диаграмме, а затем обратно к процессу.

14.2.2 Контрольная (операционная) семантика в контексте детализированного процесса

14.2.2.1 Неявная инициализирующая связь

Детализация процесса должна определять передачу контроля исполнения к подпроцессам с различными степенями детализации. Выполнение процесса в детализированном контексте должно рекурсивно передавать контроль исполнения на самый верхний подпроцесс (подпроцессы) в том контексте процесса, который находится на другой OPD-диаграмме (в случае детализации на новой диаграмме). Контроль исполнения должен возвращаться к детализированному процессу после завершения его разрешенного подпроцесса.

Неявная инициализирующая связь должна инициализировать связи между процессом и детализированным подпроцессом, между двумя подпроцессами в контексте детализированного процесса или между детализированным подпроцессом и его родительским процессом. Подобно своему явному аналогу неявная инициализирующая связь должна означать инициализацию последующего процесса или одновременно начинающихся процессов.

По достижении контекста детализированного процесса контроль исполнения должен немедленно передаваться подпроцессу (подпроцессам), имеющему на диаграмме самый высокий эллипс в контексте данного детализированного процесса. Неявная инициализирующая связь между процессом и его наивысшим детализированным подпроцессом позволяет передавать контроль исполнения. Вдоль графика процесса завершение подпроцесса источника будет немедленно инициализировать последующий подпроцесс (подпроцессы) с помощью неявной инициализирующей связи. После завершения подпроцесса, соответствующего самому высокому эллипсу, который является наинизшим в данном детализированном контексте, контроль исполнения должен возвращаться к детализированному процессу (вместе с неявной инициализирующей связью).

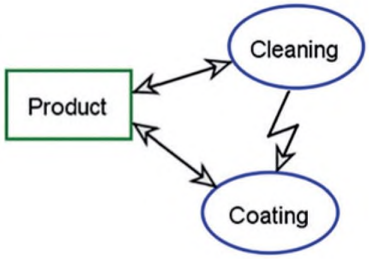
Поскольку инициализация является событием, удовлетворение предварительных условий для каждого подпроцесса необходимо для выполнения этих подпроцессов.

Если два или более подпроцессов имеют свои самые высокие точки эллипсов на одной и той же высоте, то неявная инициализирующая связь должна инициализировать каждый из процессов, и после удовлетворения индивидуальных предварительных условий они должны начинаться одновременно. Последний из завершенных процессов будет инициализировать следующий процесс (или набор параллельных подпроцессов).

Графически отсутствие символа должно означать наличие неявной инициализирующей связи. Вертикальное расположение (сверху вниз) наивысших точек символов подпроцесса (эллипсов) в контексте детализированного процесса должно обозначать наличие неявной инициализирующей связи между последовательно выполняемыми подпроцессами в данной конфигурации процессов.

Синтаксис OPL-предложения для неявной инициализирующей связи должен иметь следующий вид: Процесс **Process** (Процесс) распадается на подпроцессы **Subprocess A** и **Subprocess B** в указанной последовательности.

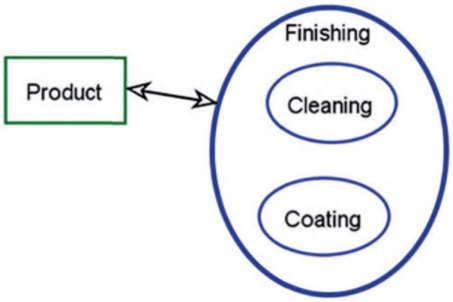
Пример — На OPD-диаграмме, изображенной в левой части рисунка 48, процесс Cleaning (Очистка) инициирует процесс Coating (Покрытие), поэтому процесс Cleaning вначале взаимодействует с объектом Product (Продукция), а затем с ним взаимодействует процесс Coating. Инициализирующая связь диктует выполнение процессов в данной последовательности. На эквивалентной OPD-диаграмме справа процесс Finishing (Отделка) разделяется на два подпроцесса — Cleaning и Coating с эллипсом первого подпроцесса над эллипсом второго из подпроцессов, поэтому после запуска процесса Finishing контроль исполнения немедленно передается на подпроцесс Cleaning, и после его завершения неявная инициализирующая связь будет инициализировать подпроцесс Coating. Эти две OPD-диаграммы семантически эквивалентны, за исключением того, что на левой диаграмме отсутствует процесс Finishing как включающий контекст, что делает его менее выразительным с системной точки зрения, хотя с использованием большего числа графических элементов.



Процесс **Cleaning** (Очистка) взаимодействует с объектом **Product** (Продукт).

Процесс **Cleaning** (Очистка) инициирует процесс **Coating** (Покрытие).

Процесс **Coating** (Покрытие) взаимодействует с объектом **Product** (Продукт).



Процесс **Finishing** (Отделка) взаимодействует с объектом **Product** (Продукт).

Процесс **Finishing** распадается на два подпроцесса **Cleaning** (Очистка) и **Coating** (Покрытие) в указанной последовательности.

Рисунок 48 — Диаграммы для инициализирующей связи (слева) и неявной инициализирующей связи (справа)

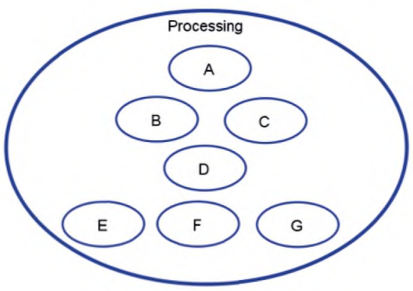
14.2.2.2 Набор неявных параллельных инициализирующих связей

Графически, если верхние точки двух или более эллипсов подпроцессов (в пределах области детализированного процесса) находятся на одинаковой высоте (в пределах возможного допустимого допуска), то эти подпроцессы должны начинать выполняться одновременно и параллельно (при условии удовлетворения ими соответствующих условий). В этой ситуации возникает набор неявных инициализирующих связей между процессом-источником подобной связи и каждым из параллельно действующих процессов.

Высоты расположения верхних точек закрытых эллипсов подпроцессов частично упорядочивает их. Подпроцессы, чьи верхние точки эллипсов находятся на одной и той же высоте, начинают выполняться параллельно. После окончания выполнения последнего из этих подпроцессов, т. е. когда происходит синхронизация процесса, контроль исполнения должен предпринимать попытку инициализировать следующий подпроцесс. При наличии двух и более подпроцессов с более низкими верхними точками эллипсов на одной и той же высоте контроль исполнения будет инициализировать их одновременное выполнение. При отсутствии дополнительных подпроцессов, требующих инициализации, контроль исполнения возвращается к детализированному процессу с сущностью типа *refineable*.

Синтаксис OPL-предложения для неявной параллельной инициализирующей связи должен иметь следующий вид: Процесс **Process** (Процесс) распадается на два параллельных подпроцесса **Subprocess A** и **Subprocess B**.

Пример — На рисунке 49 показаны подпроцессы, расположенные в следующей конфигурации: A, (B, C), D, (E, F, G). Процессы B и C начинают выполняться после завершения процесса A. Процесс D начинает выполняться после завершения наиболее продолжительного из процессов B и C. Процессы E, F и G начинают выполняться после завершения процесса D. Контроль исполнения возвращается к процессу Processing (Обработка) после завершения наиболее продолжительного из процессов E, F и G.



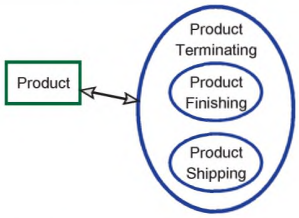
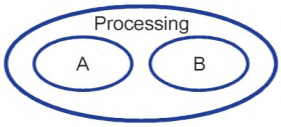
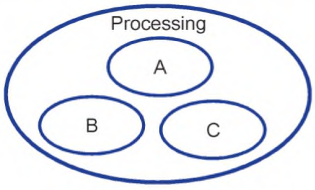
Процесс **Processing** (Обработка) распадается на подпроцесс **A,** параллельные подпроцессы **B** и **C, D** и параллельно — на подпроцессы **E, F** и **G** в указанной последовательности.

Рисунок 49 — Порядок выполнения подпроцессов и набор неявных параллельных инициализирующих связей

14.2.2.3 Обзор неявных инициализирующих связей

В таблице 24 приведены все неявные инициализирующие связи.

Таблица 24 — Обзор неявных инициализирующих связей

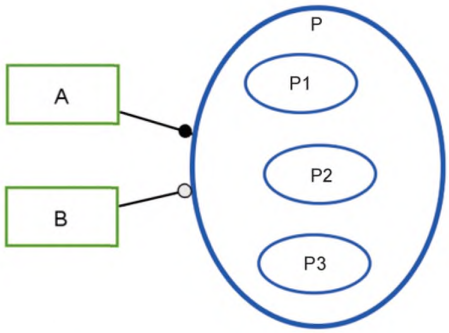
Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Неявная инициализирующая связь	По завершении подпроцесса в рамках контекста детализированного процесса подпроцесс будет немедленно инициализировать один или несколько нижележащих подпроцессов	 Процесс **Product Terminating** (Завершение выпуска продукции) распадается на процессы **Product Finishing** (Отделка продукции) и **Product Shipping** (Отгрузка продукции) в указанной последовательности	Инициализация процесса, верхняя точка эллипса которого находится выше инициализированого процесса	Инициализированный процесс, верхняя точка эллипса которого находится ниже инициализирующего процесса
Набор параллельных неявных инициализирующих связей	Верхняя диаграмма: подпроцессы **A** и **B** инициализируются одновременно сразу же после начала выполнения процесса **Processing**. Нижняя диаграмма: подпроцессы **B** и **C** инициализируются одновременно сразу же после окончания выполнения подпроцесса **A**	 Процесс **Processing** (Обработка) распадается на два параллельных процесса **A** и **B**  Процесс **Processing** (Обработка) распадается на процесс **A** и параллельно на процессы **B** и **C** в указанной последовательности	Инициализация процесса, верхняя точка эллипса которого находится выше набора инициализированных процессов, чьи верхние точки эллипсов находятся на одной и той же высоте (в пределах предварительно установленных допусков)	Набор инициализированных процессов, верхние точки эллипса которых находятся на одной и той же высоте (в пределах предварительно установленных допусков) и ниже верхних точек элипса инициализирующего процесса

14.2.2.4 Распределение связей по контексту

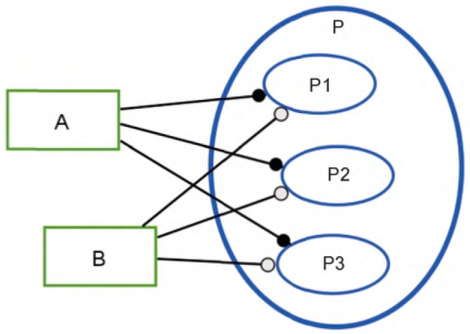
14.2.2.4.1 Семантика распределения связей

Графически процедурная связь, присоединяемая к контуру детализированного процесса, имеет дистрибутивную семантику. Прерывание этой связи, присоединенной к контуру детализированного процесса будет означать, что связь является распределенной и присоединяемой к каждому из подпроцессов. Контур детализированного процесса имеет семантику, аналогичную таковой для алгебраических скобок после символа умножения, которые распространяют оператор умножения на выражения внутри скобок.

Пример 1 — На рисунке 50 OPD-диаграммы слева и справа являются эквивалентными, однако диаграмма слева является более четкой и менее скученной. Агентская связь между объектом A и процессом P означает, что объект A управляет подпроцессами P1, P2 и P3. Инструментальная связь между объектом B и процессом P означает, что подпроцессы P1, P2 и P3 требуют объекта B. Так же как и в алгебре предполагается, что агентская (или инструментальная) связь является оператором умножения, объект A — множителем, а детализация — сложением, так что если P = P1 + P2 + P3 и P являются сомножителями, то A*P = A*(P1 + P2 + P3) = A*P1 + A*P2 + A*P3.



Объект **A** управляет процессом **P**.
Процесс **P** требует объекта **B**.
Процесс **P** распадается на подпроцессы **P1, P2** и **P3** в указанной последовательности.



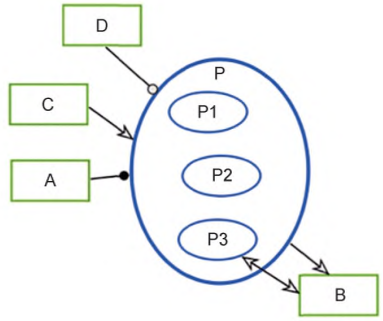
Процесс **P** распадается на подпроцессы **P1, P2** и **P3** в указанной последовательности.
Объект **A** работает с подпроцессами **P1, P2** и **P3**.
Подпроцессы **P1, P2** и **P3** требуют объекта **B**.

Рисунок 50 — Диаграммы распределения детализированных связей

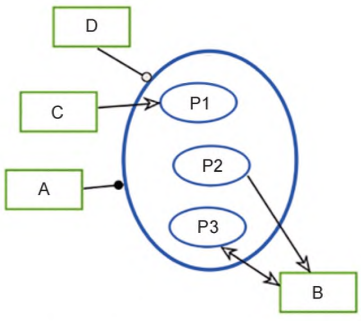
Если объект типа *enabler* присоединяется к внешнему контуру детализированного процесса, то он должен присоединяться по меньшей мере еще к одному из его подпроцессов. Потребительские и результирующие связи не должны присоединяться к внешнему контуру детализированного процесса, поскольку это будет нарушать временные логические условия. В случае распределенной потребительской связи будет предприниматься попытка потребления объектов, уже потребленных подпроцессами, которая не является первой связью. Аналогичным образом распределенная результирующая связь будет предпринимать попытку создания объектов, у которых уже существуют соответствующие экземпляры.

Примечание 1 — Разработчик модели должен проявлять осторожность в тех случаях, когда несколько процессов создают один и тот же объект (т. е. существует несколько операционных экземпляров объекта), или когда взаимодействуют между собой несколько процессов, или когда потребляется один и тот же объект. Средства OPM-моделирования необходимы для отслеживания числа и сущностей операционных экземпляров каждого объекта и каждого процесса для получения возможности проведения моделирования.

Пример 2 — На рисунке 51 левая OPD-диаграмма содержит недействующие потребительские и воздействующие связи, аннотированные на OPL-языке. Потребительская связь порождает OPL-предложение вида: «Процесс P потребляет объект C». Следствием применения распределения связей является порождение трех OPL-предложений «Подпроцесс P1 потребляет объект C», «Подпроцесс P2 потребляет объект C» и «Подпроцесс P3 потребляет объект C». Тем не менее поскольку подпроцесс P1 первым потребляет объект C в соответствии с установленным временным порядком, то тот же экземпляр объекта C не будет существовать при выполнении подпроцесса P2 или P3, и поэтому подпроцессы P2 и P3 не будут потреблять объект C. Аналогично тот же операционный экземпляр объекта B возникает лишь единожды. На OPD-диаграмме справа изображены действующие связи, определяющие, какие из подпроцессов процесса P потребляют процесс C (P1) и какие из них создают объект B (P2).



Объект **A** управляет процессом **P**.
Процесс **P** требует объекта **D**.
Процесс **P** распадается на подпроцессы **P1, P2** и **P3** в указанной последовательности.
Процесс **P** потребляет объект **C**. — Недействительно!
Процесс **P** создает объект **B**. — Недействительно!
Подпроцесс **P3** взаимодействует с объектом **B**.



Объект **A** управляет процессом **P**.
Процесс **P** требует объекта **D**.
Процесс **P** преобразуется в подпроцессы **P1, P2** и **P3** в указанной последовательности.
Подпроцесс **P1** потребляет объект **C**.
Подпроцесс **P2** создает объект **B**.
Подпроцесс **P3** взаимодействует с объектом **B**.

Рисунок 51 — Диаграмма, иллюстрирующая ограничение на распределение потребительских и результирующих связей

Поскольку присоединение потребительской или результирующей связи к детализированному процессу является недопустимым, то все указанные связи должны присоединяться (первоначально или по умолчанию) к первому из подпроцессов.

Примечание 2 — Средство моделирования может автоматически устанавливать по умолчанию семантику, которую может модифицировать.

Пример 3 — На рисунке 51, как только разработчик модели детализирует процесс P и введет подпроцесс P1 в его контекст, конец потребительской связи, находящийся у пункта назначения, для объекта C начинает перемещаться от процесса P к подпроцессу P1. Аналогичным образом конец около источника взаимодействующей связи к объекту B также будет перемещаться от процесса P к подпроцессу P1. Если разработчик модели вводит подпроцесс P2, то он может перемещать конец потребительской связи, находящийся у пункта назначения, и/или конец источника результирующей связи от подпроцесса P1 к подпроцессу P2.

14.2.2.4.2 Ограничение на использование ситуационной и условной связей

Событийная связь от системного объекта или состояния не должна пересекать границу детализированного процесса с внешней стороны (in-zoomed process) этого процесса, чтобы обеспечить возможность инициализировать один из его подпроцессов на любом уровне, поскольку это равносильно попытке помешать предписанному временному порядку синхронизации (см. 14.2.3.5) детализированного процесса. Если пересекающая событийная связь исходит со стороны объекта внешней среды или состояния, то разработчик модели должен смоделировать, как осуществлять обработку в подобных обстоятельствах.

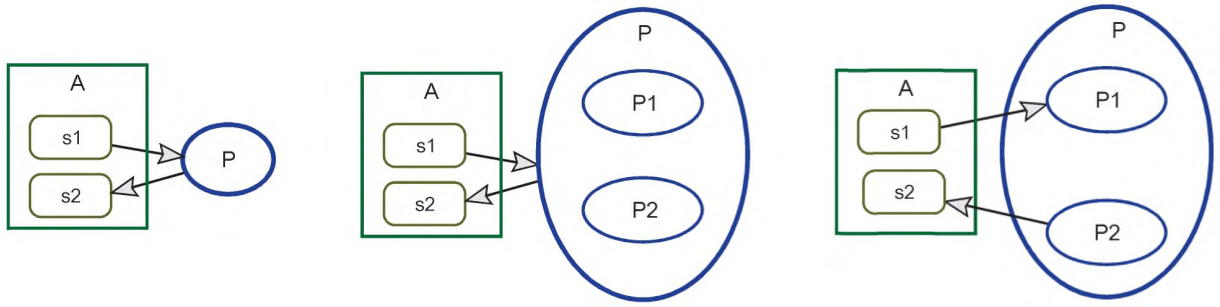
Если пропущенный процесс находится в пределах детализированного контекста и для него существует последующий процесс в этом контексте, то контроль исполнения будет инициализировать этот процесс; в противном случае контроль исполнения будет передаваться назад детализированному процессу.

14.2.2.4.3 Разделенные определяющие состояние преобразующие связи

Если процесс, который изменяет состояние объекта из входного в выходное состояние, является детализированным и содержит несколько подпроцессов, то OPD-диаграмма, существующая (in-diagram) или новая (new-diagram) становится недостаточно определенной. Для восстановления спецификации разработчик модели должен осуществимым по времени образом присоединить определяющую состояние входа связь и определяющую состояние выхода связь к одному из подпроцессов. Разделение пары определяющих вход/выход связей на две соответствующие связи должно означать разделенную пару определяющей состояние трансформирующей связи.

Графически две связи с объектом, имеющим два или более состояния, которые соединяются (через контур процесса) с различными подпроцессами с одной определяющей состояние входа связью и одной определяющей состояние выхода связь, должны означать разделенную определяющую состояние преобразующую связь.

Пример 1 — На рисунке 52 средняя OPD-диаграмма представляет детализированный процесс P, показанный на левой диаграмме, однако он является недостаточно определенным, поскольку каждый из подпроцессов P1 или P2 может изменять состояние объекта A из состояния s1 на состояние s2 или подпроцесс P1 может изменять состояние объекта A из состояния s1, а подпроцесс P2 может изменять состояние объекта A на состояние s2. OPD-диаграмма справа моделирует последний случай, приводящий к появлению новой разделенной входной связи между состоянием s1 объекта A и подпроцессом P1, а также новой разделенной выходной связи между подпроцессом P2 и состоянием s2.



Объект **A** может находиться в состоянии **s1** или **s2**.

Процесс **P** изменяет состояние объекта **A** из состояния **s1** на состояние **s2**.

Объект **A** может находиться в состоянии **s1** или **s2**.

Процесс **P** распадается на подпроцессы **P1** и **P2** в указанной последовательности.

Процесс **P** изменяет состояние объекта **A** из состояния **s1** на состояние **s2**. — Недостаточно определено!

Объект **A** может находиться в состоянии **s1** или **s2**.

Процесс **P** распадается на подпроцессы **P1** и **P2** в указанной последовательности.

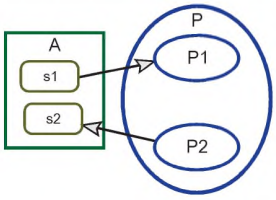
Подпроцесс **P1** изменяет состояние объекта **A** из состояния **s1**.

Подпроцесс **P2** изменяет состояние объекта **A** на состояние **s2**.

Рисунок 52 — Разделенная определяющая состояние преобразующая связь для разрешения согласно спецификации

В таблице 25 приведены все пары разделенных определяющих вход/выход воздействующих связей.

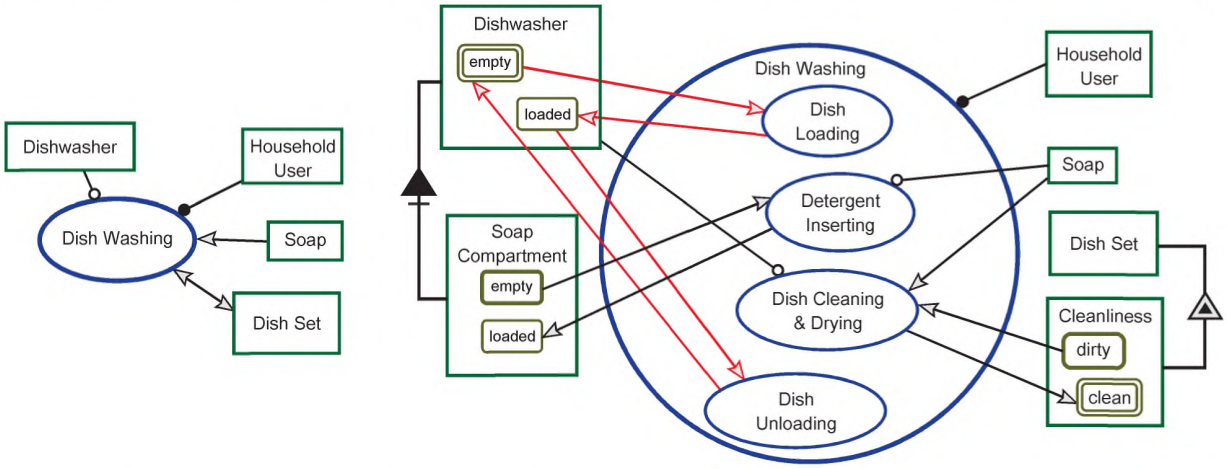
Таблица 25 — Пара разделенных определяющих вход/выход воздействующих связей

Наименование	Семантика	Пример OPD & OPL	Источник	Получатель
Пара разделенных определяющих вход/выход воздействующих связей *Верхняя стрелка:* разделенная определяющая вход воздействующая связь *Нижняя стрелка:* разделенная определяющая выход воздействующая связь	Наиболее ранний подпроцесс детализированного процесса выводит объект из его входного состояния. Наиболее поздний подпроцесс того же детализированного процесса изменяет его состояние на выходное	 Состояние **s1** объекта **A** изменяет подпроцесс **P1**. Подпроцесс **P2** изменяет состояние объекта **A** на состояние **s2**	*Верхняя стрелка:* входное состояние воздействующего объекта *Нижняя стрелка:* наиболее поздний подпроцесс детализированного процесса	*Верхняя стрелка:* наиболее ранний подпроцесс детализированного процесса *Нижняя стрелка:* входное состояние воздействующего объекта

Примечание 1 — Не существует вариантов сочетания разделенной определяющей состояние воздействующей связи с контрольной связью.

Примечание 2 — Объект может выполнять роль инструментального средства на обобщенной OPD-диаграмме и объекта типа *transformee* на другой дочерней, более подробной и конкретной OPD-диаграмме. На обобщенной OPD-диаграмме процесс не оказывает влияния на объект, поскольку начальное состояние объекта аналогично его конечному состоянию. Таким образом, на обобщенной OPD-диаграмме объект является инструментом, что показано с помощью инструментальной связи. Тем не менее на дочерней, более конкретизированной OPD-диаграмме тот же процесс не приведет к изменению состояния этого объекта из начального состояния, а затем обратно в это же состояние.

Пример 2 — На левой структурной диаграмме SD [структурная диаграмма Dish Washing System (Посудомоечная система)] рисунка 53 объект Dishwasher (Посудомоечная машина) имеет инструментальную связь с процессом Dish Washing (Мытье посуды), поскольку отсутствуют видимые изменения состояния объекта Dishwasher на данной степени обобщения. На дочерней OPD-диаграмме SD1 (структурная диаграмма детализированного процесса Dish Washing) процесс Dish Washing распадается на подпроцесс Dish Loading (Загрузка посуды) с состоянием dirty (грязное) объекта Dish Set (Партия посуды), на подпроцесс Dish Cleaning (Очистка посуды), который изменяет состояние объекта Dish Set из состояния dirty (грязное) на состояние clean (чистое), на подпроцесс Dish Unloading (Выгрузка посуды) с состоянием clean для объекта Dish Set. Процесс Dish Loading изменяет состояние объекта Dishwasher из состояния empty (незагруженное) на состояние loaded (загруженное), тогда как процесс Dish Unloading вызывает обратное изменение состояния из состояния loaded на состояние empty, поэтому состояние empty является и начальным, и конечным состоянием (выделены коричневым цветом). Хотя объект Dishwasher на структурной диаграмме является инструментальным средством, на дочерней более детализированной OPD-диаграмме объект Dishwasher является объектом типа affectee — он переходит в состояние loaded, а затем вновь возвращается в состояние empty. Единственное действие, заметное на структурной диаграмме SD — это воздействие на объект Dish Set.



SD: Структурная диаграмма **Dish Washing System** (Посудомоечная система).

Объект **Household User** (Бытовой потребитель) управляет процессом **Dish Washing** (Мытье посуды).

Процесс **Dish Washing** (Мытье посуды) требует объекта **Dishwasher** (Посудомоечная машина).

Процесс **Dish Washing** (Мытье посуды) потребляет объект **Soap** (Мыло).

Процесс **Dish Washing** (Мытье посуды) взаимодействует с объектом **Dish Set** (Партия посуды).

SD1: Структурная диаграмма детализированного процесса **Dish Washing** (Мытье посуды).

Объект **Dish Washer** (Посудомоечная машина) содержит объект **Soap Compartment** (Отделение для мыла) и другие части.

Объект **Dishwasher** (Посудомоечная машина) может находиться в состоянии **empty** (незаполненное) или **loaded** (заполненное).

Состояние **empty** (незагруженное) объекта **Dishwasher** (Посудомоечная машина) является и начальным, и конечным.

Объект **Soap Compartment** (Отделение для мыла) может находиться в состоянии **empty** (незагруженное) или **loaded** (загруженное).

Состояние **empty** (незагруженное) объекта **Soap Compartment** (Отделение для мыла) является начальным.

Объект **Dish Set** (Партия посуды) представляется объектом **Cleanliness** (Чистота).

Объект **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) может находиться в состоянии **dirty** (грязное) или **clean** (чистое).

Рисунок 53, лист 1 — Диаграмма, иллюстрирующая роль обобщения с разделяющими состояние преобразующими связями

Состояние **dirty** (грязное) объекта **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) является начальным.

Состояние **clean** (чистое) объекта **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) является конечным.

Объект **Household User** (Бытовой потребитель) управляет процессом **Dish Washing** (Мытье посуды).

Процесс **Dish Washing** (Мытье посуды) распадается на подпроцессы **Dish Loading** (Загрузка посуды), **Detergent Inserting** (Введение моющего средства), **Dish Cleaning & Drying** (Очистка & Просушка посуды) и **Dish Unloading** (Выгрузка посуды) в указанной последовательности.

Процесс **Dish Loading** (Загрузка посуды) изменяет состояние объекта **Dishwasher** (Посудомоечная машина) из состояния **empty** (незагруженное) на состояние **loaded** (загруженное).

Процесс **Detergent Inserting** (Введение моющего средства) требует объекта **Soap** (Мыло).

Процесс **Detergent Inserting** (Введение моющего средства) изменяет состояние объекта **Soap Compartment** из состояния **empty** (незагруженное) на состояние **loaded** (загруженное).

Процесс **Dish Cleaning & Drying** (Очистка & Просушка посуды) требует объекта **Dishwasher** (Посудомоечная машина).

Процесс **Dish Cleaning & Drying** (Очистка & Просушка посуды) потребляет объект **Soap** (Мыло).

Процесс **Dish Cleaning & Drying** (Очистка & Просушка посуды) изменяет состояние объекта **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) из состояния **dirty** (грязное) на состояние **clean** (чистое).

Процесс **Dish Unloading** (Выгрузка посуды) изменяет состояние объекта **Dishwasher** (Посудомоечная машина) из состояния **loaded** (загруженное) на состояние **empty** (незагруженное).

Рисунок 53, лист 2

14.2.2.4.4 Операционные экземпляры набора действующих объектов

Как следствие распределения связей к операционным экземплярам объекта типа *transformees* должны применяться следующие ограничения:

- каждый операционный экземпляр объекта типа *consumee* в наборе предварительно обработанных объектов прекращает свое существование в начале наиболее детализированного подпроцесса данного процесса, который потребляет операционный экземпляр объекта, причем этот экземпляр не содержится в указанном наборе;
- каждый операционный экземпляр объекта типа *affectee* в наборе предварительно обработанных объектов в процессе, который изменяет этот экземпляр в результате выполнения процесса, должен выходить из своего входного состояния, состояния, из которого он изменялся в начале наиболее детализированного подпроцесса, который изменяет объект типа *affectee*;
- каждый операционный экземпляр объекта типа *affectee* в наборе апостериорно обработанных объектов в процессе, который изменяет этот экземпляр в результате выполнения процесса, должен входить в свое выходное состояние по завершении наиболее детализированного подпроцесса, который изменяет объект типа *affectee*;
- каждый операционный экземпляр объекта типа *resultee* в наборе апостериорно обработанных объектов в процессе начинает свое существование по завершении наиболее детализированного подпроцесса, который создает операционный экземпляр объекта типа *resultee*, причем этот экземпляр не содержится в указанном наборе.

Примечание 1 — Структурированный объект **B**, для которого выполнение процесса **P** приводит к изменению состояния объекта **B**, выходит из входного состояния в начале наиболее детализированного подпроцесса процесса **P**, который изменяет объект **B**, и входит в выходное состояние того же подпроцесса процесса **P** или некоторого другого последующего подпроцесса. Поскольку выполнение процесса **P** занимает определенное количество времени, объект **B** находится в переходном состоянии между состояниями (его входным состоянием и его выходным состоянием); объект выходит из своего входного состояния, однако еще не переходит к своему выходному состоянию.

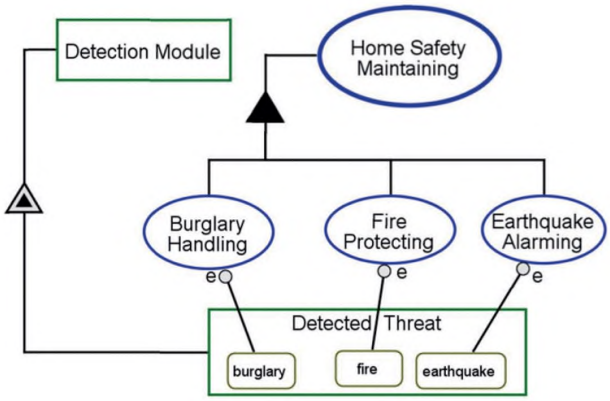
14.2.2.4.5 Уточнение синхронного процесса относительно асинхронного процесса

Поскольку фундаментальная структурная реляционная связь типа «агрегация — является частью» (aggregation-participation fundamental structural relation) не предусматривает какого-либо «частичного упорядочивания» (partial order) рабочих параметров процесса, для моделирования уточненного синхронного процесса необходимо использовать операцию детализации.

Пример 1 — Система, изображенная на рисунке 53, является синхронной: существует фиксированный, четкий порядок следования каждого подпроцесса в контексте детализированного процесса Dish Washing (Мойка посуды).

Для моделирования уточненного синхронного процесса следует использовать фундаментальную структурную связь типа «агрегация — является частью» посредством группового разворачивания на существующей или новой диаграмме процесса.

Пример 2 — На рисунке 54 показана часть системы Home Safety System (Домашняя система безопасности), которая выполняет функцию процесса Home Safety Maintaining (Техническое обслуживание домашней системы безопасности), который включает в себя подпроцессы Burglary Handling (Защита от проникновения в дом), Fire Protecting (Защита от пожара) и Earthquake Alarming (Предупреждение о землетрясении). Поскольку порядок выполнения этих трех подпроцессов неизвестен, на OPD-диаграмме используются групповое разворачивание и связь типа «агрегация — является частью» от этой функции, а не от детализированного варианта процесса Home Safety Maintaining, который детализируется в рекуррентный систематический процесс (на рисунке не показан) Monitoring & Detecting (Мониторинг & Обнаружение), для которых объект Detection Module (Модуль обнаружения) является инструментальным средством, а процесс Threat Appearing является процессом внешней среды.



Процесс **Home Safety Maintaining** (Техническое обслуживание домашней системы безопасности) включает процессы **Burglary Handling** (Защита от проникновения в дом), **Fire Protecting** (Защита от пожара) и **Earthquake Alarming** (Предупреждение о землетрясении).

Объект **Detection Module** (Модуль обнаружения) представляется объектом **Detected Threat** (Выявленная угроза).

Detected Threat (Выявленная угроза) может находиться в состоянии **burglary** (проникновение), **fire** (пожар) или **earthquake** (землетрясение).

Состояние **burglary** (проникновение) объекта **Detected Threat** (Выявленная угроза) инициирует процесс **Burglary Handling** (Защита от проникновения в дом), который требует состояния **burglary** объекта **Detected Threat**.

Состояние **fire** (пожар) объекта **Detected Threat** (Выявленная угроза) инициирует процесс **Fire Protecting** (Защита от пожара), который требует состояния **fire** объекта **Detected Threat**.

Состояние **earthquake** (землетрясение) объекта **Detected Threat** (Выявленная угроза) инициирует процесс **Earthquake Alarming** (Предупреждение о землетрясении), который требует состояния **earthquake** объекта **Detected Threat**.

Рисунок 54 — Диаграмма, показывающая, что процесс **Home Safety Maintaining** соответствует асинхронной системе

14.2.2.6 Выражение контекста системы

14.2.2.6.1 Навигация по контексту системы

14.2.2.6.1.1 Древо OPD-процессов

Древо OPD-процессов, называемое также OPD-древом, должно быть направленным древом с корневым узлом SD (системной диаграммой), а также с другими OPD-диаграммами в качестве узлов с соответствующими метками. Направленные ребра OPD-древа должны иметь метки с каждым ребром, направленным от родительской OPD-диаграммы, которая содержит процесс типа *refineable*, к дочерней OPD-диаграмме, содержащей объекты типа *refines*, которые уточняют процесс типа *refineable* на родительской OPD-диаграмме посредством детализации новой диаграммы для синхронных подпроцессов или группового разворачивания на новой диаграмме асинхронных подпроцессов.

14.2.2.6.1.2 Древо OPD-объектов

В отличие от древа OPD-процессов, которое имеет лишь один корень, древо OPD-объектов больше похоже на лес из множества деревьев, каждое из которых произрастает из отдельной сущности типа *refineable*, которая разворачивается или детализируется до требуемой степени. Вместо того чтобы идентифицировать возможный поток сигналов контроля исполнения на древе OPD-процесса, древо OPD-объектов должно формировать информацию об объекте в виде иерархической структуры. Работа системы должна поддерживать взаимозависимости между элементами древа OPD-объектов и древа OPD-процессов.

14.2.2.6.1.3 Метки на диаграммах в OPM-методологии

Именем OPM-системы должно быть имя OPM-модели, которая определяет эту систему. Имя OPD-диаграммы — это имя, которое должно идентифицировать каждую OPD-диаграмму на древе OPD-процесса.

SD должно быть меткой — обозначением корневой OPD-диаграммы в иерархическом OPD-древе. SD-диаграмма занимает уровень 0 в иерархии OPD-древа и имеет лишь один процесс; уровни с большими номерами, т. е. те, которые соответствуют последовательным уточнениям, могут содержать один или несколько процессов. SD-диаграмма должна содержать только один системный процесс, который представляет собой комплексную системную функцию, которая придает функциональную ценность для заинтересованных сторон. SD-диаграмма также может содержать один или несколько процессов внешней среды.

14.2.2.6.1.4 Метки ребер древа OPD-процесса

Поскольку каждый уточненный процесс на древе OPD-процессов имеет уникальное имя, каждая метка ребра древа должна относиться к уточнению этого процесса на другой OPD-диаграмме. Каждое ребро на OPD-древе процесса должно иметь метку, выражающую уточняющую связь, которая соответствует неявной инициализирующей связи или разворачивающейся связи. Принимая во внимание каждую OPD-диаграмму как объект и полное древо OPD-процессов в качестве единой OPD-диаграммы, каждым ребром древа должна быть однонаправленная тегированная структурная связь с тегом «уточняется с помощью детализированного объекта **Refineable Name** в» (**is refined by in-zooming** Refineable Name **in**) или «уточняется путем разворачивания объекта **Refineable Name** в» (**is refined by unfolding** Refineable Name **in**).

OPD-уточнением OPL-предложения должно быть OPL-предложение, описывающее уточненную связь между сущностью типа *refineable,* присутствующей на уровне N OPD-диаграммы и на уровне N + 1 уточненной OPD-диаграммы.

Синтаксис детализированной OPD-диаграммы уточненного OPL-предложения должен иметь следующий вид: Метка $\mathbf{Tier_N}$ **OPD label** уточняется путем детализации имени **Refineable Process Name** в метке $\mathbf{Tier_{N+1}}$ **OPD Label**.

Синтаксис разворачиваемой OPD-диаграммы уточненного OPL-предложения должен иметь следующий вид: Метка $\mathbf{Tier_N}$ **OPD label** уточняется путем разворачивания имени **Refineable Process Name** в метке $\mathbf{Tier_{N+1}}$ **OPD Label**.

Примечание — Несколько OPD-диаграмм в С.6 иллюстрируют использование синтаксиса для имени ребер.

14.2.2.6.1.5 Отображение системы и представление модели

Отображением (мэппингом) системы должно быть древо OPD-процесса, которое явно представляет контент элемента (сущности и связи) каждой OPD-диаграммы (узла). Поскольку отображение системы может быть достаточно большим и громоздким, методы должны обеспечивать доступ к контенту модели и межэлементным связям. Эти методы, в своей совокупности именуемые «представлениями

системы» и состоящие из фактов (событий) модели, должны содержать перечень всех сущностей и OPD-диаграмм, в которых эти сущности появляются, а также древа OPD-процессов и объектов.

Кроме того, набор OPM-средств должен обеспечивать метод создания ракурсов как OPD-диаграмм со связанными с ними OPL-предложениями, объектов и процессов, которые отвечают определенным критериям. Представления модели могут включать в себя критический путь минимальной длительности выполнения системы, или перечень системных агентов и инструментов, или OPD-диаграмму для объектов и процессов, содержащихся в связях конкретного вида (или наборе связей).

Пример — OPD-диаграмма может создаваться путем:

a) уточнения (разворачивания или детализации) объекта или

b) сбора и представления OPD-сущностей на новой OPD-диаграмме, которые проявляются в различных OPD-диаграммах для выражения назначения системных подфункций для объектов системного модуля.

14.2.2.6.2 Полная системная OPL-спецификация

OPL-разделом должна быть совокупность OPL-предложений, которые в своей совокупности будут определять в тексте семантическое выражение соответствующей OPD-диаграммы.

Примечание 1 — Имя OPL-раздела, использовавшего имя OPD-диаграммы, может предшествовать первому OPL-предложению каждого OPL-раздела.

Моделью OPM-системы должна быть совокупностью последовательных OPL-разделов, соответствующих совокупности существующих OPD-диаграмм.

Полная OPL-спецификация системы должна начинаться с начального заголовка OPL-спецификации. OPL-разделы должны следовать за заголовком в виде последовательных блоков, каждый из которых должен начинаться с новой строки с соответствующей OPD-диаграммой и последующим OPL-разделом предложения.

Примечание 2 — Последовательность OPL-разделов обычно начинается с SD-диаграммы и следует с первого порядка, если разработчик модели не идентифицирует другую последовательность этих разделов.

Пример — В таблице 26 содержится полная OPL-спецификация OPM-модели, изображенной на рисунке 53.

Таблица 26 — Полная OPL-спецификация на посудомоечную систему

OPL-спецификация на **Dish Washing System** (Посудомоечная система) SD: Структурная диаграмма **Dish Washing System** (Посудомоечная система) Объект **Household User** (Бытовой потребитель) управляет процессом **Dish Washing** (Мытье посуды). Процесс **Dish Washing** (Мытье посуды) требует объекта **Dishwasher** (Посудомоечная машина). Процесс **Dish Washing** (Мытье посуды) потребляет объект **Soap** (Мыло). Процесс **Dish Washing** (Мытье посуды) взаимодействует с объектом **Dish Set** (Партия посуды). SD1 — это структурная диаграмма детализированного процесса **Dish Washing** (Мытье посуды) SD. SD1: Детализированный процесс **Dish Washing** (Мытье посуды)
Объект **DishWasher** (Посудомоечная машина) содержит объект **Soap Compartment** (Отделение для мыла) и другие части. Объект **Dishwasher** (Посудомоечная машина) может находиться в состоянии **empty** (незаполненное) или **loaded** (заполненное). Состояние **empty** (незаполненное) объекта **Dishwasher** (Посудомоечная машина) является и начальным, и конечным. Объект **Soap Compartment** (Отделение для мыла) может находиться в состоянии **empty** (незаполненное) или **loaded** (заполненное). Состояние **empty** (незаполненное) объекта **Soap Compartment** (Отделение для мыла) является начальным. Объект **Dish Set** (Партия посуды) представляется объектом **Cleanliness** (Чистота). Объект **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) может находиться в состоянии **dirty** (грязное) или **clean** (чистое). Состояние **dirty** (грязное) объекта **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) является начальным. Состояние **clean** (чистое) объекта **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) является конечным. Объект **Household User** (Бытовой потребитель) управляет процессом **Dish Washing** (Мытье посуды).
Процесс **Dish Washing** распадается на подпроцессы **Dish Loading** (Загрузка посуды), **Dish Cleaning** (Очистка посуды) и **Dish Unloading** (Выгрузка посуды) в указанной последовательности. Процесс **Dish Loading** (Загрузка посуды) изменяет состояние объекта **Dishwasher** (Посудомоечная машина) из состояния **empty** (незагруженное) на состояние **loaded** (загруженное). Процесс **Detergent Inserting** (Введение моющего средства) требует объекта **Soap** (Мыло).

Окончание таблицы 26

Процесс **Detergent Inserting** (Введение моющего средства) изменяет состояние объекта **Soap Compartment** из состояния **empty** (незагруженное) на состояние **loaded** (загруженное). Процесс **Dish Cleaning & Drying** (Очистка & Просушка посуды) требует объекта **Dishwasher** (Посудомоечная машина). Процесс **Dish Cleaning & Drying** (Очистка & Просушка посуды) потребляет объект объекта **Soap** (Мыло). Процесс **Dish Cleaning & Drying** (Очистка & Просушка посуды) изменяет состояние объекта **Cleanliness** (Чистота) для объекта **Dish Set** (Партия посуды) из состояния **dirty** (грязное) на состояние **clean** (чистое).
Процесс **Dish Unloading** (Выгрузка посуды) изменяет состояние объекта **Dishwasher** (Посудомоечная машина) из состояния **empty** (незагруженное) на состояние **loaded** (загруженное). Конец OPL-спецификации на **Dish Washing System** (Посудомоечная система)

14.2.3 OPM-принцип согласованности фактов

OPM-принцип согласованности фактов предусматривает, что:

a) модель факта, проявляющаяся на одной OPD-диаграмме, должна быть истинной и для всей совокупности OPD-диаграмм в рамках данной модели OPM-системы, и

b) никакая OPD-диаграмма на древе OPD-процессов или на древе OPD-объектов не должна содержать модели фактов, которые будут противоречить моделям фактов на той же или на любой другой OPD-диаграмме.

Факт на одной OPD-диаграмме может быть уточненным или обобщенным фактом на любой другой OPD-диаграмме (в рамках той же модели OPM-системы).

Примечание — Данный принцип не исключает возможности представления любого элемента модели произвольное число раз на стольких OPD-диаграммах, сколько этого пожелает разработчик модели. Поскольку связь не может существовать без сущностей, которые она соединяет (если эта связь существует), то также необходимо представлять две сущности на концах этой связи.

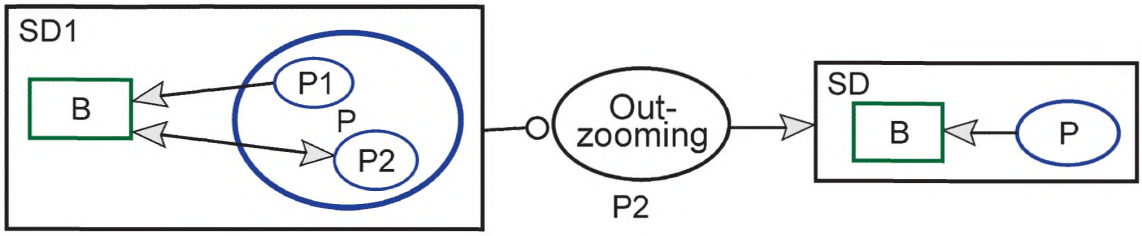
Пример — Невозможно для одной OPD-диаграммы выражать тот факт, что «P создает A» и для той же самой или другой OPD-диаграммы на том же OPD-древе выразить тот же факт, что «P потребляет A». Тем не менее для одной OPD-диаграммы допустимо выражать тот факт, что «P взаимодействует с A», а для другой OPD-диаграммы на том же OPD-древе, чтобы выразить тот факт, что «P изменяет состояние A с S1 на S2», поскольку последний факт является уточнением, не противоречащим первому.

14.2.4 Устранение обобщенной неоднозначности для процедурных связей

14.2.4.1 Обобщение и приоритетность процедурной связи

Обобщение (out-zooming) абстрагирует набор взаимосвязанных сущностей — объектов типа *refinees* и связанных с ними связей в сущности типа *refineable*. Когда разработчик модели выполняет абстрагирование, процедурные связи между объектами типа *refinees* и сущностями, которые не являются объектами типа *refinees*, должны перемещаться по контексту OPD-диаграммы, на которой изображены сущности типа *refineable*. Это перемещение может приводить к ситуации, в которой две или более процедурные связи различных видов связывают объект и процесс. В соответствии с OPM-принципом уникальности процедурной связи (см. 8.1.2) объект или его состояние должны связываться с процессом только с помощью одной процедурной связи. Во исполнение этого принципа разработчик модели должен разрешить конфликт между потенциальными связями и определить, какая связь будет оставаться или какая новая связь будет заменять другую на обобщенной OPD-диаграмме. Потеря более детальной информации согласуется с понятием абстракции (обобщения).

Пример — Рисунок 55 иллюстрирует проблему обобщения процедурной связи. На структурной диаграмме SD1 результирующая связь между подпроцессом P1 и объектом B является более значимой, чем воздействующая связь между подпроцессом P2 и объектом B, поэтому если структурная диаграмма SD1 является обобщением структурной диаграммы SD, результирующая связь становится приоритетной.



Out-zooming = Обобщение

Рисунок 55 — Диаграмма обобщающих процедурных связей

Семантическая сила (semantic strength) и приоритетность связи (link precedence) — это два понятия для определения того, какие связи сохранять и какие скрывать, когда OPD-диаграмма обобщается или сворачивается.

Семантическая сила процедурной связи должна определяться значимостью информации, которую эта связь переносит. Информация, касающаяся изменений в части существования, создания или ликвидации, является более значимой, чем информация об изменениях в существующих сущностях. Относительная семантическая сила двух противоречащих друг другу процедурных связей должна определять приоритетность этих связей. Если две или более процедурные связи конкурируют за их сохранение на уточненной или обобщенной OPD-диаграмме, то будет оставаться связь с наивысшей семантической силой (приоритетом).

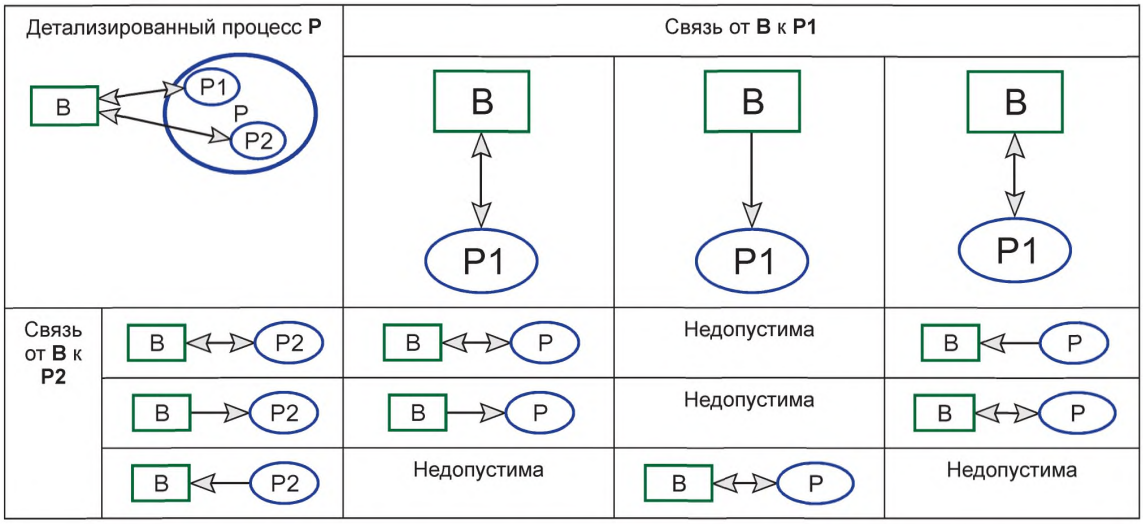
Примечание — Концепция приоритетности связи позволяет разработчику модели разрешать конфликты при представлении различных OPD-контекстов и устанавливать соответствующие процедурные связи с той или иной степенью детализации.

14.2.4.2 Приоритетность среди преобразующих связей

Преобразующие связи включают в себя результирующие, воздействующие и потребительские связи. Поскольку создание и потребление объекта семантически сильнее (т. е. они имеют большую силу, чем воздействие на объект в части изменения его состояния), то результирующая и потребительская связи будут иметь приоритет над воздействующими связями (см. рисунок 55), однако результирующие и потребительские связи семантически эквивалентны, когда они конкурируют; приоритетной связью должна быть воздействующая связь, поскольку она может рассматриваться как неявное изменение объекта из существующего состояния в несуществующее состояние, или наоборот.

Таблица 27 иллюстрирует приоритетность преобразующей связи: процесс P в верхнем левом углу является обобщенным. Заголовки столбцов показывают три возможные преобразующие связи между процессом P1 и объектом B, заголовки строк показывают три возможные связи между подпроцессом P2 и объектом B. В ячейках этой таблицы приведены преобладающие связи между объектом B и процессом P после того, как процесс P будет обобщен. Точнее, в таблице 27 показано, как разрешается конфликт между результирующей, воздействующей и потребительской связями. Например, если связь между объектом B и подпроцессом P1 является потребительской (см. средний столбец), а связь между объектом B и подпроцессом P2 является результирующей (см. нижнюю строку), то после того, как процесс P будет обобщен (out-zoomed), связь между объектом B и процессом P станет воздействующей. Ячейки таблицы, помеченные как «недопустимые» (invalid) указывают на невозможность подобного сочетания. Например, если проверить центральную ячейку, то можно отметить, что если подпроцесс P1 потребляет объект B, то объект B перестанет существовать, когда подпроцесс P2 позднее попытается повторно потребить этот объект. Таким образом, сочетание из двух потребительских связей является недопустимым.

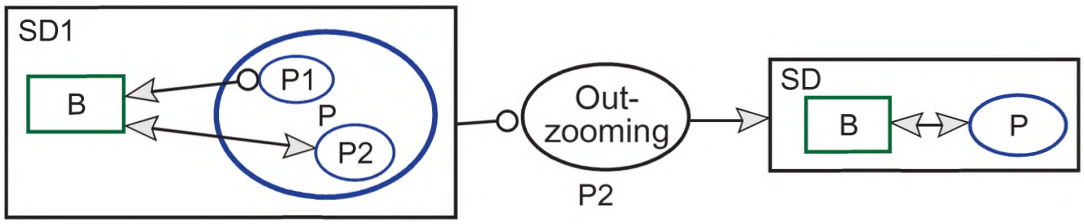
Таблица 27 — Приоритетность преобразующих связей: разрешение конфликтов между воздействующей, результирующей и потребительской связями



14.2.4.3 Приоритеты для преобразующих и разрешающих связей

Преобразующие связи семантически более сильные, чем разрешающие связи, поскольку первые связаны с созданием, потреблением или изменением связанных с ними объектов, тогда как разрешающие связи предназначены лишь для того, чтобы обозначить условия реализации, поэтому преобразующая связь должна иметь приоритет перед разрешающей связью (см. рисунок 56).

В рамках разрешающих связей агентская связь должна иметь приоритет над инструментальной связью, поскольку в искусственных системах люди играют основную роль в процессах, обеспечивая надлежащую работу этой системы. Кроме того, везде, где есть взаимодействие между людьми, должен существовать интерфейс, а информация о сущности должна быть доступна для разработчика сущности *refineable*, чтобы он мог осуществлять соответствующее планирование.



Out-zooming = Обобщение

Рисунок 56 — Приоритеты для преобразующих и разрешающих связей

Подводя итог рассмотрению семантической силы процедурных неуправляющих связей, преимущественный порядок приоритетов для этих связей таков: потребительская = результирующая > воздействующая > агентская > инструментальная, где знаки «=» и «>» относятся к семантической силе связей. Определяющие состояние связи должны обладать наивысшим приоритетом над не определяющими состояние связями.

14.2.4.4 Вторичные приоритеты среди неуправляющих и управляющих связей одного рода

Почти каждый вид неуправляющих связей имеет соответствующее событие и состояние связи, которые полезны для определения более точных, вторичных приоритетов в рамках каждого вида процедурных связей. Относительную семантическую силу этих вторичных приоритетов в каждом элементе с первичным приоритетом должна иметь семантическая связь с самой высокой семантической силой,

чем у соответствующей неуправляющей связи, тогда как условная связь должна иметь меньшую семантическую силу, чем у соответствующей неуправляющей связи.

Семантическая сила ситуационной связи должна быть выше, чем у соответствующей неуправляющей связи, поскольку любая ситуационная связь обладает семантикой соответствующей неуправляющей связи плюс событие, способное инициализировать соответствующий процесс. Семантическая сила условной линии связи должна быть слабее, чем у соответствующей неуправляющей связи, поскольку модификатор состояния ослабляет критерии удовлетворенности предварительными условиями для соединяющего процесса.

14.2.4.5 Обзор семантической силы процедурных связей

Подводя итог рассмотрению семантической силы процедурных связей, основанному на различиях между первичными и вторичными приоритетами, полный порядок приоритетности должен быть следующим:

1.	событие потребления	>	потребление
2.	потребление	=	результат
3.	результат	>	условие потребления
4.	условие потребления	>	воздействующее событие
5.	воздействующее событие	>	воздействие
6.	воздействие	>	условие воздействия
7.	условие воздействия	>	агентское событие
8.	агентское событие	>	агент
9.	агент	>	агентское условие
10.	агентское условие	>	инструментальное событие
11.	инструментальное событие	>	инструмент
12.	инструментальное средство	>	инструментальное условие

Приложение А
(обязательное)

Формальный OPL-синтаксис в расширенной форме Бэкуса — Наура (EBNF)

А.1 Общие положения

OPL-язык является подвидом английского языка, который призван текстуально выражать OPM-спецификацию, которая в графическом виде выражается совокупностью OPD-диаграмм.

OPL-язык является языком двойного назначения. Во-первых, он используется экспертами предметной области и системными архитекторами, занимающимися анализом и проектированием таких систем, как системы электронной торговли или веб-системы планирования ресурсов предприятия (web-based ERP-system). Во-вторых, он создает прочную основу для автоматического формирования проектируемых приложений.

OPL-язык является текстовым аналогом графической спецификации OPM-системы, которая соответствует схематичному описанию в виде OPD-диаграмм. OPL-язык должен быть автоматически формируемым текстовым описанием системы, генерируемым в подмножество естественного английского языка. После лишения специфических особенностей и чрезмерно усложненных деталей, которые характерны для языков программирования, OPL-предложения должны стать понятными для людей без технического опыта или опыта программирования.

Из-за большого разнообразия возможностей выражения моделей, допускаемого OPM-методологией, выражение OPL-синтаксиса с использованием расширенной формы Бэкуса — Наура (EBNF), рассмотренное ниже, не является исчерпывающим, например, для утверждений, касающихся вероятности (см. 12.7) и управления путями выполнения (см. раздел 13), отсутствуют EBNF-выражения. Огромное разнообразие ограничений на участие, в особенности ограничений, представимых в виде математических формул, не имеют формального описания в данном приложении.

А.2 OPL-язык в контексте OPD-диаграммы

Данное приложение содержит формальное описание OPL-языка, соответствующее ИСО/МЭК 14977, что вызывает появление различных графических OPD-структур (описанных в разделах 7—14). Для облегчения понимания данного приложения в нем, если это возможно, приведены ссылки на соответствующие OPD-подразделы, заголовки разделов/подразделов, что поможет разделить EBNF в соответствии с синтаксическими формами для моделирования элементов.

А.3 Предварительные сведения

А.3.1 EBNF-синтаксис

В нижеописанном синтаксисе используются EBNF-обозначения согласно ИСО/МЭК 14977. Стандартные символы, представляющие каждый оператор в расширенной форме Бэкуса — Наура и его подразумеваемый приоритет, должны быть следующими (с наивысшим приоритетом сверху):

«*» — символ повторения;
«–» — символ исключения;
«,» — символ соединения (конкатенации);
«|» — символ определения-разделителя;
«=» — символ определения;
«;» — символ окончания (ограничения).

Стандартный приоритет должен представляться с помощью следующих пар скобок:

«'» — символ одинарной кавычки;
«"» — символ двойной кавычки;
«(*» — символ начала комментария; «*)» — символ конца комментария;
«(» — символ начала группы; «)» — символ конца группы;
«[» — символ начала опции; «]» — символ конца опции;
«{» — символ начала повтора; «}» — символ конца повтора;
«?» — символ специальной последовательности.

Примечание 1 — Символ пробела, заключенный в кавычки " ", будет означать, что требуется литеральный символ пробела; в противном случае символы пробела и символы окончания строки (так называемое свободное пространство) не будут иметь никакого значения.

Примечание 2 — Мета-идентификатор может находиться как с левой, так и с правой стороны от правила, тем самым обеспечивая рекурсию.

Примечание 3 — Символ одиночной кавычки определяет синтаксические элементы изменяемых OPL-меток, которыми являются имена и значения, существующие в графических OPD-моделях и OPL-предложениях. Эти частные синтаксические элементы встречаются только в А.3.2.

Примечание 4 — Символ двойной кавычки определяет синтаксические элементы OPL-постоянных, которыми являются слова и фразы, существующие в OPL-предложениях как интерпретации графических конфигураций элементов и тегов связей на OPD-диаграммах.

Примечание 5 — Начиная с А.3.2 и заканчивая оставшейся частью приложения А, весь его текст (за исключением заголовков) соответствует ИСО/МЭК 14977.

А.3.2 Основные определения

(* Область OPL EBNF *)

(*Основные определения в этой области: Следующие основные определения области задаются определенными строками: *)

ненулевое число = '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;

десятичное число = '0' | ненулевое число ;

целое положительное число = ненулевое число, {десятичное число} ;

положительное действительное число = {десятичное число}, десятичное число, {десятичное число} " . ";

заглавная буква = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' ;

строчная буква = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' ;

буква = заглавная буква | строчная буква ;

символ строки = буква | десятичное число | '_' | '-' | '&' | '/' | ' ' ; (*отметим, что символом строки может быть пробел *)

имя = буква, { символ строки }; (* отметим, что первым символом здесь является буква *)

слово заглавными буквами = заглавная буква {символ строки} ;

слово строчными буквами = строчная буква { символ строки} ;

фраза строчными буквами = строчное слово, {", (строчное слово | заглавное слово)} ;

идентификатор типа = " булев "
| "строка "
| тип числа
| "перечислимый " ;

префикс = "беззнаковый " ;

тип числа = [префикс], "целое число "
| "плавающий "
| "двойной "
| " короткий "
| " длинный " ;

предельное число участников = целое положительное число | положительное действительное число ;

ограничение участников = нижнее одиночное
| верхнее одиночное
| нижнее множественное
| верхнее множественное
| ("0" | предельное число участников, ["до", предельное число участников]) ;

выражение ограничения = "где", имя, ((логическая операция, имя значения)
| (логическое начальное множество, (имя | имя значения), { ",", [(имя | имя значения)]}, логическое конечное множество)) ;

нижнее одиночное = "a" | " an " | "дополнительное" | " как минимум одно " ;

верхнее одиночное = "A" | " An " | " дополнительное " | " как минимум одно " ;

нижнее множественное = " дополнительное " | "множественное " ;

верхнее множественное = " Дополнительное " | " Множественное " ;

диапазон условий = "является", имя значения | "находится от", имя значения, "до", имя значения;

логическая операция = "=" | "<" | ">" | "<=" | "> =" ;

логическое начальное множество = "в { " ;

логическое конечное множество = " }" ;

(* ограничения участия имеют множество форм выражения, а основные определения не содержат все эти формы. *)

(* зарезервированные слова и символы, содержащиеся в OPL-утверждениях, разделяются вторыми символами двойных кавычек *)

(* EndRegion: Основные определения *)

А.3.3 Специальные OPL-последовательности

(* Область: — Эта область определяет все специальные последовательности типа New Line (Новая строка), Plural Objects (Множественное число объектов) и Processes (Процессы) *)

новая строка = ? специфическая для данного приложения последовательность символов, определяющая символ новой строки, с последующим возвратом на первую позицию на строке ? ;

единица измерений = ? любая указанная или стандартно воспринимаемая единица измерения времени, пространства, количества или качества ? ;

имя значения = ? число или имя, приемлемые для соответствующей единицы измерения ? ;

имя единичного объекта = ? словосочетание для единичного объекта заглавными буквами ? ; (* см. 7.1.2 *)

имя множественного объекта = ? словосочетание для множественного объекта заглавными буквами ? ;

имя единичного процесса = ? деепричастная фраза заглавными буквами ? | ? именное словосочетание для единичного процесса заглавными буквами ? ;

имя множественного процесса = ? деепричастная фраза заглавными буквами ? | ? именное словосочетание для множественного процесса заглавными буквами ? ; (* см. 7.2.2 *)

родительская OPD-диаграмма = ? OPD-диаграмма, из которой происходит новая детализированная диаграмма или новая разворачиваемая диаграмма ? ;

дочерняя OPD-диаграмма =? OPD-диаграмма, получаемая в результате детализации или развертывания родительской OPD-диаграммы ? ;

единицы измерения максимального промежутка времени = ? значение максимальной продолжительности выполнения данного процесса, выражаемой в единицах времени ? ;

единицы измерения минимального промежутка времени = ? значение минимальной продолжительности выполнения данного процесса, выражаемой в единицах времени ? ;

(* EndRegion: Специальные последовательности *)

А.4 OPL-синтаксис

А.4.1 Структура OPL-документа

(* Область: OPL-документ *)

OPL раздел = OPL предложение, {новая строка, OPL предложение} ;

OPL предложение = OPL формальное предложение, "." ;

OPL формальное предложение = описательное предложение сущности
| процедурное предложение
| структурное предложение
| предложение для управления контекстом;

А.4.2 OPL-идентификаторы

(* Область: Идентификаторы — Эта область определяет все идентификаторы, используемые в данной грамматике *)

идентификатор объекта = имя единичного одиночного объекта, ["в", единица измерений], [диапазон пунктов]
| имя единичного объекта, "объект", ["в", единица измерений], [диапазон пунктов]
| имя множественного объекта, ["в", единица измерений], [диапазон пунктов]
| имя множественного объекта, "объекты", ["в", единица измерений], [диапазон пунктов] ;

идентификатор процесса = имя единичного процесса
| имя единичного процесса, "процесс"
| имя множественного процесса
| имя множественного процесса, "процессы" ;

идентификатор сущности = идентификатор объекта
| идентификатор процесса; (* см. 7.1 и 7.2 *)

идентификатор состояния = слово строчными буквами ;

выражение тега = фраза строчными буквами ;

(* EndRegion: Идентификаторы *)

А.4.3 OPL-перечни

(* Область: Перечни — Эта область определяет различные перечни: перечень объектов, перечень процессов, перечень объектов с дополнительными состояниями *)

перечень процессов = идентификатор процесса
| идентификатор процесса, [{ ",", идентификатор процесса}], "и", идентификатор процесса; (* см. 12.1 *)

перечень процессов типа Or = идентификатор процесса, [{ ",", идентификатор процесса}], "или", идентификатор процесса ;

перечень процессов типа Xor в начале = "Один из", перечень процессов Or;

перечень процессов типа Xor в конце = "один из", перечень процессов Or;

перечень объектов = идентификатор объекта
| идентификатор объекта, [{ "," идентификатор объекта}], "и", идентификатор объекта; (* см. 12.1 *)
объект с дополнительным состоянием = [идентификатор состояния, " "], идентификатор объекта;
(* объект с перечнем дополнительных состояний может заменять идентификатор объекта во многих OPL-выражениях, используя для этого идентификатор объекта *)
объект с перечнем дополнительных состояний = объект с дополнительным состоянием
| объект с дополнительным состоянием, [{ "," объект с дополнительным состоянием}], "и", объект с дополнительным состоянием ;
перечень объектов типа Or = объект с дополнительным состоянием, [{ "," объект с дополнительным состоянием}], "или", объект с дополнительным состоянием; (* см. 12.2 *)
перечень объектов не в состоянии Or = идентификатор объекта, [{ "," идентификатор объекта}], "или", идентификатор объекта;
перечень объектов типа Xor в начале = "Один из", перечень объектов Or;
перечень объектов типа Xor в конце = "один из", перечень объектов Or;
перечень объектов не в состоянии Or в конце = "один из", перечень объектов;
перечень состояний = идентификатор состояния
| идентификатор состояния, [{ "," идентификатор состояния}], "и", идентификатор состояния ;
перечень состояний типа Or = идентификатор состояния, [{ "," идентификатор состояния}], "или", идентификатор состояния ;
перечень состояний типа Xor в конце = "один из", перечень состояний типа Or ;
(* EndRegion: Перечни *)

A.4.4 OPL-описание сущности

A.4.4.1 Предложения для описания сущностей

(* Область: Описание сущности — Эта область определяет все предложения, используемые для описания сущности *)
предложение для описания сущности = общее свойство предложения
| тип предложения для описания
| состояние предложения для описания ;

A.4.4.2 Общее свойство предложения

общее свойство предложения = идентификатор сущности ,
" есть ", [сущность], [принадлежность], [устойчивость] ; (* см. 7.3.3 *)
сущность = " Информационная " | " Физическая " ; (* Физическая сущность — это значение сущности, устанавливаемое не по умолчанию; значением по умолчанию которой является Информационное *)
принадлежность = "Системная | " Внешней среды " ; (* Принадлежность внешней среде — это значение принадлежности, устанавливаемое не по умолчанию; значением по умолчанию которой является Системная *)
устойчивость = " Постоянная " | " Временная "; (* Временная устойчивость — это значение устойчивости, устанавливаемое не по умолчанию; значением по умолчанию которой является Постоянное *)

A.4.4.3 Тип предложения

тип предложения для описания = объект, "является типом", идентификатор типа ;

A.4.4.4 Состояния предложения

состояние предложения для описания = состояние перечислимого предложения
| начальные состояния предложения
| конечные состояния предложения
| состояние предложения по умолчанию
| сложное состояние предложения ; (* см. 7.3.5 *)
состояние перечислимого предложения = идентификатор объекта, " есть ", идентификатор состояния
| идентификатор объекта, " может быть ", состояние идентификатора, [{ "," идентификатор состояния}], " и ", идентификатор состояния
| идентификатор объекта, " может быть ", состояние идентификатор, [{ "," идентификатор состояния}], " и другие состояния " ;
начальные состояния предложения = единственное начальное состояние предложения
| множественные начальные состояния предложения ;
единственное начальное состояние предложения = " Состояние ", идентификатор состояние, " от ", идентификатор объекта, " является начальным " ;
множественные начальные состояния предложения = "Состояния", перечень состояний " от ", идентификатор объекта, " являются начальными " ;
конечные состояния предложения = единственное конечное состояние предложения
| множественные начальные состояния предложения ;
единственное начальное состояние предложения = " Состояние ", идентификатор состояния, " от ", идентификатор объекта, "является начальным" ;

множественные начальные состояния предложения = " Состояния ", перечень состояний " от ", идентификатор объекта, "являются начальными" ;

состояние по умолчанию предложения = " Состояние ", идентификатор состояния, " от ", идентификатор объекта, "является состоянием по умолчанию" ;

сложное состояние предложения = идентификатор объекта, {" является изначальным ", [идентификатор состояния | идентификатор состояния,

{ " и ", идентификатор состояния }], " и конечное ", перечень состояний типа OR } ;

входное состояние = идентификатор состояния; (* состояние или состояния связанного объекта в априорном наборе процесса *)

выходное состояние = идентификатор состояния; (* состояние или состояния связанного объекта в апостериорном наборе процесса *)

идентификатор активного процесса = идентификатор процесса ;

(* EndRegion: Описание сущности *)

А.4.5 OPL-процедурные предложения

А.4.5.1 Процедурное предложение (Procedural sentences)

(* Область: Процедурные предложения — Эта область определяет все процедурные предложения *)

процедурное предложение = преобразующее предложение

| разрешающее предложение (enabling sentence)

| управляющее предложение (control sentence) ; (* см. 8.1.1 *)

А.4.5.2 OPL-преобразования

А.4.5.2.1 Преобразующее предложение (Transforming sentence)

(* Область: Преобразующие предложения — Эта область определяет потребительские, результирующие, воздействующие и изменяющие предложения, а также их различные варианты *)

преобразующее предложение = потребляющее предложение

| результирующее предложение (result sentence)

| воздействующее предложение (effect sentence)

| изменяющее предложение (change sentence) ; (* см. 9.1.1 и 9.3.3 *)

А.4.5.2.2 Потребительское предложение (Consumption sentence)

потребительское предложение = (идентификатор процесса, " потребляет ", объект с дополнительным перечнем состояний)

| потребительское предложение выбора; (* см. 9.1.2 *)

потребительское предложение выбора = потребительское предложение типа Or

| потребительское предложение типа Xor ; (* см. 12.3 *)

потребительское предложение типа Or = исходное потребительское предложение типа Or

| целевое потребительское предложение типа Or ;

исходное потребительское предложение типа Or = идентификатор процесса, " потребление по меньшей мере одного из ", перечень объектов типа Or ;

целевое потребительское предложение типа Or = " По меньшей мере один из ", перечень процессов типа Or, " потребляет ", объект с дополнительным состоянием ;

потребительское предложение типа Xor = исходное потребительское предложение типа Xor

| целевое потребительское предложение типа Xor ;

исходное потребительское предложение типа Xor = идентификатор процесса, " потребление ровно ", перечень объектов типа Xor в конце ;

целевое потребительское предложение типа Xor = " Точно ", перечень процессов типа Xor в конце, " потребление ",

объект с дополнительным состоянием ;

А.4.5.2.3 Результирующее предложение (Result sentence)

результирующее предложение = (идентификатор процесса, " дает ", объект с дополнительным перечнем состояний)

| результирующее предложение выбора ; (* см. 9.1.3 *)

результирующее предложение выбора = результирующее предложение типа Or

| результирующее предложение типа Xor ; (* см. 12.3 *)

результирующее предложение типа Or = исходное результирующее предложение типа Or

| целевое результирующее предложение типа Or ;

исходное результирующее предложение типа Or = " по меньшей мере один из ", перечень процессов типа Or, " дает ", объект с дополнительным состоянием;

целевое результирующее предложение типа Or = идентификатор процесса, " дает по меньшей мере один из ", перечень объектов типа Or ;

результирующее предложение типа Xor = исходное результирующее предложение типа Xor

| целевое результирующее предложение типа Xor ;

исходное результирующее предложение типа Xor = " точно ", перечень процессов типа Xor в конце, " дает ", объект с дополнительным состоянием ;

целевое результирующее предложение типа Xor = идентификатор процесса, " дает точно ", перечень объектов типа Xor в конце ;

A.4.5.2.4 Воздействующее предложение (Effect sentence)

воздействующее предложение = (идентификатор процесса, " взаимодействует ", перечень объектов)

| воздействующее предложение выбора; (* см. 9.1. 4 *)

воздействующее предложение выбора = воздействующее предложение типа Or

| воздействующее предложение типа Xor ;

воздействующее предложение типа Or = воздействующее предложение типа Or для объекта

| воздействующее предложение типа Or для процесса ; (* см. 12.3 *)

воздействующее предложение типа Or для объекта = идентификатор процесса, " воздействует по меньшей мере на один из ", перечень объектов типа Or, не имеющих состояния ;

воздействующее предложение типа Or для процесса = " По меньшей мере один из ", перечень процессов типа Or, " взаимодействует ", идентификатор объекта ;

воздействующее предложение типа Xor = действующее предложение типа Xor для объекта

| воздействующее предложение типа Xor для объекта ;

воздействующее предложение типа Xor для объекта = идентификатор процесса, " воздействует точно ", перечень объектов типа Xor, не имеющих состояния, в конце ;

воздействующее предложение типа Xor для процесса = " точно ", перечень процессов типа Xor в конце, идентификатор объекта ;

A.4.5.2.5 Изменяющее предложение (Change sentence)

изменяющее предложение = специфицированное предложение, изменяющее вход/выход

| специфицированное предложение, изменяющее вход

| специфицированное предложение, изменяющее выход ; (* см. 9.3.3.1 *)

специфицированное предложение, изменяющее вход/выход = (идентификатор процесса, " изменяет ", в перечне изменений объекта на входе/выходе)

| предложение выбора с перечнем специфицированных изменений объекта на входе/выходе ; (* см. 9.3.3.2 *)

предложение с перечнем изменений объекта на входе/выходе = фраза с изменениями объекта на входе/выходе

| фраза с изменениями объекта на входе/выходе, [{"," , фраза с изменениями объекта на входе/выходе a}], " и ", фраза с изменениями объекта на входе/выходе ;

фраза с изменениями объекта на входе/выходе = идентификатор объекта, " от ", входное состояние, " к ", выходное состояние ;

предложение выбора со специфицированными изменениями объекта на входе/выходе = предложение типа Or со специфицированными изменениями объекта на входе/выходе

| предложение типа Xor со специфицированными изменениями объекта на входе/выходе ;

предложение типа Or со специфицированными изменениями объекта на входе/выходе = (идентификатор процесса, " изменяет ", перечень изменений объектов типа Or на входе/выходе)

| (перечень процессов типа Or, " изменяет ", фраза с изменениями объекта на входе/выходе)

| предложение типа Or с определенными изменениями объекта на входе/выходе ;

предложение типа Or со специфицированными изменениями объекта на входе/выходе = (идентификатор процесса, " изменяет ", перечень изменений объектов типа Or на входе/выходе);

предложение типа Or со специфицированными изменениями объекта на входе/выходе = (идентификатор процесса, " изменяет ", идентификатор объекта, " из ", перечень состояний типа Or.

| (идентификатор процесса, " изменяет ", идентификатор объекта, " из ", идентификатор состояния, " в ", перечень состояний типа Or) ;

предложение типа Xor со специфицированными изменениями объекта на входе/выходе = предложение с определенными изменениями объекта типа Xor на входе/выходе

| предложение со специфицированными изменениями процесса типа Xor на входе/выходе

| предложение со специфицированными изменениями состояния типа Xor на входе/выходе ;

i предложение со специфицированными изменениями объекта типа Xor на входе/выходе = идентификатор процесса, " изменяет один из ", перечень изменений объекта типа Or на входе/выходе ;

предложение со специфицированными изменениями процесса типа Xor на входе/выходе = перечень процессов типа Xor в начале, " изменяет ", фраза с изменениями объекта на входе/выходе ;

предложение со специфицированными изменениями состояния типа Xor на входе/выходе, (идентификатор процесса, " изменяет ", идентификатор объекта, " от ", перечень состояний типа Xor в конце, " в ", идентификатор состояния)

| (идентификатор процесса, " изменяет ", идентификатор объекта, " от ", идентификатор состояния, " в ", перечень состояний типа Xor в конце) ;

предложение, изменяющее вход = (идентификатор процесса, " изменяет ", перечень изменений на входе объекта)

| специфицированное предложение выбора, изменяющее вход ; (* см. 9.3.3.3 *)

фраза, изменяющая вход объекта = идентификатор объекта, "от", состояние входа ;

перечень изменений объекта на входе = фраза с изменениями на входе объекта

| фраза с изменениями на входе объекта, [{ ", ", фраза с изменениями на входе объекта }], " и ", фраза с изменениями на входе объекта ;

специфицированное предложение выбора, изменяющее вход = определенное предложение выбора типа Or с изменениями на входе

| специфицированное предложение типа Xor с изменениями на входе ;

специфицированное предложение типа Or, изменяющее вход = (идентификатор процесса, " изменяет ", или список изменений на входе объекта)

| (перечень процессов типа Or, " изменяет ", фраза с изменениями на входе объекта)

| (идентификатор процесса, " изменяет ", идентификатор объекта, "от", перечень состояний типа Or) ;

перечень объектов типа Or на входе = фраза, изменяющая вход объекта, [{ ", ", фраза, изменяющая вход объекта }], " или ", фраза с изменениями на входе объекта;

специфицированное предложение типа Xor, изменяющее вход = (идентификатор процесса, " изменяет один из ", перечень изменений объектов типа Or на входе)

| (перечень процессов типа Xor в начале, " изменяет ", фраза с изменениями на входе объекта)

| (идентификатор процесса, " изменяет ", идентификатор объекта, " от ", перечень состояний типа Xor в конце) ;

специфицированное предложение, изменяющее выход = (идентификатор процесса, " изменяет ", перечень изменений объектов на выходе)

| специфицированное предложение, изменяющее выход; (* см. 9.3.3.4 *)

перечень изменений объектов на выходе = фраза с изменениями объектов на выходе

| фраза с изменениями объектов на выходе, [{ ", " фраза с изменениями объектов на выходе}], " и ",

фраза с изменениями объектов на выходе = идентификатор объекта, " к ", состояние выхода;

специфицированное предложение, изменяющее выход = специфицированное предложение, изменяющее выход

| специфицированное предложение типа Xor, изменяющее выход ;

специфицированное предложение выбора типа Or, изменяющее выход = (идентификатор процесса, " изменяет ", перечень изменений объектов типа Or на выходе)

| (перечень процессов типа Or, " изменяет ", перечень изменений объектов типа Or на выходе)

| (идентификатор процесса, " изменяет ", идентификатор объекта, " в ", перечень изменений состояния типа Or);

перечень изменений объектов типа Or на выходе = фраза с изменениями объекта на выходе, [{ ", ", фраза с изменениями объекта на выходе }], " или ", фраза с изменениями объекта на выходе;

специфицированное предложение типа Xor = (идентификатор процесса, " изменяет один из ", перечень изменений состояния типа Or)

| (перечень процессов типа Or в начале, " изменяет ", перечень изменения процессов типа Or на выходе)

| идентификатор процесса, " изменяет ", идентификатор объекта, " в ", перечень состояний типа Or в конце;

(* EndRegion: Преобразующие предложения *)

A.4.5.3 OPL-средства реализации

A.4.5.3.1 Разрешающие предложения (Enabling sentences)

(* Область: Разрешающие предложения — Эта область определяет агентские и инструментальные предложения и их возможные варианты *)

разрешающее предложение = агентское предложение

| инструментальное предложение; (* см. 9.2.1 *)

A.4.5.3.2 Агентские предложения (Agent sentence)

агентское предложение = (объект с дополнительным перечнем состояний, " обрабатывает ", идентификатор процесса)

| агентское предложение выбора ; (* см. 9.2.2 и 12.3 *)

агентское предложение выбора = агентское предложение типа Or

| агентское предложение типа Xor;

агентское предложение типа Or = исходное агентское предложение типа Or

| целевое агентское предложение типа Or ;

исходное агентское предложение типа Or = " По крайней мере один из ", перечень объектов типа Or, " обрабатывает ", идентификатор процесса;

целевое агентское предложение типа Or = объект с дополнительным состоянием, " обрабатывает по меньшей мере один из ", перечень процессов типа Or;

агентское предложение типа Xor = исходное агентское предложение типа Xor

| целевое агентское предложение типа Xor ;

исходное агентское предложение типа Xor = "Точно", перечень объектов типа Xor в конце, " обрабатывает ", идентификатор процесса;

целевое агентское предложение типа Xor = объект с дополнительным состоянием, " обрабатывает точно ", перечень объектов типа Xor в конце;

A.4.5.3.3 Инструментальное предложение (Instrument sentence)

инструментальное предложение = (идентификатор процесса, " требует ", объект с дополнительным перечнем состояний)

| инструментальное предложение выбора; (* см. 9.2.3 и 12.3 *)

инструментальное предложение выбора = инструментальное предложение типа Or

| инструментальное предложение типа Xor ;

инструментальное предложение типа Or = исходное инструментальное предложение типа Or

| целевое инструментальное предложение типа Or ;

инструментальное предложение типа Or = идентификатор процесса, " требует по меньшей мере одного из ", перечень объектов типа Or;

целевое инструментальное предложение типа Or = " По мере один из ", перечень процессов типа Or, " требует ", объект с дополнительным состоянием;

инструментальное предложение типа XOR = исходное инструментальное предложение типа Xor

| целевое инструментальное предложение типа Xor;

исходное инструментальное предложение типа Xor = идентификатор процесса, " требует точно ", перечень объектов типа Xor в конце;

целевое инструментальное предложение типа XOR = " Точно ", перечень процессов типа Xor, " требует ", объект с дополнительным состоянием;

(* EndRegion: Разрешающие предложения *)

A.4.5.4 OPL-поток сигналов управления

A.4.5.4.1 Управляющее предложение (Control sentence)

(* Область: Управляющие предложения — Эта область определяет все предложения, связанные с потоком сигналов управления в системе *)

управляющее предложение = ситуационное предложение

| условное предложение

| инициализирующее предложение

| исключающее предложение; (* см. 9.5.1 *)

A.4.5.4.2 Ситуационное предложение (Event sentence)

ситуационное предложение = потребительское ситуационное предложение

| воздействующее ситуационное предложение

| агентское ситуационное предложение

| инструментальное ситуационное предложение ; (* см. 9.5.2 *)

потребительское ситуационное предложение = объект с дополнительным состоянием " инициирует ", идентификатор процесса, " который потребляет", идентификатор объекта; (* см. 12.5 и 12.6 для ознакомления с дополнительным синтаксисом, связанным с веерными связями *)

воздействующее ситуационное предложение = простое воздействующее ситуационное предложение

| определяемое входом/выходом воздействующее ситуационное предложение

| определяемое входом воздействующее ситуационное предложение

| определяемое выходом воздействующее ситуационное предложение ;

простое воздействующее ситуационное предложение = идентификатор объекта, " инициирует ", идентификатор процесса, " который воздействует ", идентификатор объекта;

определяемое входом/выходом воздействующее ситуационное предложение = состояние входа, идентификатор объекта, " инициирует ", идентификатор процесса, " который изменяет ", фраза с изменением объекта на входе/выходе;

определяемое входом воздействующее ситуационное предложение = состояние входа, идентификатор объекта, " инициирует ", идентификатор процесса, " который изменяет ", идентификатор объекта, " из ", состояние входа;

определяемое выходом воздействующее ситуационное предложение = идентификатор объекта, " инициирует в любом состоянии ", идентификатор процесса, " который изменяет ", идентификатор объекта, " на ", состояние выхода;

агентское ситуационное предложение = объект с дополнительным состоянием, " инициирует и обрабатывает ", идентификатор процесса;

инструментальное событийное предложение = объект с дополнительным состоянием " инициирует ", идентификатор процесса, " который требует ", объект с дополнительным состоянием;

A.4.5.4.3 Условное предложение (Condition sentence)

условное предложение = условное преобразующее предложение

| условное разрешающее предложение ;

условное преобразующее предложение = условное потребительское предложение

| условное определяющее состояние потребительское предложение

| условное воздействующее предложение ; (* см. 9.5.3.1 и 9.5.3.3 *)

условное потребительское предложение = (идентификатор процесса, " возникает, если ", идентификатор объекта, " существует в том случае ", идентификатор объекта, " потребляется в противном случае ", идентификатор процесса, " пропускается ")

| (" если ", идентификатор объекта, " существует, то " идентификатор процесса, "происходит и потребляет", идентификатор объекта, " в противном случае обходится ", идентификатор процесса) ;

условное определяемое состоянием потребительское предложение = (идентификатор процесса, " происходит, если ", идентификатор объекта, " есть ", состояние входа, " в этом случае ", идентификатор объекта, " потребляется, в противном случае ", идентификатор процесса, " пропускается ")

| (" если ", состояние входа, идентификатор объекта, " существует, то ", идентификатор процесса, " происходит и потребляет ", идентификатор объекта, " в противном случае — обходится ", идентификатор процесса);

условное воздействующее предложение = простое условное воздействующее предложение

| определяемое входом/выходом условное действующее предложение

| определяемое входом условное действующее предложение

| определяемое выходом условное действующее предложение ;

простое условное воздействующее предложение = (идентификатор процесса, " происходит, если ", идентификатор объекта, " существует, и в этом случае ", идентификатор процесса, " воздействует ", идентификатор объекта, " в противном случае ", идентификатор процесса, " пропускается ")

| (" если ", идентификатор объекта, " существует, то ", идентификатор процесса, " происходит и воздействует ", идентификатор объекта, " в противном случае — обходится ", идентификатор процесса) ;

определяемое входом/выходом условное воздействующее предложение = (идентификатор процесса, " появляется, если существует ", состояние ввода, идентификатор объекта, " в этом случае ", идентификатор процесса, " изменяет ", фраза с изменениями входа/выхода объекта, " иначе ", идентификатор процесса, " пропускается ")

| (идентификатор процесса, " появляется, если существует ", состояние входа, идентификатор объекта, " в этом случае ", идентификатор процесса, " изменяет ", фраза с изменениями входа/выхода объекта, " в противном случае — обходится ", идентификатор процесса) ;

определяемое входом условное воздействующее предложение = (идентификатор процесса, " появляется, если существует ", состояние входа, идентификатор объекта, " в этом случае ", идентификатор процесса, " изменяет ", идентификатор объекта, " из ", состояние ввода, " иначе ", идентификатор процесса, " пропускается ")

| (идентификатор процесса, " появляется, если существует ", состояние входа, идентификатор объекта, " в этом случае ", идентификатор процесса, " изменяет ", идентификатор объекта " из " состояния ввода, фраза с изменениями входа/выхода объекта, " иначе ", идентификатор процесса " пропускается ") ;

определяемое выходом условное воздействующее предложение = (идентификатор процесса, " появляется, если существует ", идентификатор процесса, " изменяет ", идентификатор объекта, " к ", выходное состояние, ", в противном случае ", идентификатор процесса, " пропускается ")

| (идентификатор процесса, " появляется, если ", идентификатор объекта, " существует, и в этом случае ", идентификатор процесса, " изменяется ", идентификатор объекта, " на ", состояние выхода, " в противном случае — пропускается ", идентификатор процесса) ;

условное разрешающее предложение = условное агентское предложение

| условное инструментальное предложение; (* см. 9.5.3.2 *)

условное агентское предложение = (идентификатор процесса, " появляется, если ", объект с дополнительным состоянием, " существует, если ", идентификатор процесса, " пропускается ")

| (идентификатор процесса, " появляется, если ", объект с дополнительным состоянием, " существует, если обойти ", идентификатор процесса) ;

условное инструментальное предложение = (идентификатор процесса, " появляется, если " объект с дополнительным состоянием, " существует, если ", идентификатор процесса, " пропускается ")

| (Идентификатор процесса, " появляется, если ", объект с дополнительным состоянием, " существует, если обойти ", идентификатор процесса) ;

А.4.5.4.4 Инициализирующее предложение (Invocation sentence)

инициализирующее предложение = (идентификатор процесса, " инициирует ", перечень процессов)

| (идентификатор процесса, " инициирует сам себя ")

| инициализирующее предложение выбора; (* см. 9.5.2.5 и 12.3 *)

инициализирующее предложение выбора = инициализирующее предложение типа Or

| инициализирующее предложение типа Xor ;

инициализирующее предложение типа Or = (" по меньшей мере один из ", перечень процессов типа Or, " инициирует ", идентификатор процесса)

| (идентификатор процесса, " инициирует по меньшей мере один из ", перечень процессов типа Or) ;

инициализирующее предложение типа Xor = (" именно один из ", перечень процессов типа Or, " инициирует ", идентификатор процесса)

| (идентификатор процесса, " инициирует именно ", перечень процессов типа Xor в конце) ;

A.4.5.4.5 Исключающее предложение (Exception sentence)

исключающее предложение = дополнительное исключающее предложение

| неполное исключающее предложение ; (* см. 9.5.4 *)

дополнительное исключающее предложение = активный идентификатор процесса, " появляется, если длительность ", идентификатор процесса, " превышает ", единицы максимальной продолжительности времени;

неполное исключающее предложение = активный идентификатор процесса, " появляется, если длительность ", идентификатор процесса, " является недостаточным ", единицы минимальной продолжительности времени;

(* EndRegion: Контролируемые предложения *)

(* EndRegion: Процедурные предложения *)

A.4.6 Структурные OPL-предложения

A.4.6.1 Структурное предложение (Structural sentence)

(* Область: Структурные предложения — Эта область определяет все предложения, которые соединяют сущности в статические, не зависящие от времени, долгосрочные отношения *)

структурное предложение = тегированное структурное предложение

| групповое предложение (aggregation sentence)

| характеристическое предложение (characterization sentence)

| представительское предложение (exhibition sentence)

| специализационное предложение (specialization sentence)

| инстанциационное предложение (instantiation sentence); (* см. 10.1 *)

A.4.6.2 Тегированные OPL-структуры

A.4.6.2.1 Тегированное структурное предложение (Tagged structural sentence)

тегированное структурное предложение = однонаправленное тегированное структурное предложение

| двунаправленное тегированное структурное предложение ;

A.4.6.2.2 Однонаправленное тегированное структурное предложение (Unidirectional tagged structural sentence)

однонаправленное тегированное структурное предложение = однонаправленное тегированное предложение с единственной связью

| разветвленное тегированное структурное предложение; (* см. 10.2.1 и 11.2 *)

однонаправленное тегированное предложение с единственной связью = nullTag однонаправленное тегированное структурное предложение для объекта

| nullTag — однонаправленное тегированное структурное предложение для процесса

| non nullTag — однонаправленное тегированное структурное предложение для объекта

| non nullTag — однонаправленное тегированное структурное предложение для процесса; (* см. 10.2.2 и 11.2 *)

nullTag — однонаправленное тегированное структурное предложение для объекта = [ограничение на участие, " "], объект-источник, uniDirNullTag, [ограничение на участие, " "], объект-получатель ;

nullTag — однонаправленное тегированное структурное предложение для процесса = [ограничение на участие, " "], процесс-источник, uniDirNullTag, [ограничение на участие, " "], процесс- назначение ;

non nullTag — однонаправленное тегированное структурное предложение для объекта = [ограничение на участие, " "], объект-источник, " ", передний тег, " ", [ограничение на участие, " "], объект-назначение, [выражение для ограничения] ;

non nullTag — однонаправленное тегированное структурное предложение для процесса = [ограничение на участие, " "], процесс-источник, " ", передний тег, " ", [ограничение на участие, " "], процесс назначение ;

разветвленное тегированное структурное предложение = nullTag — разветвленное тегированное структурное предложение для объекта

| nullTag разветвленное тегированное структурное предложение для процесса

| nullTag разветвленное тегированное структурное предложение для объекта

| non nullTag разветвленное тегированное структурное предложение для процесса ;

nullTag — разветвленное тегированное структурное предложение для объекта = [ограничение на участие, " "], объект-источник; uniDirNullTag, объект-временное множество ;

nullTag — разветвленное тегированное структурное предложение для процесса = [ограничение на участие, " "], процесс-источник, uniDirNullTag, процесс-временное множество ;

non nullTag — разветвленное тегированное структурное предложение для объекта = [ограничение на участие, " "], объект-источник, " ", передний тег, " ", объект-временное множество ;

non nullTag — разветвленное тегированное структурное предложение для процесса = [ограничение участия, " "], процесс-источник, " ", передний тег, " ", процесс-временное множество ;

объект-временное множество = временной объект | ((временной объект, [{ "," временной объект}], " и ", (временной объект | " больше ")), [(", по заказу", критерий порядка) | (", в данной последовательности ")]) ;

процесс-временное множество = временной процесс | ((временной процесс, [{ ", ", временной процесс }], " и ", (временной процесс | " больше ")), [(", по заказу", критерии порядка) | (", в данной последовательности")]) ;

критерий порядка = имя ;

временной объект = [ограничение на участие, " "], объект с дополнительным состоянием ;

объект-источник = объект с дополнительным состоянием ;
объект-назначение = объект с дополнительным состоянием ;
временной процесс = [ограничение на участие, " "], идентификатор процесса ;
процесс-источник = идентификатор процесса ;
процесс-назначение = идентификатор процесса ;
uniDirNullTag = " относится к "
| " относится к "
| определяемые пользователем uniDirNullTag ;
передний тег = выражение тега;
определяемый пользователем uniDirNullTag = выражение тега ;
А.4.6.2.3 Двунаправленные тегированные структурные предложения (Bidirectional tagged structural sentences)
двунаправленное тегированное структурное предложение = асимметричное двунаправленное структурное предложение для объекта
| асимметричное двунаправленное структурное предложение для процесса
| симметричное двунаправленное структурное предложение для объекта
| симметричное двунаправленное структурное предложение для процесса ; (* см. 10.2.3 и 11.2 *)
асимметричное двунаправленное тегированное структурное предложение для объекта = ([ограничение на участие, " "], объект-источник, двунаправленный передний тег, [ограничение на участие, " "], объект-назначение, [выражение для ограничения])
| ([ограничение на участие, " "], объект-назначение, двунаправленный задний тег, [ограничение на участие, " "], объект-источник, [выражение для ограничения]) ;
асимметричное двухстороннее тегированное структурное предложение для процесса = ([ограничение на участии, " "], процесс-источник, двунаправленный передний тег, [ограничение на участие, " "], процесс-назначение)
| ([ограничение на участие, " "], процесс-назначение, двунаправленный задний тег, [ограничение на участие, " "], процесс-источник) ;
симметричное двунаправленное тегированное структурное предложение для объекта = ([ограничение на участие, " "], объект-источник, " и ", [ограничение на участие, " "], целевой объект-назначение, "является", biDirNullTag)
| ([ограничение на участие, " "], объект-источник, " и ", [участие на ограничение, " "], объект-назначение), " является ", симметричный тег ;
симметричное двунаправленное тегированное структурное предложение для процесса = ([ограничение на участие, " "], процесс-источник, " и ", [ограничение на участие, " "], процесс-назначение, "является", biDirNullTag)
| ([ограничение на участие, " "], процесс-источник, " и ", [ограничение на участие, " "], процесс назначения), " является ", симметричный тег ;
симметричный тег = выражение для тега ;
двунаправленный передний тег = выражение для тега ;
двунаправленный передний тег = выражение для тега ;
biDirNullTag = " связанный "
| определяемый пользователем biDirNullTag ;
определяемый пользователем biDirNullTag = выражение для тега ;
А.4.6.3 Фундаментальные OPL-структуры
А.4.6.3.1 Групповые предложения (Aggregation sentences)
групповое предложение = разветвленное групповое предложение для объекта
| разветвленное групповое предложение для процесса ; (* см. 10.3.2 *)
разветвленное групповое предложение для объекта = целый объект, " состоит из ", перечень частей объекта ;
разветвленное групповое предложение для процесса = весь процесс, "состоит из", перечень частей процесса ;
перечень частей объекта = частичный объект
| (частичный объект, [{ ", ", частичный объект }, " и ", (частичный объект | " по меньшей мере одна из других частей ")]) ;
перечень частей процесса = частичный процесс
| (частичный процесс, [{ ", ", частичный процесс }, " и ", (частичный процесс | " по крайней мере, одна из других частей ")]) ;
целый объект = идентификатор объекта ;
частичный объект = [ограничение на участие, " "], идентификатор объекта ;
целый процесс = идентификатор процесса;
частичный процесс = [ограничение на участие, " "], идентификатор процесса ;
А.4.6.3.2 Характеристические предложения (Characterization sentences)
характеристическое предложение = разветвленное характеристическое предложение для объекта
| разветвленное характеристическое предложение для процесса ; (* см. 10.3.3 *)
разветвленное характеристическое предложение для объекта = разветвленное характеристическое предложение для базового объекта

| разветвленное предложение для частичного объекта
| разветвленное характеристическое предложение для ASWellas-объекта
| разветвленное характеристическое предложение для частичного ASWellas-объекта ;
разветвленное характеристическое предложение для базового объекта = идентификатор объекта, " представляет ", (перечень атрибутов | операторов) ;
характеристическое предложение для частичного объекта = идентификатор объекта, " представляет ", ((перечень атрибутов , " и по меньшей мере еще один атрибут ")
| (перечень операторов, " и по меньшей мере еще один оператор ")) ;
разветвленное характеристическое предложение для ASWellas-объекта = идентификатор объекта, " представляет ", перечень атрибутов, и также перечень операторов ;
разветвленное характеристическое предложение для частичного ASWellas-объекта = идентификатор объекта, " представляет ", перечень атрибутов, и " по крайней мере еще один атрибут ", ", а также, " перечень операторов, и " по меньшей мере еще один оператор " ;
атрибут = идентификатор объекта ;
оператор = идентификатор процесса ;
перечень атрибутов = перечень объектов ;
перечень операторов = перечень процессов ;
разветвленное характеристическое предложение для процесса = разветвленное характеристическое предложение для базового процесса
| разветвленное характеристическое предложение для частичного процесса
| разветвленное характеристическое предложение для частичного ASWellas-процесса
| разветвленное характеристическое предложение для ASWellas-процесса ;
разветвленное характеристическое предложение для базового процесса = идентификатор процесса, " представляет ", (перечень операторов | атрибутов) ;
разветвленное характеристическое предложение для частичного процесса = идентификатор процесса, " представляет ", ((перечень операторов, и " по меньшей мере еще один оператор ")
| (перечень атрибутов и " по крайней мере еще один атрибут ")) ;
разветвленное характеристическое предложение для ASWellas-процесса = идентификатор процесса, " представляет ", перечень операторов, а также перечень атрибутов ;
разветвленное характеристическое предложение для частичного ASWellas-процесса = идентификатор процесса, " представляет ", перечень операторов, и " по меньшей мере еще один оператор ", а также перечень атрибутов, и " по меньшей мере еще один атрибут " ;
А.4.6.4 Представительские предложения (Exhibition sentences)
представительское предложение = представительское предложение для объекта
| представительское предложение для процесса ; (* см. 10.3.3.2.2 и 11.3 *)
представительское предложение для объекта = признак, "из", идентификатор объекта (диапазон пунктов | " является ", ((перечень атрибутов | операторов) | (перечень атрибутов, "а также", перечень операторов))) ;
представительское предложение для процесса = признак, "из", идентификатор процесса, " является ", ((перечень операторов | объектов)
| (перечень операторов, " а также ", перечень атрибутов)) ;
признак = атрибут | оператор ;
А.4.6.5 Специализированные предложения (Specialization sentences)
специализированное предложение = специализированное предложение для объекта
| специализированное предложение для процесса
| специализированное предложение для состояния ; (* см. 10.3.4 *)
специализированное предложение для объекта = специализированное предложение для базового объекта
| специализированное предложение для множественного объекта
| специализированное предложение для частичного объекта
| специализированное предложение для объекта типа Xor
| специализированное предложение для множественного унаследованного объекта ;
специализированное предложение для базового объекта = специальный объект, " является ", общий объект ;
специализированное предложение для множественного объекта = перечень специальных объектов, " является ", общий объект ;
специализированное предложение для частичного объекта = перечень специальных объектов, " имеются другие специализации ", общий объект ;
специализированное предложение для объекта типа Xor = специализированное предложение типа Xor для базового объекта
| специализированное предложение для объектов типа Xor, разделенных запятыми ;
специализированное предложение для базового объекта типа Xor = специальный объект, " может быть либо ", общий объект, либо "или" общий объект ;

специализированное предложение для объектов типа Xor, разделенных запятыми = специальный объект " может быть одним из ", общий объект, { ", ", общий объект}, " или ", общий объект ;

специализированное предложение для множественного унаследованного объекта = специальный объект, " есть ", перечень общих объектов ;

общий объект = идентификатор объекта ;

специальный объект = идентификатор объекта ;

перечень общих объектов = " а " , идентификатор объекта, [{ " а ", идентификатор объекта}], " и а ", идентификатор объекта ;

перечень специальных объектов = перечень объектов ;

специализированное предложение для процесса = специализированное предложение для базового процесса

| специализированное предложение для множественного процесса

| специализированное предложение для частичного процесса

| специализированное предложение для процессов типа Xor

| специализированное предложение для множественного унаследованного процесса ;

специализированное предложение для базового процесса = специальный процесс, " есть ", общий процесс ;

специализированное предложение для множественного процесса = перечень специальных процессов, " являются ", общий процесс ;

специализированное предложение для частичного процесса = перечень специальных процессов и " другими специализациями являются ", общий процесс ;

специализированное предложение для процессов типа Xor = специализированное предложение для базового процесса типа Xor

| специализированные предложения для процессов типа Xor, разделенные запятыми ;

специализированное предложение для базового процесса типа Xor = специальный процесс, " может быть ", общий процесс, " или ", общий процесс ;

специализированные предложения для процессов типа Xor, разделенные запятыми = специальный процесс, " может быть одним из ", общий процесс, { ", ", общий процесс}, " или ", общий процесс ;

специализированное предложение для множественного унаследованного процесса = специальный процесс, " есть ", перечень общих процессов ;

общий процесс = идентификатор процесса ;

специальный процесс = идентификатор процесса ;

перечень общих процессов = " а ", идентификатор процесса, [{ " а ", идентификатор процесса}] " и а ", идентификатор процесса ;

перечень специальных процессов = перечень процессов ;

специализированное предложение для состояния = специализированное предложение для базового состояния

| специализированное предложение для множественного состояния

| специализированное предложение для частичного состояния ;

специализированное предложение для базового состояния = состояние определенного объекта, " является ", состояние определенного объекта ;

специализированное предложение для множественного состояния = перечень определяющих состояние объектов, " являются ", состояние определенного объекта ;

специализированное предложение для частичного состояния = перечень определяющих состояние объектов, и "другие специализации ", состояние определенного объекта ;

определяющий состояние объект = идентификатор состояния, " ", идентификатор объекта ;

перечень определяющих состояние объектов = состояние определенного объекта

| определяющий состояние объект, [{ ", ", определяющий состояние объект }], " и ", с определяющий состояние объект ;

А.4.6.6 Конкретизирующие (инстанцирующие) предложения (Instantiation sentences)

конкретизирующее предложение = конкретизирующее предложение для объекта

| конкретизирующее предложение для процесса ; (* см. 10.3.5 *)

конкретизирующее предложение для объекта = конкретизирующее предложение для базового объекта

| конкретизирующее предложение для множественного объекта ;

конкретизирующее предложение для базового объекта = объект-экземпляр, " является экземпляром ", класс объектов ;

конкретизирующее предложение для множественного объекта = перечень объектов-экземпляров, " являются экземплярами ", класс объектов ;

конкретизирующее предложение для процесса = конкретизирующее предложение для базового процесса

| конкретизирующее предложение для множественного процесса ;

конкретизирующее предложение для базового объекта = процесс-экземпляр, " является экземпляром ", класс процесса ;

конкретизирующее предложение для множественного процесса = перечень экземпляров процесса, " являются экземпляром ", класс процесса;

объект-экземпляр = идентификатор объекта ;

процесс-экземпляр = идентификатор процесса ;

класс объектов = идентификатор объекта ;

класс процессов = идентификатор процесса ;

перечень объектов-экземпляров = перечень объектов ;

перечень процессов-экземпляров = перечень процессов ;

(* EndRegion: Структурные предложения *)

А.4.7 OPL-управление контекстом

А.4.7.1 Предложение для управления контекстом (Context management sentence)

(*Область: Предложения для управления контекстом — Эта область определяет все предложения, которые управляют преобразованием OPD-контекста *)

предложение для управления контекстом = разворачивающее предложение

| сворачивающее предложение

| детализирующее предложение

| обобщающее предложение ; (* см. 14.2.1 *)

(* на диаграмме объекта и процесса, разворачивание эквивалентно соответствующим структурным предложениям *)

А.4.7.2 Разворачивающие предложения (Unfolding sentences)

разворачивающее предложение = разворачивающее предложение для объекта

| разворачивающее предложение для процесса ;

разворачивающее предложение для объекта = разворачивающее предложение для недоопределенного объекта

| разворачивающее предложение для полного объекта

| разворачивающее предложение для целого объекта

| разворачивающее предложение для класса объектов

| разворачивающее предложение для объекта-представителя ;

разворачивающее предложение для недостаточно определенного объекта = идентификатор объекта, "разворачивается в ", перечень атрибутов, [" а также ", перечень операторов] ;

разворачивающее предложение для полного объекта = полный объект, " от ", родительская OPD-диаграмма, " часть, разворачиваемая в ", дочерняя OPD-диаграмма, " в ", перечень частей объекта ;

разворачивающее предложение для целого объекта = целый объект, " от ", родительская OPD-диаграмма, " специализация, разворачиваемая в ", дочерняя OPD-диаграмма, " в ", перечень специальных объектов ;

разворачивающее предложение для класса объектов = класс объектов, " от ", родительская OPD-диаграмма, " экземпляр, разворачиваемый в ", дочерняя OPD-диаграмма, " в ", перечень экземпляров объектов ;

разворачивающее предложение для объекта-представителя = идентификатор объекта, " от ", родительская OPD-диаграмма, " признак, разворачиваемый в ", дочерняя OPD-диаграмма, " в ", перечень атрибутов, [" а также ", перечень операторов] ;

разворачивающее предложение для процесса = разворачивающее предложение для недостаточно определенного процесса

| разворачивающее предложение для полного процесса

| разворачивающее предложение для целого процесса

| разворачивающее предложение для класса процессов

| разворачивающее предложение для процесса-представителя ;

разворачивающее предложение для недоопределенного процесса = идентификатор процесса, " разворачиваемого в ", перечень операторов, [" а также ", перечень атрибутов] ;

разворачивающее предложение для полного процесса = полный процесс, " от ", родительская OPD-диаграмма, " часть, разворачиваемая в ", дочерняя OPD-диаграмма, " в ", перечень частей процесса ;

разворачивающее предложение для целого процесса = целый процесс, " от ", родительская OPD-диаграмма, " специализация, разворачиваемая ", дочерняя OPD-диаграмма, " в ", перечень специальных процессов ;

разворачивающее предложение для класса процессов = класс процессов, " от ", родительская OPD-диаграмма, " экземпляр, разворачиваемый в ", дочерняя OPD-диаграмма, " в ", перечень экземпляров процесса ;

разворачивающее предложение для процесса-представителя = идентификатор процесса, " от ", родительская OPD-диаграмма , " признак, разворачиваемый в ", дочерняя OPD-диаграмма, " в ", перечень операторов, [" а также ", перечень атрибутов] ;

А.4.7.3 Сворачивающие предложения (Folding sentences)

сворачивающее предложение = сворачивающее предложение для объекта

| сворачивающее предложение для процесса ;

(*сворачивающее предложение относится только к OPD-объекту или процессу, для которого разворачивание приводит к дочерней OPD-диаграмме и является OPL-эквивалентом графического обозначения контура, выделяемого жирным шрифтом *)

сворачивающее предложение для объекта = идентификатор объекта, "является сворачиванием ", дочерняя OPD-диаграмма ;

сворачивающее предложение для процесса = идентификатор процесса, " является сворачиванием ", дочерняя OPD-диаграмма ;

А.4.7.4 Детализирующее предложение (In zoom sentence)

детализирующее предложение = детализирующее предложение для процесса

| детализирующее предложение для объекта ;

детализирующее предложение для процесса = детализирующее предложение для существующей диаграммы процесса

| детализирующее предложение для новой диаграммы процесса ;

детализирующее предложение для существующей диаграммы процесса = (идентификатор процесса, " разбиение на ", перечень процессов, " в этой последовательности ", [", а также", перечень детализированных объектов])

| (идентификатор процесса, " разбиение на параллельные ", перечень процессов, [", а также", перечень детализированных объектов])

| (идентификатор процесса, " разбиение на ", перечень процессов, " и параллельно ", перечень процессов, " в этой последовательности ", [" а также ", перечень детализированных объектов]) ;

детализирующее предложение для новой диаграммы процесса = (идентификатор процесса, " от ", родительская OPD-диаграмма, " детализирует ", дочерняя OPD-диаграмма, " в ", перечень процессов, " в этой последовательности ", [", а также", перечень детализированных объектов])

| (идентификатор процесса, " от ", родительская OPD-диаграмма, " детализирует ", дочерняя OPD-диаграмма " в параллель ", перечень процессов, [", а также ", перечень детализированных объектов])

| (идентификатор процесса, " от ", родительская OPD-диаграмма, " детализирует ", дочерняя OPD-диаграмма, " в ", перечень процессов, " и параллельно ", перечень процессов ", в этой последовательности ", [", а также ", перечень детализированных объектов]);

детализирующее предложение для объекта = детализирующее предложение для существующей диаграммы процесса

| детализирующее предложение для новой диаграммы процесса ;

детализирующее предложение для существующей диаграммы процесса = (идентификатор объекта, " разбиение на ", перечень объектов, " в этой последовательности ", [", а также ", перечень детализированных процессов]) ;

детализирующее предложение для новой диаграммы процесса = (идентификатор объекта " от ", родительская OPD-диаграмма, " разбиение на ", дочерняя OPD-диаграмма, " в ", перечень объектов, " в этой последовательности ", [", а также ", перечень детализированных процессов]) ;

перечень детализированных объектов = идентификатор объекта, [{ ", ", идентификатор объекта}, " и ", идентификатор объекта ", в этой последовательности "] ;

перечень детализированных процессов = идентификатор процесса, [{ ", ", идентификатор процесса}, " и ", идентификатор процесса, ", в этой последовательности "] ;

А.4.7.5 Обобщающее предложение (Out zooming sentence)

обобщающее предложение = обобщающее предложение для процесса

| обобщающее предложение для объекта ;

(* обобщающее предложение актуально только для OPD-процесса или объекта, для которого детализация формирует дочернюю OPD-диаграмму и является OPL-эквивалентом графического обозначения жирно выделенного контура *)

обобщающее предложение для процесса = идентификатор процесса, " является обобщением ", дочерняя OPD-диаграмма ;

обобщающее предложение для объекта = идентификатор объекта, " является обобщением ", дочерняя OPD-диаграмма ;

(* EndRegion: Предложения для управления контекстом *)

(* EndRegion: OPL-документ *)

(* EndRegion: OPL EBNF *)

Приложение В
(справочное)

Руководство по применению OPM-методологии

В.1 Общие сведения

В связи с быстрым развитием сложных и усложненных систем необходимость в интуитивном, не формальном способе документирования стандартов, проектирования новых систем или получения знаний о существующих системах становится все более очевидной. Эта потребность, в свою очередь, требует создания надежной инфраструктуры для записи, хранения, формирования и представления накопленной информации и творческих идей, которые основываются на этих знаниях.

Концептуальное моделирование относится к методике представления знаний, связанных с системой. Результатом этого моделирования является создание концептуальной модели, которое обычно предшествует математическому и физическому моделированию. Концептуальное моделирование является основным видом деятельности, которое необходимо не только для понимания инженерных систем, их разработки и управления, но также и для создания стандартов, настолько полных и согласованных, насколько это возможно. Подобное моделирование является существенно важным и приводящим к моделе-ориентированной системной инженерии (MBSE).

Понимание физических, биологических, искусственных и социальных систем и разработка стандартов, связанных с ними, требует хорошо обоснованной, формальной, но интуитивно понятной методологии и языка, которые будут обеспечивать возможность моделирования всех проблем в согласованной и простой форме. Та же парадигма моделирования (ядро методологии) должна служить как для разработки новых систем, так и для изучения и совершенствования существующих систем. Эту же парадигму необходимо применять к искусственным и естественным системам, и должным образом представлять физические и информационные сущности в моделируемой области. OPM-методология дает соответствующие средства для реализации этих целей.

В.2 Сущности, принципиально значимые для OPM-методологии

Основные процессы системного уровня могут быть столь же важными (или даже более важными), чем объекты в модели системы. В частности, OPM-методология указывает, что процесс верхнего уровня в OPM-модели системы является функцией системы и создающим ценность процессом, который воплощает в себе назначение и область применения системы. Таким образом, процесс должен быть пригоден для моделирования и не должен зависеть от какого-то ни было конкретного набора объектов, связанных с его возникновением.

Относительная значимость сущности **Т** в OPM-модели системы, как правило, пропорциональна OPD-диаграмме наиболее высокого уровня в OPD-иерархии, где проявляется сущность **Т**.

В.3 Что нового должна содержать OPD-диаграмма?

Качественный набор OPD-диаграмм должен быть удобочитаемым и хорошо понимаемым. Нижеприведенные эмпирические правила будут полезны при принятии решений относительно создания новой OPD-диаграммы и способов поддержания других OPD-диаграмм в удобочитаемом и простом виде (насколько это возможно):

- OPD-диаграмма не должна занимать более одной страницы или одного экрана монитора среднего размера;
- OPD-диаграмма не должна содержать более 20—25 сущностей;
- сущности не должны перекрывать друг друга, т. е. они должны либо полностью содержаться в сущностях более высокого уровня, например в случае масштабирования, или не иметь областей пересечения;
- OPD-диаграмма не должна содержать слишком много связей — примерно столько же связей, сколько и сущностей;
- связь не должна пересекать область, занимаемую сущностью; а также
- число пересекающихся связей должно быть сведено к минимуму.

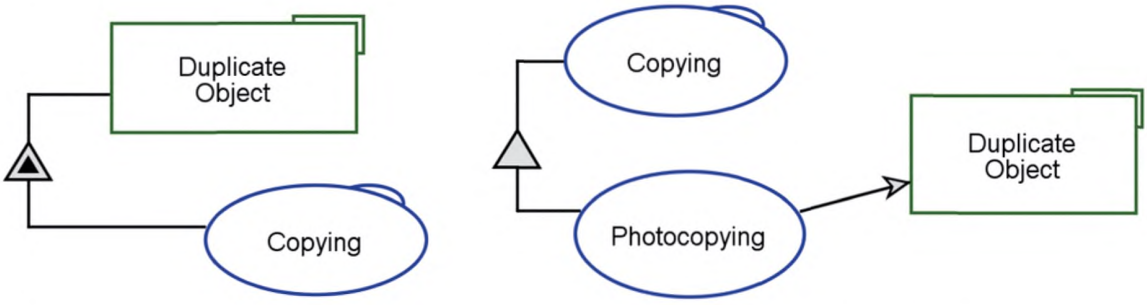
В.4 OPM-принцип представления элементов

Элемент OPM-модели, появляющийся на одной OPD-диаграмме, может появляться и на любой другой OPD-диаграмме. Этот принцип дает возможность представления любого элемента модели (сущности или связи) любое количество раз на таком числе OPD-диаграмм, какое проектировщик посчитает необходимым. Поскольку связь не может существовать без сущностей, которые она связывает, для связи, которая должна появляться на OPD-диаграмме, необходимо наличие на этой диаграмме как минимум двух связываемых сущностей.

Хотя проектировщик модели может вводить на любую OPD-диаграмму любое количество сущностей, по соображениям сохранения ясности и во избежание затруднений часто бывает крайне желательно вводить на OPD-диаграмму только те элементы, которые необходимы для понимания определенного аспекта или представления системы.

B.5 Соглашение относительно многократного копирования сущности

Во избежание создания длинных и извилистых связей, которые будут пересекать OPD-диаграмму с одного края до другого и тем самым загромождать эту диаграмму, она может содержать несколько копий одной и той же сущности. Это соглашение относительно многократного копирования сущности дополняет OPM-принцип представления элемента. Подобно тому, как элемент OPM-модели появляется на одной OPD-диаграмме, он может появляться и на любой другой OPD-диаграмме. Соответственно, во избежание затруднений, связанных с длинными, пересекающимися связями на OPD-диаграмме, любая сущность может появляться в другом месте на той же диаграмме, используя для этого более короткую связь. Для облегчения выявления повторов проектировщик модели может заменить символ сущности на соответствующий дублирующий символ этой же сущности в виде небольшого символа объекта или процесса, слегка выступающего за символ основной сущности (см. рисунок B.1). Тем не менее проектировщик модели должен с осторожностью использовать эту альтернативу, поскольку это требует от читателя определять и иметь в виду более длинные связи, которые отсутствуют в явном виде на существующей OPD-диаграмме.



Dublicate Object — дублированный объект; Coping — копирование; Photocoping — фотокопирование

Рисунок B.1 — Дублированный объект и символы дублирования процесса

B.6 Рекомендации по именованию элементов

B.6.1 Важность выбора имен элементов

Выбор подходящих имен для маркировки элементов в OPM-модели, т. е. объектов, процессов и связей, очень важен, поскольку эта маркировка может влиять на легкость понимания модели со стороны целевой аудитории, ясности логического потока и принятия здравых решений на основании соответствующих OPL-предложений.

B.6.2 Именование объектов

Именование объекта должно быть уникальным, с возможностью преобразования наименований множественного числа в особую форму. Рекомендуемый способ преобразования объекта с несколькими элементами — введение слова «набор» (обычно для неодушевленных объектов) или слова «группа» (как правило, относящегося к человеку) после формы единственного числа.

Пример 1 — Наименование «Ингредиенты» (скажем, торта) становится наименованием «Набор ингредиентов», а наименование «Клиенты» становится наименованием «Группа клиентов».

Поскольку наименования объектов должны быть уникальными в рамках данной модели системы, проектировщик модели может использовать имя сущности *refineable* в качестве префикса для своих уточненных наименований или может использовать имя сущности *refineable*, с суфиксом «of» после уточненного наименования. Любой из этих случаев обеспечивает получение контекстных различий для уточнения сходной семантики.

Именами объектов могут быть фразы, состоящие из нескольких слов, например, «яблочный пирог» или «автомобильная авария».

Пример 2 — Если проектировщик модели желает использовать слово Size (Размер) в качестве атрибута для объектов Clock Set (Установка часов) и Watch Set (Установка наблюдения), а затем различать эти два атрибута, то первый объект может именоваться как Clock Set Size, а второй — как Watch Set Size или же как Size of Clock Set и Size of Watch Set соответственно.

Примечание 1 — Реализация OPM-методологии может уведомлять разработчика модели при попытке введения объекта как объекта типа *refinee* в более чем одном контексте, так что он смог бы определять целесообразность подобного включения.

Примечание 2 — Реализация OPM-методологии может устанавливать синтаксис по умолчанию для разрешения именования объектов типа *refinee*.

В.6.3 Именование процессов

Наименованием процесса является фраза, последнее слово в которой должно быть герундием от формы глагола, т. е. глагола с суффиксом «ing». При наличии нескольких вариантов, например наименования процесса «Создание» — Construction или Constructing, предпочтение должно быть отдано последнему варианту.

Существуют следующие варианты наименований процессов:

- «глагольный вариант», который является просто герундийной формой глагола, а именно глагол + суффикс ing, например, **Making** (Создание) или **Responding** (Реагирование);

- «неглагольный вариант», который представляет собой ассоциации существительного (OPM-объект) с герундием, а именно существительное + глагол + ing, например, **Cake Making** (Приготовление торта) или **Crash Responding** (Реагирование на аварию);

- вариант типа «прилагательное-глагол», который представляет собой ассоциации прилагательного с герундийной формой глагола, а именно прилагательное + глагол + ing, например, **Cake Making** (Приготовление торта) или **Crash Responding** (Реагирование на аварию); а также

- версия «прилагательное-существительное-глагол», которая представляет собой ассоциации прилагательного с существительным и герундием, а именно прилагательное + существительное + глагол + ing, например, **Quick Cake Making** (Быстрое приготовление торта) или **Automatic Crash Responding** (Автоматическое реагирование на аварию).

В последнем случае прилагательное ограничивает процесс (герундий, который является существительным), однако прилагательное может также ограничивать объект (существительное), например, **Sweet Cake Making** ((Приготовление сладкого торта) или **Fatal Crash Responding** (Реагирование на аварию со смертельными исходами).

Наименование функции, а также наименования всех OPM-процессов должны содержать не более четырех слов с большой буквы, оканчивающихся с герундийной формой глагола, например, **Large City Population Securing** (Защита населения большого города).

Поскольку наименования процессов должны быть уникальными, разработчик модели может использовать имя сущности *refineable* в качестве суффикса «of» после уточненного наименования. Схема именования позволяет контекстуально различать (при обращении) объекты типа *refinees* со схожей семантикой.

В.6.4 Именование состояний

Наименования состояний должны отражать соответствующие ситуации, в которых может находиться «владеющий» ими объект в любой заданный момент времени. Предпочтительные наименования состояний являются пассивными, а не герундийными формами «владеющего» объекта.

Пример — Если объект Product окрашивается и затем проверяется, его состояниями должны быть painted (окрашенное) и inspected (проверенное), а не окраска и проверка. Painting (Окраска) — это процесс, который изменяет объект Product (Изделие) из состояния unpainted (неокрашенное) на состояние painted (окрашенное), а процесс Inspecting (Проверка) изменяет объект Product (Изделие) из состояния painted (окрашенное) на состояние inspected (проверенное). Хотя процесс Painting для объекта Product возникает, состояние unpainted этого объекта будет сохраняться до тех пор, пока будет протекать процесс Painting, и будет оставаться в промежуточном состоянии и не будет входить в состояние painted до завершения процесса Painting.

В.6.5 Соглашение относительно заглавных букв

В OPM-методологии первые буквы каждого слова в наименовании сущностей (объектов или процессов) должны быть заглавными, тогда как наименования состояний объектов или связей должны состоять из строчных букв. Данное соглашение облегчает формирование OPL-предложений, которые являются более удобочитаемыми.

Приложение С
(справочное)

OPM-моделирование с помощью OPD-диаграмм

С.1 OPM-модели в OPM-методологии

OPD-диаграмма, приведенная на рисунке С.1, отражает различные аспекты OPM-методологии в виде OPM-моделей. В разделе С.4 рассмотрены специфические элементы этих моделей, в разделе С.5 представлена модель, связанная с обработкой связей в процессе разворачивания и детализации, а в разделе С.6 — модель для оценки инициализации, выполнения и завершения процессов.

Данная группа разделов позволяет выражать OPM-методологию в виде набора OPD-диаграмм (вместе с соответствующим OPL-синтаксисом), для представления которого разработчик модели должен выбирать ограничения на контент модели с целью оптимального использования OPM-методологии, т. е. выбирать минимальное число сложных связей и не предпринимать попыток объединения OPD-диаграмм в обобщенную OPM-модель. Тем не менее некоторые улучшенные OPL-выражения, которые ограничивают избыточность текста и облегчают восприятие различаются, в противном случае могут возникнуть различные, но связанные с ними модельные факты.

С.2 Структура OPM-модели

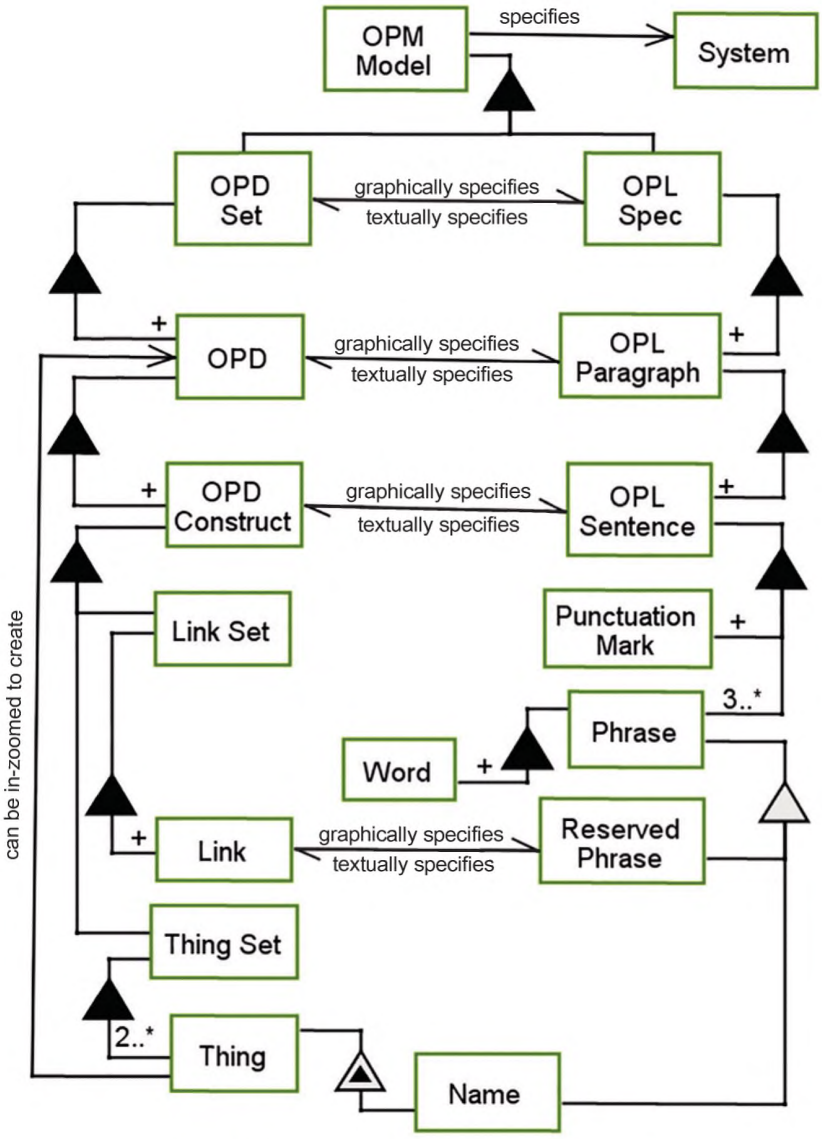


Рисунок С.1, лист 1 — Структура OPM-модели

Объект **OPM Model** (OPM-модель) связан с объектом **System** (Система) и имеет тег **specifies** (определяет).

Объект **OPM Model** состоит из объектов **OPD Set** (OPD-набор) и **OPL Spec** (OPL-спецификация).

Объект **OPL Spec** (OPL-спецификация) состоит по меньшей мере из одного объекта **OPL Paragraph** (OPL-раздел).

Объект **OPD Set** (OPD-набор) состоит по меньшей мере из одного объекта **OPD** (OPD-диаграмма).

Объект **OPD Set** (OPD-набор) связан с объектом **OPL Spec** (OPL-спецификация) и имеет тег **graphically specifies** (графически определяет).

Объект **OPL Spec** (OPL-спецификация) связан с объектом **OPD Set** (OPD-набор) и имеет тег **textually specifies** (текстуально определяет).

Объект **OPD** (OPD-диаграмма) содержит по меньшей мере один объект **OPD Construct** (OPD-структура).

Объект **OPL Paragraph** (OPL-раздел) содержит по меньшей мере один объект **OPL Sentence** (OPL-предложение).

Объект **OPD** (OPD-диаграмма) связан с объектом **OPL Paragraph** (OPL-раздел) и имеет тег **graphically specifies** (графически определяет).

Объект **OPL Paragraph** (OPL-раздел) связан с объектом **OPD** (OPD-диаграмма) и имеет тег **textually specifies** (текстуально определяет).

Объект **OPD Construct** (OPD-структура) связан с объектом **OPL Sentence** (OPL-предложение) и имеет тег **graphically specifies** (графически определяет).

Объект **OPL Sentence** (OPL-предложение) связан с объектом **OPD Construct** (OPD-структура) и имеет тег **textually specifies** (текстуально определяет).

Объект **OPD Construct** (OPD-структура) содержит объекты **Thing Set** (Набор сущностей) и **Link Set** (Набор связей).

Объект **Thing Set** (Набор сущностей) содержит не менее двух объектов **Things** (Сущности).

Объект **Link Set** (Набор связей) содержит по меньшей мере один объект **Link** (Связь).

Объект **Thing** (Сущность) представляет объект **Name** (Имя).

Объект **OPL Sentence** (OPL-предложение) содержит не менее трех объектов **Phrases** (Фразы) и по меньшей мере один объект **Punctuation Mark** (Знак препинания).

Объект **Phrase** (Фразы) содержит по меньшей мере один объект **Word** (Слово).

Объекты **Reserved Phrase** (Зарезервированная фраза) и **Name** (Имя) для объекта **Thing** (Сущность) являются объектами **Phrases** (Фразы).

Объект **Link (Связь)** связан с объектом **Reserved Phrase** (Зарезервированная фраза) и имеет тег **graphically specifies** (графически определяет).

Объект **Reserved Phrase** (Зарезервированная фраза) связан с объектом **Link** (Связь) и имеет тег **textually specifies** (текстуально определяет).

Объект **Thing** (Сущность) связан с объектом **OPD** (OPD-диаграмма) и имеет тег **can be in-zoomed to create** (может быть детализирован для создания объекта **OPD**).

Рисунок С.1, лист 2

На рисунке С.1 представлена модель структуры объекта **OPM model** (OPM-модель), на которой изображены концептуальные аспекты OPM-методологии в виде иерархических параллельных структур из графических и текстуальных OPM-форм и их взаимозависимостей, которые необходимы для создания эквивалентных выражений модели. Объект **OPD Construct** (OPD-структура) — это графическое выражение соответствующего текстуального объекта **OPL Sentence** (OPL-предложение), выражающего ту же самую модель. OPD-диаграмма и соответствующий ей объект **OPL Paragraph** (OPL-раздел) — это совокупность проявлений модели (фактов модели), которую разработчик модели может вводить в один и тот же ее контекст.

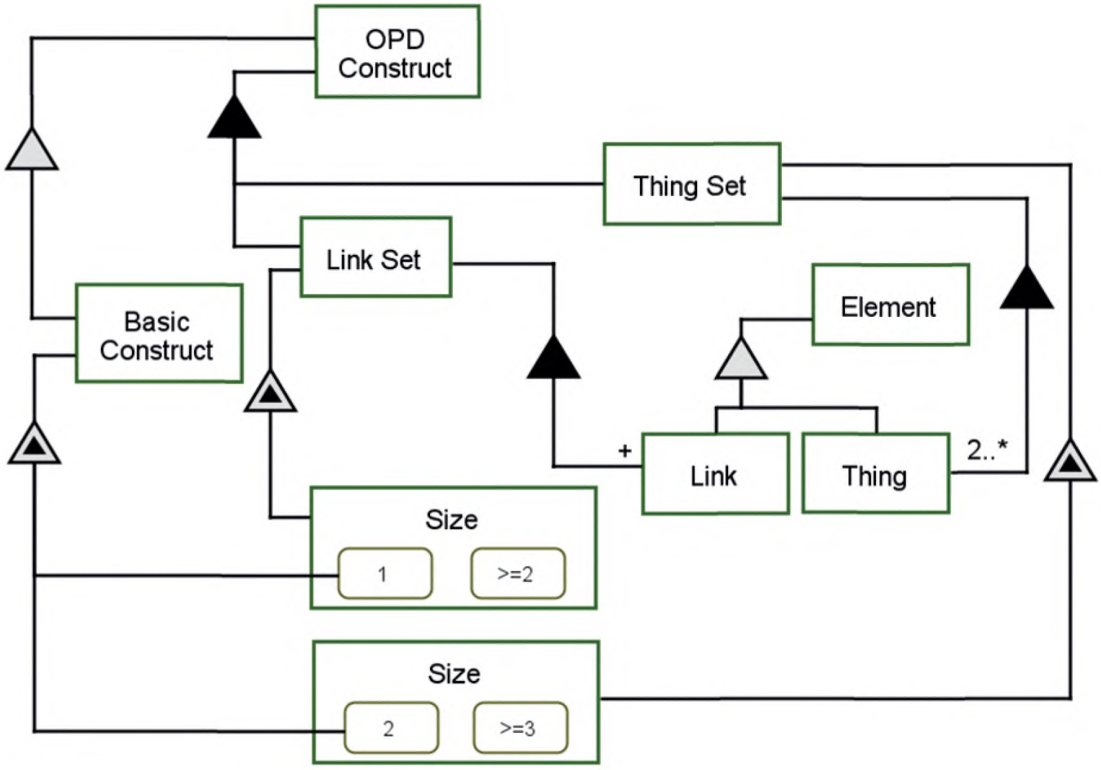
С.3 Модель OPD-структуры

На рисунке С.2 подробно представлена концепция объекта **OPD Construct** (OPD-структура), которая призвана для различения объекта **Basic Construct** (Базовая структура) от любых других возможных объектов **OPD Construct**. Объект **Basic Construct** является специализацией объекта **OPD Construct**, который содержит лишь два объекта **Things** (Сущности), соединенных лишь с одним объектом **Link** (Связь). Небазовые структуры включают в себя, помимо прочего, объекты с веерными связями или не менее двух объектов типа *refinees*.

Пример 1 — На рисунке С.1 два объекта — OPM Model (OPM-модель) и OPD Set (OPD-набор) вместе со связью типа «агрегация — является частью» между ними образуют базовую структуру. OPL-предложение, соответствующее данной базовой структуре, может выглядеть следующим образом: Объект OPM Model содержит объект OPD Set.

Пример 2 — На рисунке С.1 три объекта — OPM Model (OPM-модель), OPD Set (OPD-набор) и OPL Spec (OPL-спецификация) вместе со связью типа «агрегация — является частью» между объектами OPM Model, OPD Set и OPL Spec образуют сложную конструкцию. OPL-предложение, соответствующее данной базовой конструкции, выглядит следующим образом: Объект OPM Model содержит объекты OPD Set и OPL Spec.

Примечание — Определяющая состояние объекта связь является неявно определенной между объектом и каждым из его состояний. Графически это выражение связи возникает путем помещения состояния в треугольничек объекта, эффективно связывая состояние с объектом, поэтому объект не менее чем с двумя состояниями является объектом **OPD Construct** (OPD-структура), а объект с одним состоянием является объектом **Basic Construct** (Базовая структура). Объект, не имеющий (не меняющий) состояния, вообще не является структурой, поскольку даже не является неявной связью.

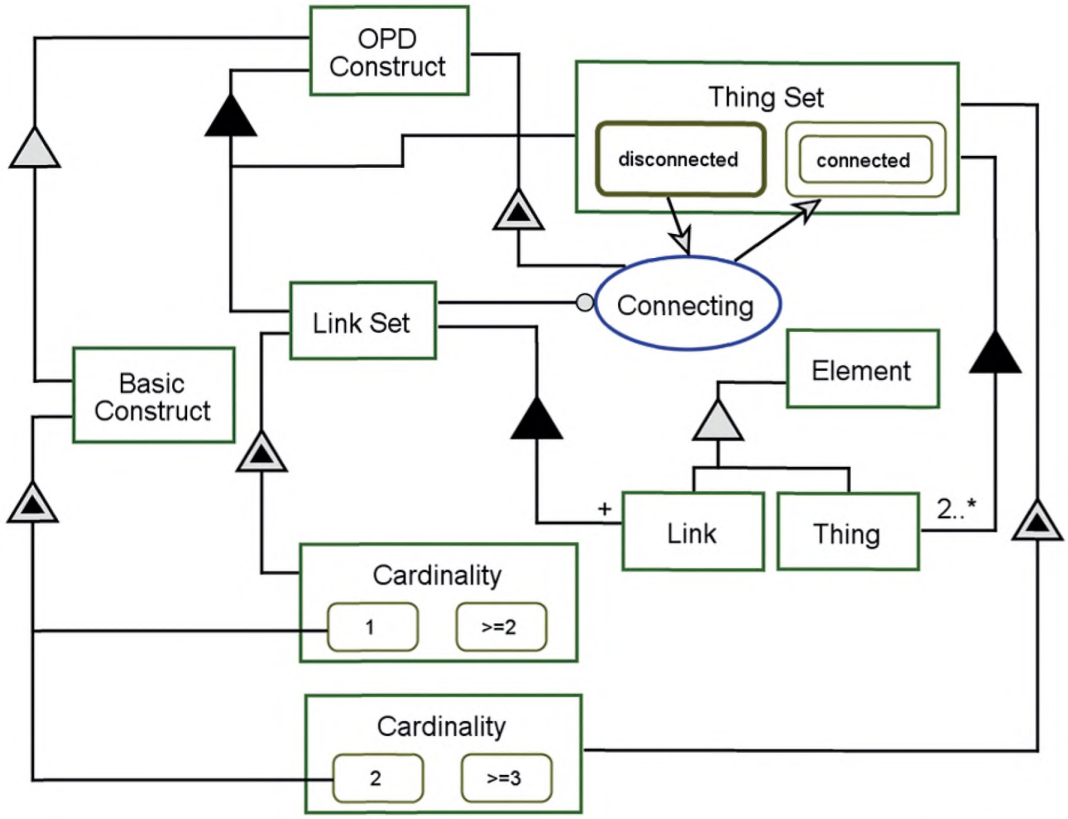


Объект **OPD Construct** (OPD-конструкция) состоит из объектов **Thing Set** (Набор сущностей) и **Link Set** (Набор связей).
Объекты **Thing** (Сущность) и **Link** (Связь) являются объектом **Elements** (Элементы).
Объект **Thing Set** (Набор сущностей) содержит не менее двух объектов **Things** (Сущности).
Объект **Link Set** (Набор связей) состоит из по меньшей мере одного объекта **Link** (Связь).
Объект **Thing Set** (Набор сущностей) представляет объект **Size** (Размер) объекта **Thing Set**.
Объект **Link Set** (Набор связей) представляет объект **Size** (Размер) объекта **Link Set**.
Объект **Cardinality** (Мощность связи) для объекта **Thing Set** (Набор сущностей) может быть в состоянии **2** или **>=3**.
Объект **Size** (Размер) для объекта **Link Set** (Набор связей) может быть в состоянии **1** или **>=2**.
Объект **Basic Construct** (Базовая структура) является объектом **OPD Construct** (OPD-конструкция).
Объект **Basic Construct** (Базовая структура) представляет состояние **1** объекта **Size** (Размер) для объекта **Link Set** (Набор связей).
Объект **Basic Construct** (Базовая структура) представляет состояние **2** объекта **Cardinality** (Мощность связи) для объекта **Thing Set** (Набор сущностей).

Рисунок С.2 — Модель OPD-структуры и базовой структуры

В некоторых ситуациях синтаксис этих двух структур могут быть легко объединен в сложное OPL-предложение, которое снизит избыточность текста, как это показано в следующем варианте объекта **OPD Construct** (OPD-структура).

Разработчик модели может ввести процесс в модель рисунка С.2 для указания того, что объект **OPD Construct** (OPD-структура) представляет процесс **Connecting** (Соединение), см. рисунок С.3. Путем введения состояний **disconnected** (отсоединенный) и **connected** (соединенный) объекта **Thing Set** (Набор сущностей) назначение данной модели включает действие по преобразованию состояния **disconnected** объекта **Thing Set** в состояние **connected** объекта **Thing Set**, используя для этого объект **Link Set** (Набор связей) в качестве инструментального средства соединения.

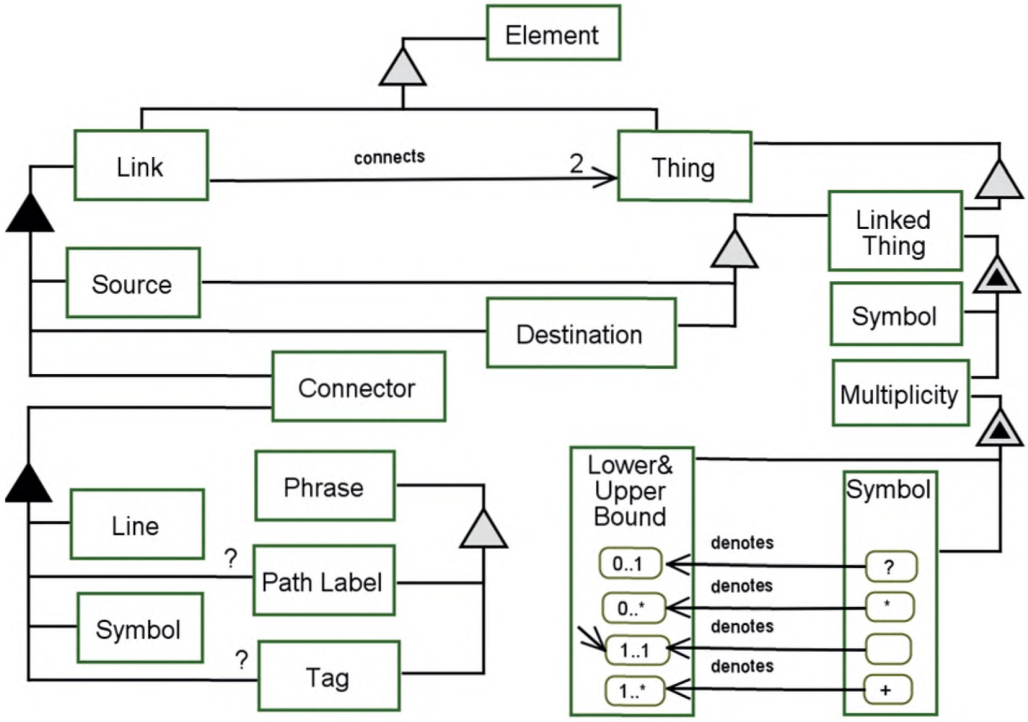


Объект **OPD Construct** (OPD-структура) состоит из объектов **Thing Set** (Набор сущностей) и **Link Set** (Набор связей).
Объект **OPD Construct** (OPD-структура) представляет процесс **Connecting** (Соединение).
Объект **Link Set** (Набор связей) состоит по меньшей мере из одного объекта **Link** (Связь).
Объект **Link Set** (Набор связей) представляет объект **Cardinality** (Мощность связи).
Объект **Cardinality** (Мощность связи) для объекта **Link Set** (Набор связей) может быть в состоянии **1** или **>=2**.
Объект **Thing Set** (Набор сущностей) представляет объект **Cardinality** (Мощность связи).
Объект **Thing Set** (Набор сущностей) состоит из не менее двух объектов **Cardinality** (Мощность связи).
Объект **Cardinality** (Мощность связи) для объекта **Thing Set** (Набор сущностей) может быть в состоянии **2** или **>=3**.
Объекты **Link** (Связь) и **Thing** (Сущность) являются объектами **Elements** (Элементы).
Объект **Connecting** требует объекта **Link Set** (Набор связей).
Объект **Connecting** изменяет состояние объекта **Thing Set** (Набор сущностей) с состояния **disconnected** (отсоединенный) на состояние **connected** (соединенный).
Состояние **disconnected** (отсоединенный) объекта **Thing Set** (Набор сущностей) является начальным.
Состояние **connected** (соединенный) объекта **Thing Set** (Набор сущностей) является конечным.
Объект **Basic Construct** (Базовая структура) является объектом **OPD Construct**.
Объект **Basic Construct** (Базовая структура) представляет состояние **1** объекта **Size** (Размер) для объекта **Link Set** (Набор связей) и состояние **2** объекта **Cardinality** (Мощность связи) для объекта **Thing Set** (Набор сущностей).

Рисунок С.3 — OPD-структура и базовая структура

С.4 Модели OPM-элемента

Модель, изображенная на рисунке С.4, применима только для базовых структур, поскольку объект **Link** (Связь) соединяется с 2 объектами **Things** (Сущности) с тегом **connects** (соединяется) и не превышает 2.



Объекты **Thing** (Сущность) и **Link** (Связь) являются объектом **Elements** (Элементы).

Объект **Link** (Связь) соединяется с 2 объектами **Things** (Сущности).

Объект **Link** (Связь) содержит объекты **Source** (Источник), **Destination** (Получатель) и **Connector** (Соединитель).

Объект **Connector** содержит объекты **Line** (Строка), **Symbol** (Символ), а также дополнительные объекты **Tag** (Тег) и **Path Label** (Метка пути).

Объекты **Tag** (Тег) и **Path Label** (Метка пути) являются объектами **Phrases** (Фразы).

Объекты **Source** (Источник) и **Destination** (Получатель) являются объектами **Linked Things** (Связанные сущности).

Объект **Linked Thing** (Связанная сущность) является объектом **Thing** (Сущность).

Объект **Linked Thing** (Связанная сущность) представляет объекты **Symbol** (Символ) и **Multiplicity** (Кратность).

Объект **Multiplicity** (Кратность) представляет объекты **Symbol** (Символ) и **Lower & Upper Bound** (Нижний & Верхний предел).

Объект **Lower & Upper Bound** (Нижний & Верхний предел) может быть в состоянии **0..1, 0..*, 1..1,** или **1..***.

Объект **Lower & Upper Bound** (Нижний & Верхний предел) по умолчанию находится в состоянии **1..1**.

Объект **Symbol** (Символ) для объекта **Multiplicity** (Кратность) может быть в состоянии **?, *, NONE,** или **1**.

Состояние **?** объекта **Symbol** (Символ) для объекта **Multiplicity** (Кратность) связано с состоянием **0..1** объекта **Lower & Upper Bound** (Нижний & Верхний предел) и имеет тег **denotes** (означает).

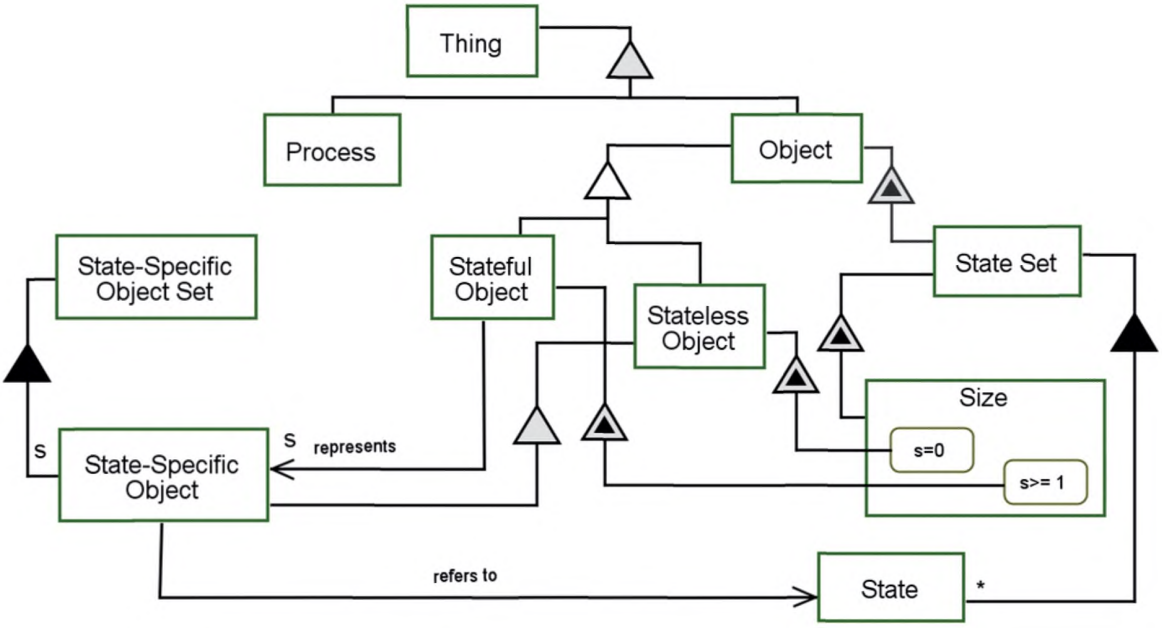
Состояние * объекта **Symbol** (Символ) для объекта **Multiplicity** (Кратность) связано с состоянием **0..*** объекта **Lower & Upper Bound** (Нижний & Верхний предел).

Состояние **NONE** объекта **Symbol** (Символ) для объекта **Multiplicity** (Кратность) связано с состоянием **1..1** объекта **Lower & Upper Bound** (Нижний & Верхний предел).

Состояние **+** объекта **Symbol** (Символ) для объекта **Multiplicity** (Кратность) связано с состоянием **1..*** объекта **Lower & Upper Bound** (Нижний & Верхний предел).

Рисунок С.4 — Диаграмма OPM-модели OPM-элемента

Модель, изображенная на рисунке С.5, применима для OPM-объекта **Thing** (Сущность), которая показывает его специализацию в объекты **Object** (Объект) и **Process** (Процесс). Совокупность объектов **States** (Состояния) характеризует объект **Object**, который может быть пустым в объекте **Stateless Object** (Объект без внутреннего состояния) или непустым — в случае объекта **Stateful Object** (Объект с внутренним состоянием). Объект **Stateful Object** с **s**-объектами **States** (Состояния) может приводить к возникновению набора объектов без внутреннего состояния — объектов **State-Specific Objects** (Зависящие от состояния объекты) — по одному на каждый объект **State** (Состояние). Конкретный объект **State-Specific Object** относится к объекту в специфическом состоянии. Моделирование концепции объекта **State-Specific Object** как объектов **Object** и **State** позволяет упрощать концептуальную модель в отношении объекта и любого его состояния путем простого определения объекта **Object**.



Объекты **Process** (Процесс) и **Object** (Объект) являются объектами **Things** (Сущности).

Объект **Object** (Объект) представляет объект **State Set** (Набор состояний).

Объект **State Set** (Набор состояний) представляет объект **Size** (Размер).

Объект **Cardinality** (Мощность связи) объекта **State Set** (Набор состояний) может быть в состоянии **s=0** или **s>= 1**.

Объект **State Set** (Набор состояний) состоит из дополнительных объектов **States** (Состояния).

Объект **Current State** (Текущее состояние) является объектом **State** (Состояние).

Объекты **Stateless Object** (Объект без внутреннего состояния) и **Stateful Object** (Объект с внутренним состоянием) являются объектами **Objects** (Объекты).

Объект **Stateless Object** (Объект без внутреннего состояния) представляет состояние **s= 0** объекта **Size** (Размер) для объекта **State Set** (Набор состояний).

Объект **Stateful Object** (Объект с внутренним состоянием) представляет состояние **s>= 1** объекта **Size** (Размер) для объекта **State Set** (Набор состояний).

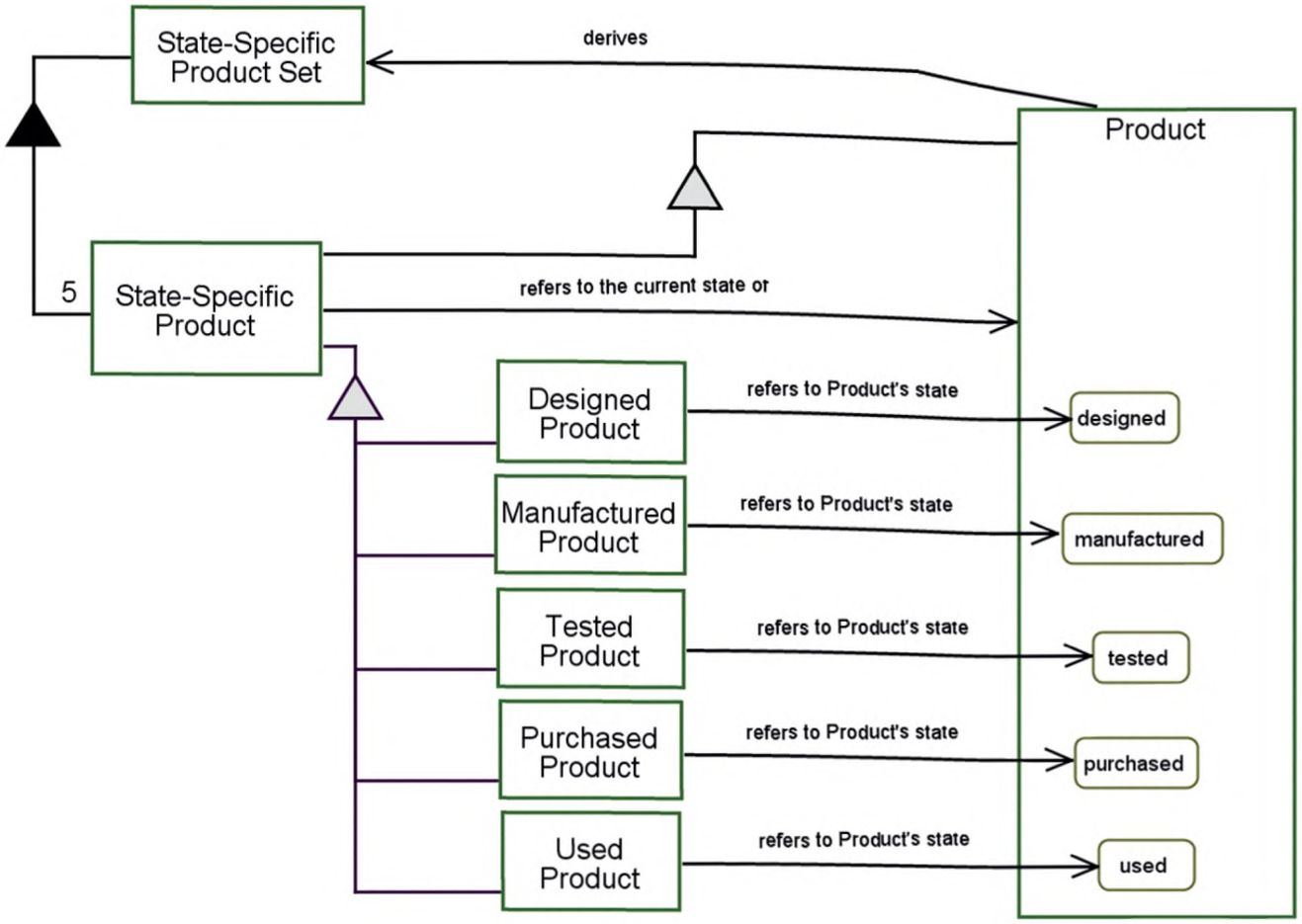
Объект **Stateful Object** представляет состояние **s**-объектов **State-Specific Objects** (Определяющие состояние объекты).

Объект **State-Specific Object Set** (Набор определяющих состояние объектов) содержит **s**-объектов **State-Specific Objects** (Определяющие состояние объекты).

Объект **State-Specific Object** (Определяющий состояние объект) связан с объектом **State** (Состояние) и имеет тег **refers to** (относится к).

Рисунок С.5 — Диаграмма OPM-модели сущности

Пример — На рисунке С.6 объект Product (Продукт) является объектом с 5 внутренними состояниями, из которых получают 5 различных специализаций объекта Product, каждое из которых относится к различным состояниям объекта Product. Таким образом, объект State-Specific Product (Определяющий состояние объект), называемый объектом Tested Product (Испытанный продукт), относится к состоянию tested (испытанный) объекта Product. Безусловно, аналогичный объект Tested Product также относится к самому объекту Product, поскольку является состоянием; состояние tested не имеет смысла без ссылки на тот объект, который находится в данном состоянии. Этим способом получают 5 объектов State-Specific Products, каждый из которых является специализацией объекта Product и имеет его конкретное состояние.



Объект **Product** (Продукт) может находиться в состоянии **designed** (разработанный), **manufactured** (изготовленный), **tested** (испытанный), **purchased** (приобретаемый) или **used** (используемый).

Объект **Product** (Продукт) производит объект **State-Specific Product Set** (Набор определяющих состояние объектов).

Объект **State-Specific Product Set** (Набор определяющих состояние объектов) содержит 5 объектов **State-Specific Products** (Определяющие состояние объекты).

Объект **State-Specific Product** (Определяющие состояние объекты) является объектом **Product** (Продукт).

Объект **State-Specific Product** (Определяющие состояние объекты) связан с текущим состоянием объекта **Product** (Продукт).

Объекты **Designed Product** (Разработанный продукт), **Manufactured Product** (Изготовленный продукт), **Tested Product** (Испытанный продукт), **Purchased Product** (Приобретенный продукт) и **Used Product** (Используемый продукт) являются объектами **State-Specific Products** (Определяющие состояние объектов).

Объект **Designed Product** (Разработанный продукт) связан с состоянием **designed** (разработанный) объекта **Product** (Продукт) и имеет тег **refers to Product's state** (относится к состоянию объекта Product).

Объект **Manufactured Product** (Изготовленный продукт) связан с состоянием **manufactured (изготовленный)** объекта **Product** (Продукт) и имеет тег **refers to Product's state** (относится к состоянию объекта Product).

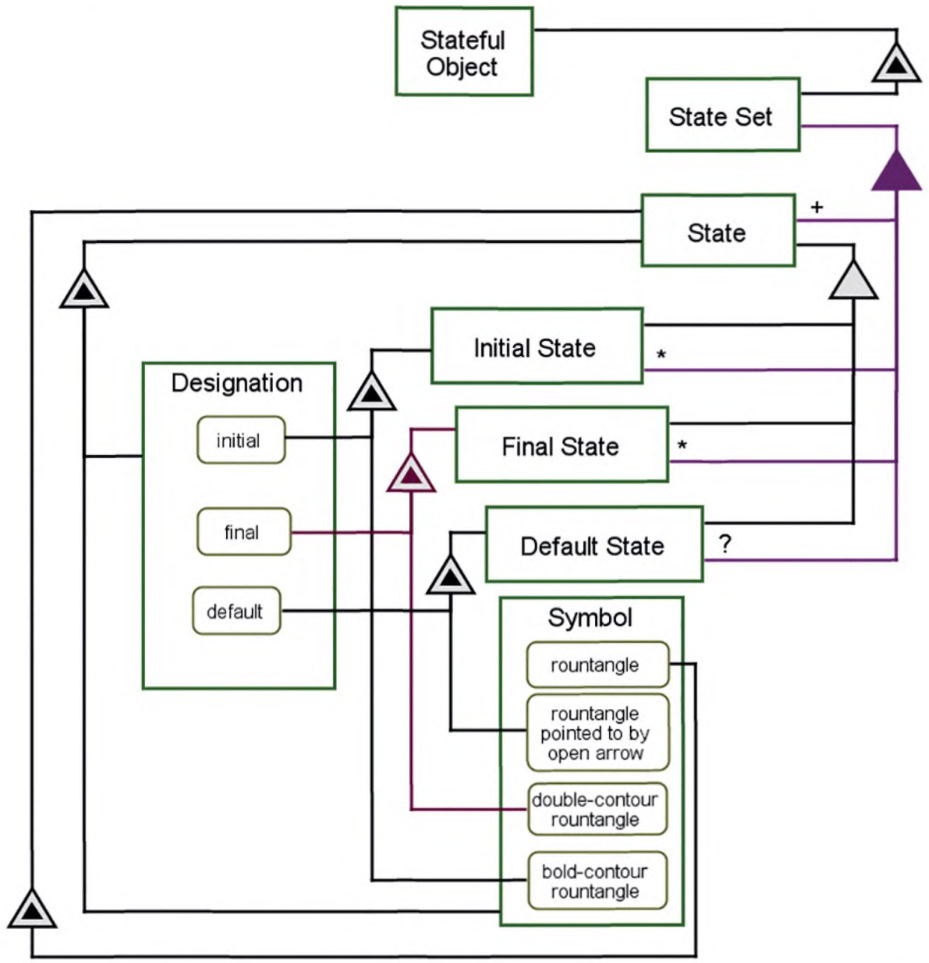
Объект **Tested Product** (Испытанный продукт) связан с состоянием **tested** (испытанный) объекта **Product** (Продукт) и имеет тег **refers to Product's state** (относится к состоянию объекта Product).

Объект **Purchased Product** (Приобретенный продукт) связан с состоянием **purchased** (приобретаемый) объекта **Product** (Продукт) и имеет тег **refers to Product's state** (относится к состоянию объекта Product).

Объект **Used Product** (Используемый продукт) связан с состоянием **used** (используемый) объекта **Product** (Продукт) и имеет тег **refers to Product's state** (относится к состоянию объекта Product).

Рисунок С.6 — Пример относящегося к состоянию объекта

На рисунке С.7 представлена OPM-модель объекта, имеющего внутреннее состояние.



Объект **Stateful Object** (Объект с внутренним состоянием) представляет объект **State Set** (Набор состояний).

Объект **State Set** (Набор состояний) содержит по меньшей мере один из объектов **State** (Состояние), дополнительный объект **Initial States** (Начальные состояния), **Final States** (Конечные состояния) и **Default State** (Состояние по умолчанию).

Объект **State** (Состояние) представляет объекты **Designation** (Обозначение) и **Symbol** (Символ).

Объект **Designation** (Обозначение) может быть в состоянии **initial** (начальное), **final** (конечное) или **default** (по умолчанию).

Объекты **Initial State** (Начальное состояние), **Final State** (Конечное состояние) и **Default State** (Состояние по умолчанию) является объектами **States** (Состояния).

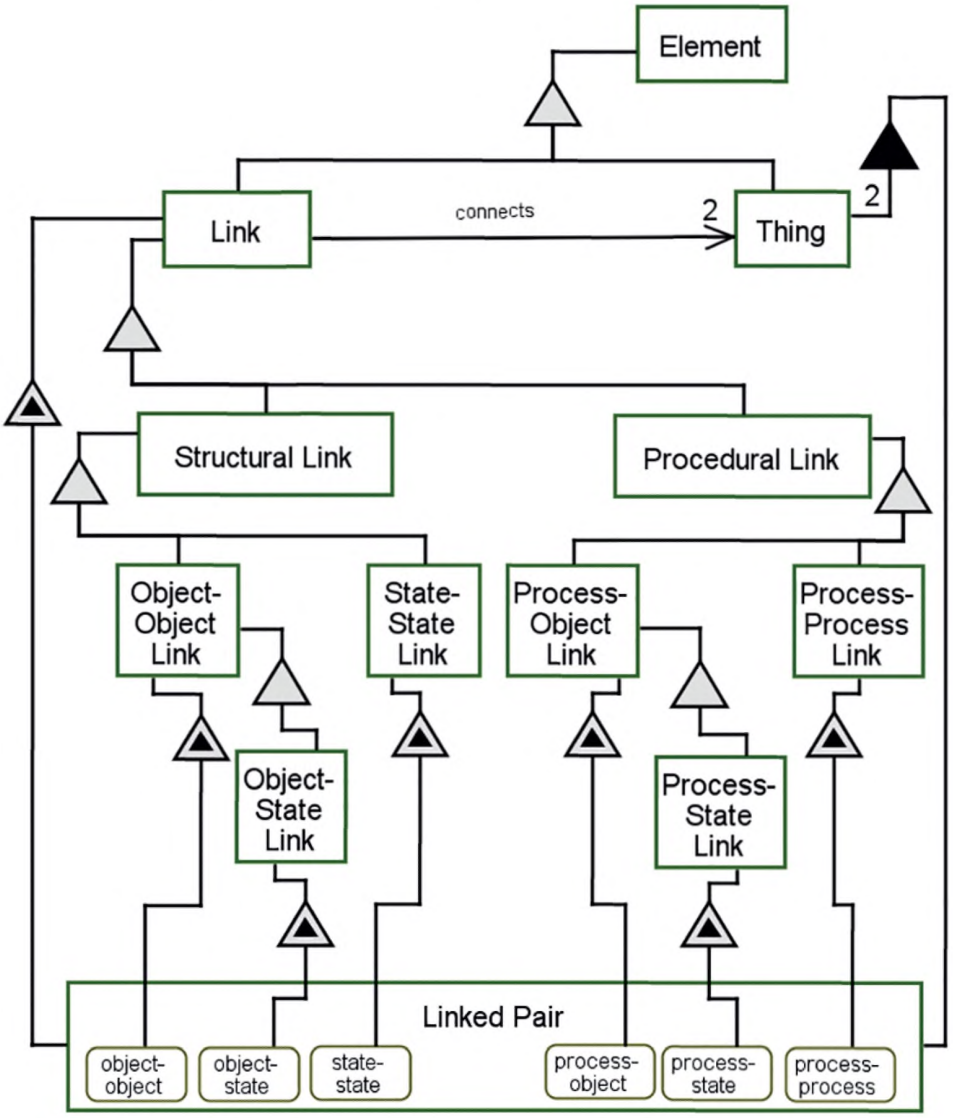
Объект **Initial State** (Начальное состояние) представляет состояние **initial** (начальное) объекта **Designation** (Обозначение) и состояние **bold-contour rountangle** (прямоугольник с закругленными углами и с жирным контуром) объекта **Symbol** (Символ) для объекта **State** (Состояние).

Объект **Final State** представляет состояние **final** (Конечное состояние) объекта **Designation** (Обозначение) и состояние **double-contour rountangle** (прямоугольник с закругленными углами и с двойным контуром) объекта **Symbol** (Символ) для объекта **State** (Состояние).

Объект **Default State** представляет состояние **default** (по умолчанию) объекта **Designation** (Обозначение) и состояние **rountangle pointed to by open arrow** (прямоугольник с закругленными углами, отмеченный стрелкой) объекта **Symbol** (Символ) для объекта **State** (Состояние).

Рисунок С.7 — Диаграмма OPM-модели объекта с внутренним состоянием

На рисунке С.8 представлена модель, применимая только к базовым структурам, поскольку объект **Link** (Связь) соединяется с 2 (и не более) объектами **Things** (Сущности) и имеет тег **connects** (соединяет).



Объекты **Thing** (Сущность) и **Link** (Связь) являются объектами **Elements** (Элементы).

Объект **Link** (Связь) соединяется с 2 объектами **Things** (Сущности) и имеет тег **connects** (соединяется).

Объект **Link** (Связь) представляет объект **Linked Pair** (Связанная пара).

Объект **Linked Pair** (Связанная пара) содержит 2 объекта **Things** (Сущности).

Объект **Linked Pair** (Связанная пара) может быть в состоянии **object-object** (объект-объект), **object-state** (объект-состояние), **state-state** (состояние-состояние), **process-object** (процесс-объект), **process-state** (процесс-состояние) или **process-process** (процесс-процесс).

Объекты **Structural Link** (Структурная связь) и **Procedural Link** (Процедурная связь) являются объектами **Links** (Связи).

Объекты **Object-Object Link** (Связь типа «объект-объект») и **State-State Link** (Связь типа «состояние-состояние») являются объектами **Structural Links** (Структурные связи).

Объект **Object-State Link** (Связь типа «объект-состояние») является объектом **Object-Object Link** (Связь типа «объект-объект»).

Объект **Object-Object Link** (Связь типа «объект-объект») представляет состояние **object-object** (объект-объект) объекта **Linked Pair** (Связанная пара).

Объект **Object-State Link** (Связь типа «объект-состояние») представляет состояние **object-state** (объект-состояние) объекта **Linked Pair** (Связанная пара).

Объект **State-State Link** (Связь типа «состояние-состояние») представляет состояние **state-state** (состояние-состояние) объекта **Linked Pair** (Связанная пара).

Рисунок С.8, лист 1 — Диаграмма OPM-модели связей

Объекты **Process-Object Link** (Связь типа «процесс-объект») и **Process-Process Link** (Связь типа «процесс-процесс») являются объектами **Procedural Links** (Процедурные связи).

Объект **Process-State Link** (Связь типа «процесс-состояние») является объектом **Process-Object Link** (Связь типа «процесс-объект»).

Объект **Process-Object Link** (Связь типа «процесс-объект») представляет состояние **process-object** (процесс-объект) объекта **Linked Pair** (Связанная пара).

Объект **Process-State Link** (Связь типа «процесс-состояние») представляет состояние **process-state** (процесс-состояние) объекта **Linked Pair** (Связанная пара).

Объект **Process-Process Link** (Связь типа «процесс-процесс») представляет состояние **process-process** (процесс-процесс) объекта **Linked Pair** (Связанная пара).

Рисунок С.8, лист 2

На рисунке С.9 изображен объект **Thing** (Сущность) с его основными свойствами — **Perseverance** (Устойчивость), **Essence** (Смысл) и **Affiliation** (Принадлежность), моделируемый как атрибут объекта типа *refinees* для связи «представление-характеризация». Объект **Perseverance** — это дискриминирующий атрибут между объектами **Object** (Объект) и **Process** (Процесс). Объект **Essence** — это дискриминирующий атрибут между объектами **Physical Object** (Физический объект) и **Physical Process** (Физический процесс) с одной стороны и объектами **Informatical Object** (Информационный объект) и **Informatical Process** (Информационный процесс) с другой стороны. Объект **Affiliation** — это дискриминирующий атрибут между объектами **Systemic Object** (Системный объект) и **Systemic Process** (Системный процесс) с одной стороны и объектами **Environmental Object** (Объект внешней среды) и **Environmental Process** (Процесс внешней среды) с другой стороны.

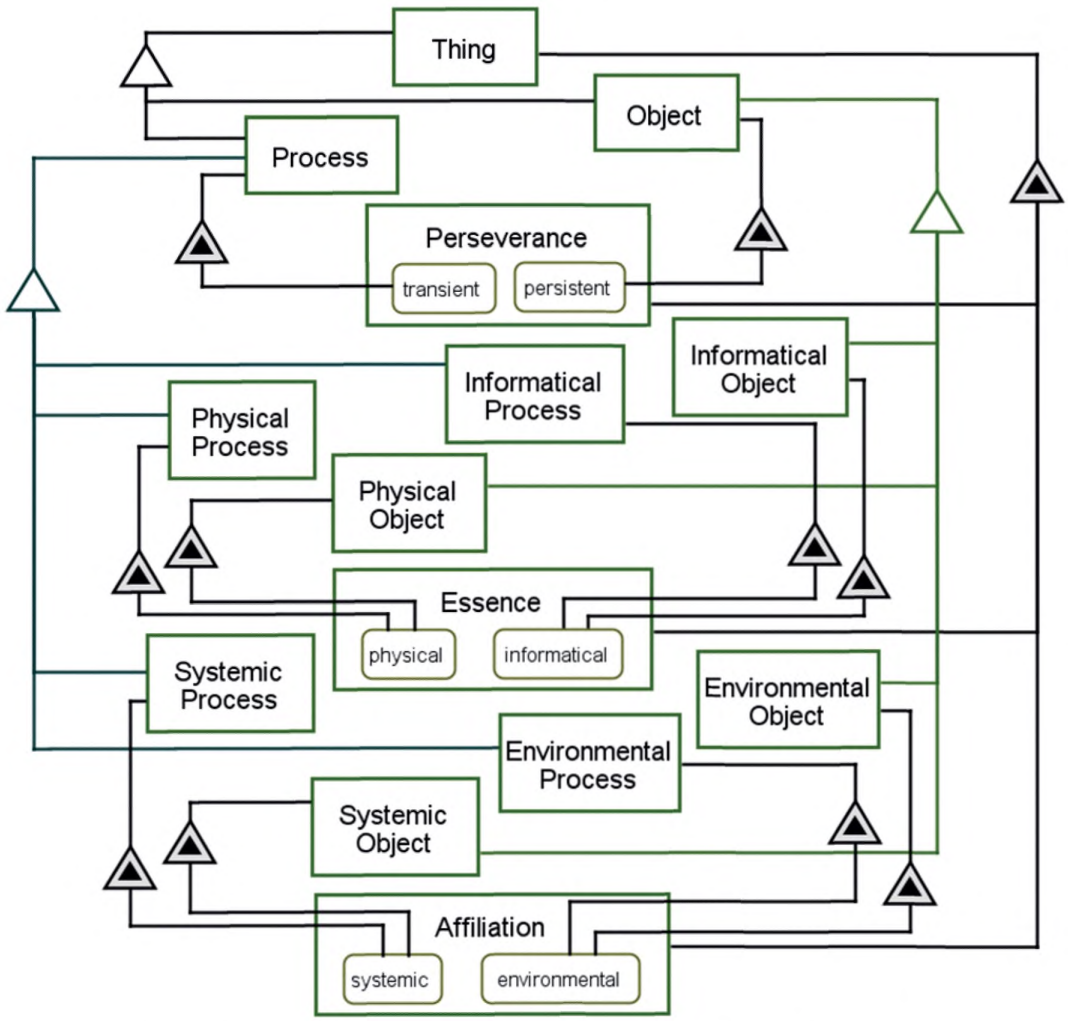


Рисунок С.9, лист 1 — Диаграмма OPM-модели общих свойств объекта **Thing**

Объект **Thing** (Сущность) представляет объекты **Perseverance** (Устойчивость), **Essence** (Смысл) и **Affiliation** (Принадлежность).

Объект **Perseverance** (Устойчивость) может иметь состояние **transient** (переходное) или **persistent** (постоянное).

Объект **Essence** (Смысл) может иметь состояние **physical** (физическое) или **informatical** (информационное).

Объект **Affiliation** (Принадлежность) может иметь состояние **systemic** (систематическое) или **environmental** (внешней среды).

Объекты **Object** (Объект) и **Process** (Процесс) являются объектами **Things** (Сущности).

Объект **Process** (Процесс) представляет состояние **transient** (переходное) объекта **Perseverance** (Устойчивость).

Объект **Object** (Объект) представляет состояние **persistent** (постоянное) объекта **Perseverance** (Устойчивость).

Объекты **Physical Process** (Физический процесс), **Informatical Process** (Информационный процесс), **Systemic Process** (Систематический процесс) и **Environmental Process** (Процесс внешней среды) являются объектами **Processes** (Процессы).

Объекты **Physical Object** (Физический объект), **Informatical Object** (Информационный объект), **Systemic Object** (Систематический объект) и **Environmental Object** (Объект внешней среды) являются объектами **Objects** (Объекты).

Объекты **Physical Process** (Физический процесс) и **Physical Object** (Физический объект) представляют состояние **physical** (физическое) объекта **Essence** (Смысл).

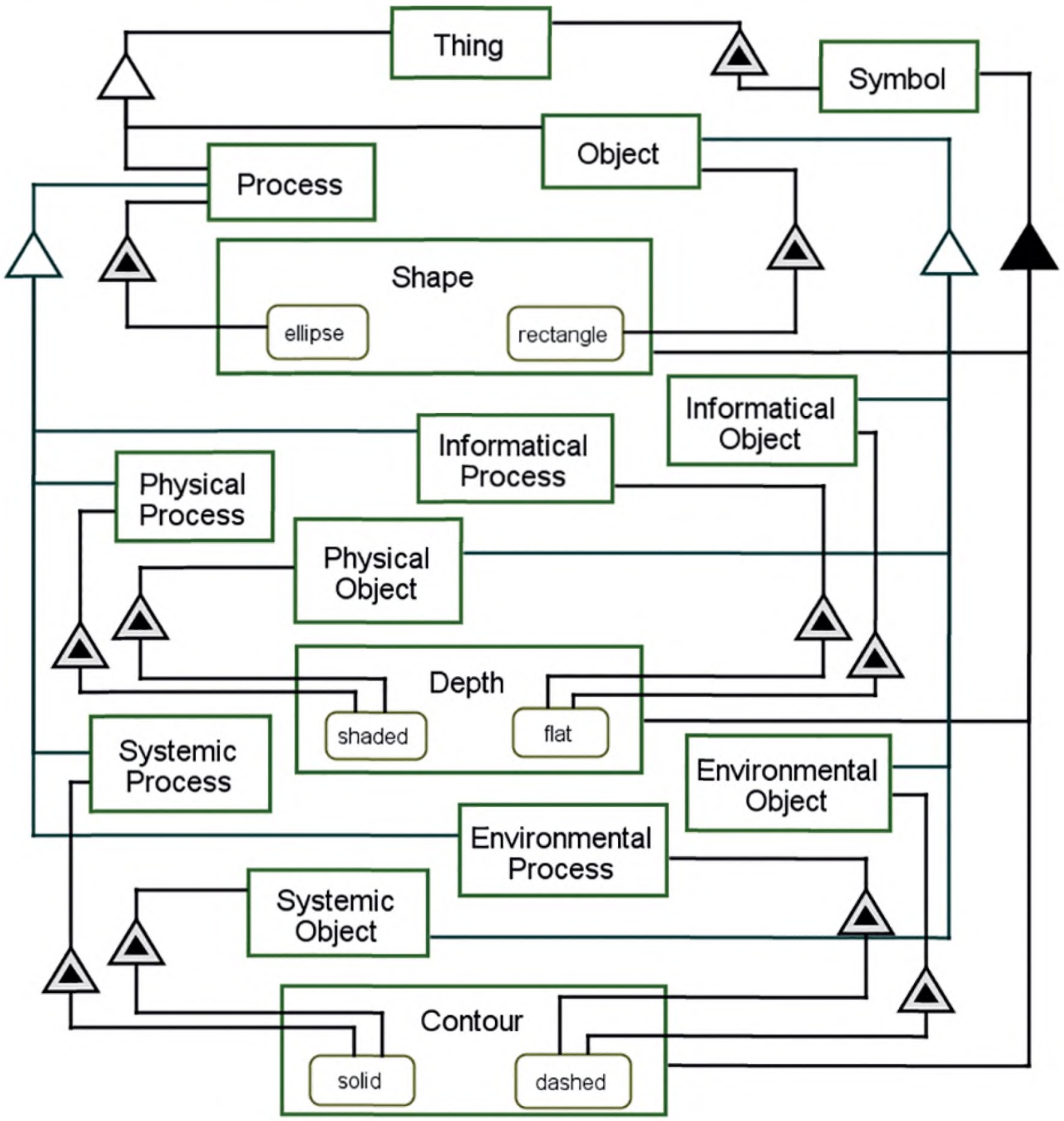
Объекты **Informatical Process** (Информационный процесс) и **Informatical Object** (Информационный объект) представляют состояние **informatical** (информационное) объекта **Essence** (Смысл).

Объекты **Systemic Process** (Системный процесс) и **Systemic Object** (Системный объект) представляют состояние **systemic** (системное) объекта **Affiliation** (Принадлежность).

Объекты **Environmental Process** (Процесс внешней среды) и **Environmental Object** (Объект внешней среды) представляют состояние **environmental** (внешней среды) объекта **Affiliation** (Принадлежность).

Рисунок С.9, лист 2

На рисунке С.10 представлена OPM-модель для графического представления OPM-сущностей, показывающая атрибут объекта типа *refinee* **Symbol** (Символ) и три его части: объекты **Shape** (Форма), **Depth** (Глубина) и **Contour** (Контур). Объект **Shape** — это часть, позволяющая различать объекты **Object** (Объект) и **Process** (Процесс). Объект **Depth** — это часть, которая позволяет различать объекты **Physical Object** (Физический объект) и **Physical Process** (Физический процесс) с одной стороны и объекты **Informatical Object** (Информационный объект) и **Informatical Process** (Информационный процесс) с другой стороны. Объект **Contour** — это часть, позволяющая различать объекты **Systemic Object** (Системный объект) и **Systemic Process** (Системный процесс) с одной стороны и объекты **Environmental Object** (Объект внешней среды) и **Environmental Process** (Процесс внешней среды) с другой стороны. Поскольку состояния объекта связаны с объектом, то объекты **Essence** (Сущность) и **Affiliation** (Принадлежность), связанные с конкретным состоянием объекта **Object**, такие же, как у объекта **Object**.



Объект **Thing** (Сущность) представляет объект **Symbol** (Символ).

Объект **Symbol** (Символ) для объекта **Thing** (Сущность) состоит из объектов **Shape** (Форма), **Depth** (Глубина) и **Contour** (Контур).

Объект **Shape** (Форма) может быть в состоянии **ellipse** (эллипс) или **rectangle** (прямоугольник).

Объект **Depth** (Глубина) может быть в состоянии **shaded** (с тенью) или **non-shaded** (без тени).

Объект **Contour** (Контур) может быть в состоянии **solid** (сплошной) или **dashed** (пунктирный).

Рисунок С.10, лист 1 — Диаграмма OPM-модели символического представления объекта **Thing**

Объекты **Process** (Процесс) и **Object** (Объект) являются объектами **Things** (Сущности).

Объект **Process** (Процесс) представляет состояние **ellipse** (эллипс) объекта **Shape** (Форма).

Объект **Object** (Объект) представляет состояние **rectangle** (прямоугольный) объекта **Shape** (Форма).

Объекты **Physical Process** (Физический процесс), **Informatical Process** (Информационный процесс), **Systemic Process** (Системный процесс) и **Environmental Process** (Процесс внешней среды) являются объектами **Processes** (Процессы).

Объекты **Physical Object** (Физический объект), **Informatical Object** (Информационный объект), **Systemic Object** (Системный объект) и **Environmental Object** (Объект внешней среды) являются объектами **Objects** (Объекты).

Объекты **Physical Process** (Физический процесс) и **Physical Object** (Физический объект) представляют состояние **shaded** (с тенью) объекта **Depth** (Глубина).

Объекты **Informatical Process** (Информационный процесс) и **Informatical Object** (Информационный объект) представляют состояние **flat** (плоское) объекта **Depth** (Глубина).

Объекты **Systemic Process** (Системный процесс) и **Systemic Object** (Системный объект) представляют состояние **solid** (сплошной) объекта **Contour** (Контур).

Объекты **Environmental Process** (Процесс внешней среды) и **Environmental Object** (Объект внешней среды) представляют состояние **dashed** (пунктирный) объекта **Contour** (Контур).

Рисунок С.10, лист 2

На рисунке С.11 представлена разновидность модели, изображенной на рисунке С.10, на которой три части атрибута объекта **Symbol** (Символ) для объекта **Thing** (Сущность) проявляются в виде 8 значений — по одному на каждую возможную конфигурацию объекта **Thing**. При этом на некоторых других рисунках с моделями в данном приложении реальные символы появляются в нижней части OPD-диаграммы. В этом случае символ находится ниже соответствующей модели объекта и значения объекта **Symbol** для объекта **Thing**. Указанные восемь символов в нижней части OPD-диаграммы являются просто иллюстративными и поэтому отличаются от приведенных на самой OPD-диаграмме. На рисунке С.11 объект **Symbol** (объект типа *refinee*) усовершенствован в сравнении с рисунком С.10 путем перечисления 8 состояний объекта **Symbol**, которые являются прямым произведением 2x2x2-значений атрибутов *refinee* объектов **Depth** (Глубина), **Contour** (Контур) и **Shape** (Форма) для объекта **Symbol**.

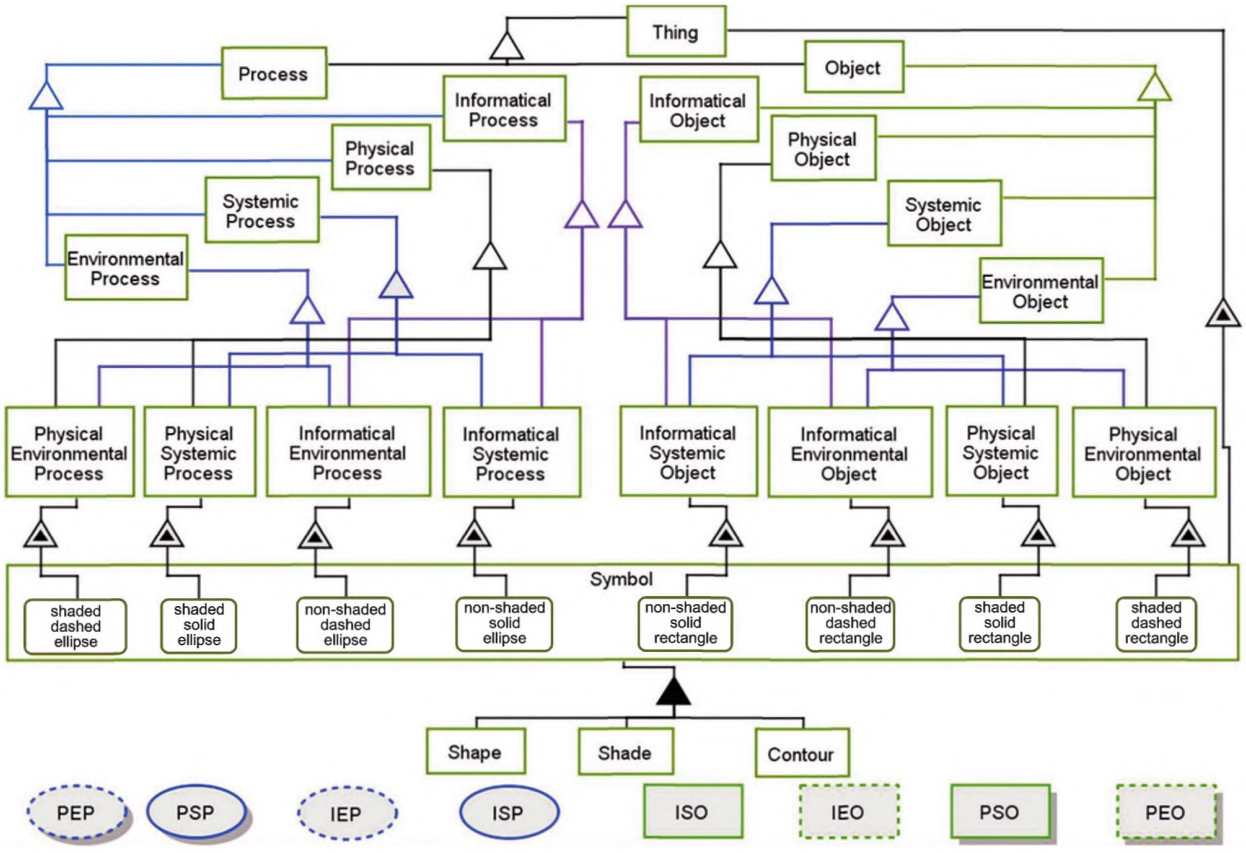


Рисунок С.11, лист 1 — Диаграмма OPM-модели восьми представлений символов объекта **Thing**

Объект **Thing** (Сущность) представляет объект **Symbol** (Символ).

Объект **Symbol** (Символ) для объекта **Thing** (Сущность) содержит объекты **Depth** (Глубина), **Contour** (Контур) и **Shape** (Форма).

Объект **Symbol** (Символ) для объекта **Thing** (Сущность) может иметь состояния **shaded** (с тенью), **dashed** (пунктирное), **rectangle** (прямоугольное), **shaded solid ellipse** (эллипс со сплошным контуром и тенью), **non-shaded dashed ellipse** (эллипс с пунктирным контуром и без тени), **non-shaded solid ellipse** (эллипс со сплошным контуром и без тени), **non-shaded solid rectangle** (прямоугольник со сплошным контуром и без тени), **non-shaded dashed rectangle** (прямоугольник с пунктирным контуром и без тени), **shaded solid rectangle** (прямоугольник со сплошным контуром и с тенью) или **shaded dashed rectangle** (прямоугольник с пунктирным контуром и тенью).

Объекты **Object** (Объект) и **Process** (Процесс) являются объектами **Things** (Сущности).

Объекты **Physical Process** (Физический процесс), **Informatical Process** (Информационный процесс), **Systemic Process** (Системный процесс) и **Environmental Process** (Процесс внешней среды) являются объектами **Processes** (Процессы).

Объекты **Physical Object** (Физический объект), **Informatical Object** (Информационный объект), **Systemic Object** (Системный объект) и **Environmental Object** (Объект внешней среды) являются объектами **Objects** (Объекты).

Физический процесс **Physical Systemic Process** (Физический системный процесс) является объектами **Physical Process** (Физический процесс) и **Systemic Process** (Системный процесс).

Физический процесс **Physical Systemic Process** (Физический системный процесс) представляет состояние **shaded solid ellipse** (эллипс со сплошным контуром и тенью) объекта **Symbol** (Символ) для объекта **Thing** (Сущность).

Процесс внешней среды **Physical Environmental Process** (Физический процесс внешней среды) является процессами **Physical Process** (Физический процесс) и **Environmental Process** (Процесс внешней среды).

Процесс внешней среды **Physical Environmental Process** (Физический процесс внешней среды) представляет состояние **shaded dashed ellipse** (эллипс с пунктирным контуром и тенью) объекта **Symbol** (Символ) для объекта **Thing** (Сущность).

Процесс **Informatical Environmental Process** (Информационный процесс внешней среды) является процессами **Informatical Process** (Информационный процесс) и **Environmental Process** (Процесс внешней среды).

Процесс **Informatical Environmental Process** (Информационный процесс внешней среды) представляет состояние **non-shaded dashed ellipse** (эллипс с пунктирным контуром и без тени) объекта **Symbol** (Символ) для объекта **Thing** (Сущность).

Процесс **Informatical Systemic Process** (Информационный системный процесс) является процессами **Informatical Process** (Информационный процесс) и **Systemic Process** (Системный процесс).

Процесс **Informatical Systemic Process** (Информационный системный процесс) представляет состояние **non-shaded solid ellipse** (эллипс со сплошным контуром и без тени) объекта **Symbol** (Символ) для объекта **Thing** (Сущность).

Объект **Physical Environmental Object** (Физический объект внешней среды) является объектами **Physical Object** (Физический объект) и **Environmental Object** (Объект внешней среды).

Объект **Physical Environmental Object** (Физический объект внешней среды) представляет состояние **shaded dashed rectangle** (прямоугольник с пунктирным контуром и тенью) **объекта Symbol** (Символ) **для объекта Thing** (Сущность).

Объект **Physical Systemic Object** (Физический системный объект) является объектами **Physical Object** (Физический объект) и **Systemic Object** (Системный объект).

Объект **Physical Systemic Object** (Физический системный объект) представляет состояние **shaded solid rectangle** (прямоугольник со сплошным контуром и тенью) объекта **Symbol** (Символ) для объекта **Thing** (Сущность).

Объект **Informatical Environmental Object** (Информационный объект внешней среды) является объектами **Informatical Object** (Информационный объект) и **Environmental Object** (Объект внешней среды).

Объект **Informatical Environmental Object** (Информационный объект внешней среды) представляет состояние **non-shaded dashed rectangle** (прямоугольник с пунктирным контуром без тени) объекта **Symbol** (Символ) для объекта **Thing** (Сущность).

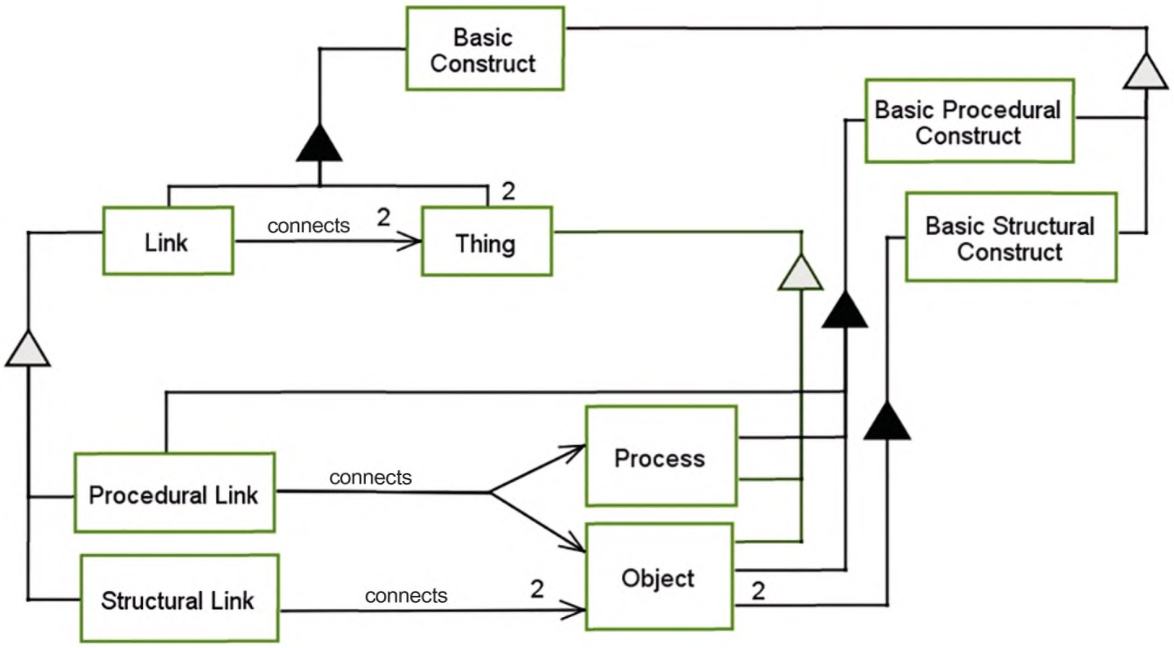
Объект **Informatical Systemic Object** (Информационный системный объект) является объектами **Informatical Object** (Информационный объект) и **Systemic Object** (Системный объект).

Объект **Informatical Systemic Object** (Информационный системный объект) представляет состояние **non-shaded solid rectangle** (прямоугольник со сплошным контуром без тени) объекта **Symbol** (Символ) для объекта **Thing** (Сущность).

Объект **Symbol** (Символ) для объекта **Thing** (Сущность) содержит объекты **Depth** (Глубина), **Contour** (Контур) и **Shape** (Форма).

Рисунок С.11, лист 2

Модель, приведенная на рисунке С.12 применима только для базовых структур, поскольку объект **Link** (Связь) соединяется с 2 (но не более чем) объектами **Things** (Сущности) и имеет тег **connects** (соединяет).



Объект **Basic Construct** (Базовая структура) состоит из объекта **Link** (Связь) и 2 объектов **Things** (Сущности).

Объект **Link** (Связь) соединяется с 2 объектами **Things** (Сущности).

Объекты **Structural Link** (Структурная связь) и **Procedural Link** (Процедурная связь) являются объектами **Links** (Связи).

Объекты **Basic Structural Construct** (Базовая структурная конструкция) и **Basic Procedural Construct** (Базовая процедурная структура) являются объектами **Basic Constructs** (Базовые структуры).

Объект **Basic Structural Construct** (Базовая структурная конструкция) состоит из объекта **Structural Link** (Структурная связь) и 2 объектов **Objects** (Объекты).

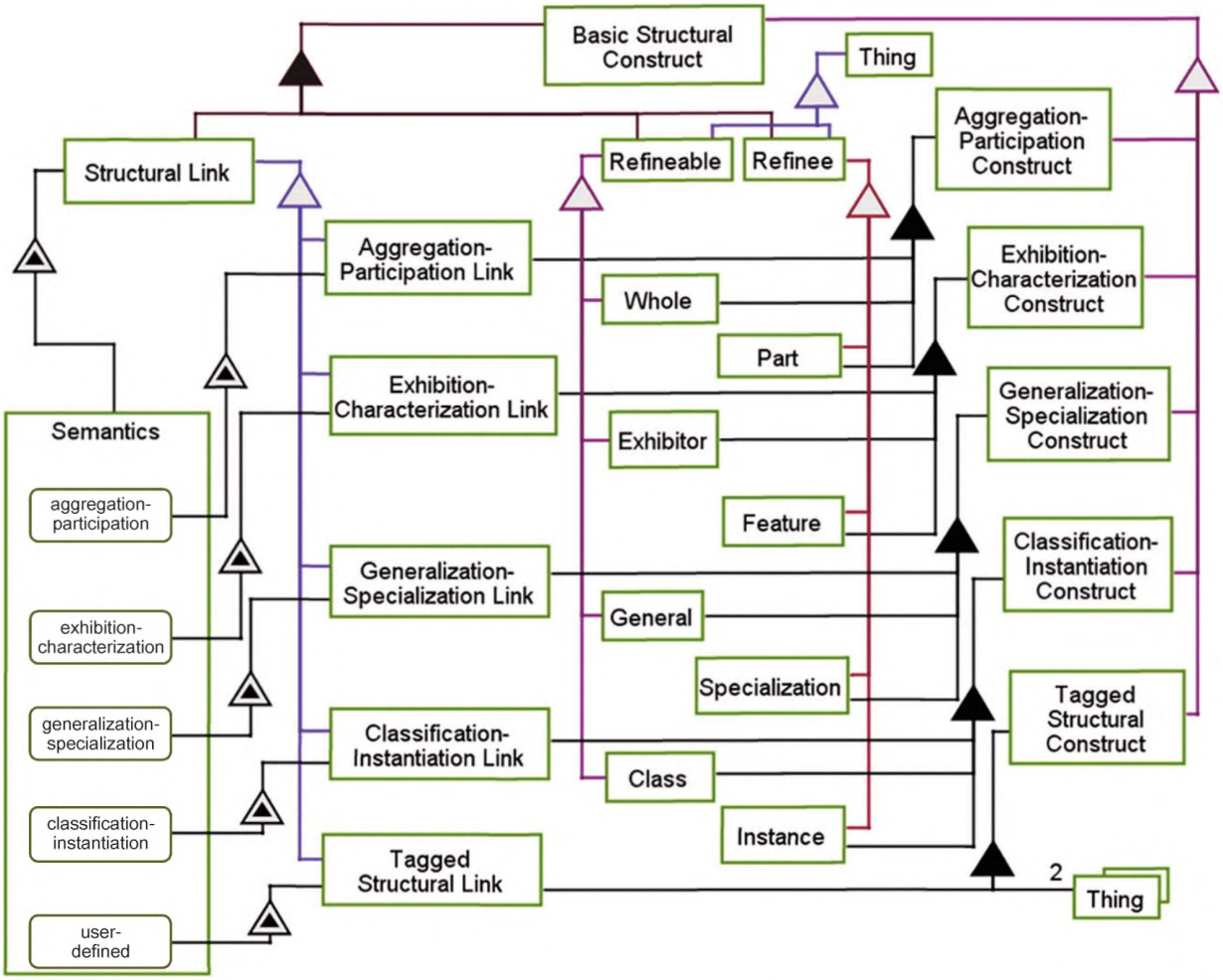
Объект **Basic Procedural Construct** (Базовая процедурная структура) содержит объекты **Procedural Link** (Процедурная связь), **Object** (Объект) и **Process** (Процесс).

Объект **Structural Link** (Структурная связь) содержит 2 объекта **Objects** (Объекты).

Объект **Procedural Link** (Процедурная связь) содержит объекты **Process** (Процесс) и **Object** (Объект) и имеет теги **connects** (соединяет).

Рисунок С.12 — Разработка объекта **Basic Construct**

На рисунке С.13 представлена OPM-модель объекта **Basic Structural Construct** (Базовая структурная конструкция).



Объект **Basic Structural Construct** (Базовая структурная конструкция) содержит объекты типа **Refineable**, **Refinee и Structural Link** (Структурная связь).

Объекты типа **Refineable** и **Refinee** являются объектами **Things** (Сущности).

Объекты **Whole** (Совокупность), **Exhibitor** (Представитель), **General** (Общие сведения) и **Class** (Класс) являются объектами **Refineables.**

Объекты **Part** (Часть), **Feature** (Признак), **Specialization** (Специализация) и **Instance** (Экземпляр) являются объектами типа **Refinees.**

Объект **Structural Link** (Структурная связь) представляет объект **Semantics** (Семантика).

Объект **Semantics** (Семантика) для объекта **Structural Link** (Структурная связь) может иметь состояния **aggregation-participation** (агрегация — является частью), **exhibition-characterization** (представление-характеризация), **generalization-specialization** (обобщение-специализация), **classification-instantiation** (классификация-инстанцирование) или **user-defined** (определяемая пользователем).

Объекты **Aggregation-Participation Link** (Связь типа «агрегация — является частью»), **Exhibition-Characterization Link** (Связь типа «представление-характеризация»), **Generalization-Specialization Link** (Связь типа «обобщение-специализация»), **Classification-Instantiation Link** (Связь типа «классификация-инстанцирование») и **Tagged Structural Link** (Тегированная структурная связь) являются объектами **Structural Links** (Структурные связи).

Объект **Aggregation-Participation Link** (Связь типа «агрегация — является частью») представляет состояние **aggregation-participation** (агрегация — является частью) объекта **Semantics** (Семантика).

Объект **Exhibition-Characterization Link** (Связь типа «представление-характеризация») представляет состояние **exhibition-characterization** (представление-характеризация) объекта **Semantics** (Семантика).

Объект **Generalization-Specialization Link** (Связь типа «обобщение-специализация») представляет состояние **generalization-specialization** (обобщение-специализация) объекта **Semantics** (Семантика).

Рисунок С.13, лист 1 — OPM-модель объекта **Basic Structural Construct**

Объект **Classification-Instantiation Link** (Связь типа «классификация-инстанцирование») представляет состояние **classification-instantiation** (классификация-инстанцирование) объекта **Semantics** (Семантика).

Объект **Tagged Structural Link** (Тегированная структурная связь) представляет состояние **user-defined** (определяемое пользователем) объекта **Semantics** (Семантика).

Объекты **Aggregation-Participation Construct** (Структура типа «агрегация — является частью»), **Exhibition-Characterization Construct** (Структура типа «представление-характеризация»), **Generalization-Specialization Construct** (Структура типа «обобщение-специализация»), **Classification-Instantiation Construct** (Структура типа «классификация-инстанцирование») и **Tagged Structural Construct** (Тегированная структурная конструкция) являются объектами **Basic Structural Constructs** (Базовые структурные конструкции).

Объект **Aggregation-Participation Construct** (Структура типа «агрегация — является частью») содержит объекты **Aggregation-Participation Link** (Связь типа «агрегация — является частью»), **Whole** (Совокупность) и **Part** (Часть).

Объект **Exhibition-Characterization Construct** состоит из объектов **Exhibition-Characterization Link** (Структура типа «представление-характеризация»), **Exhibitor** (Представитель) и **Feature** (Признак).

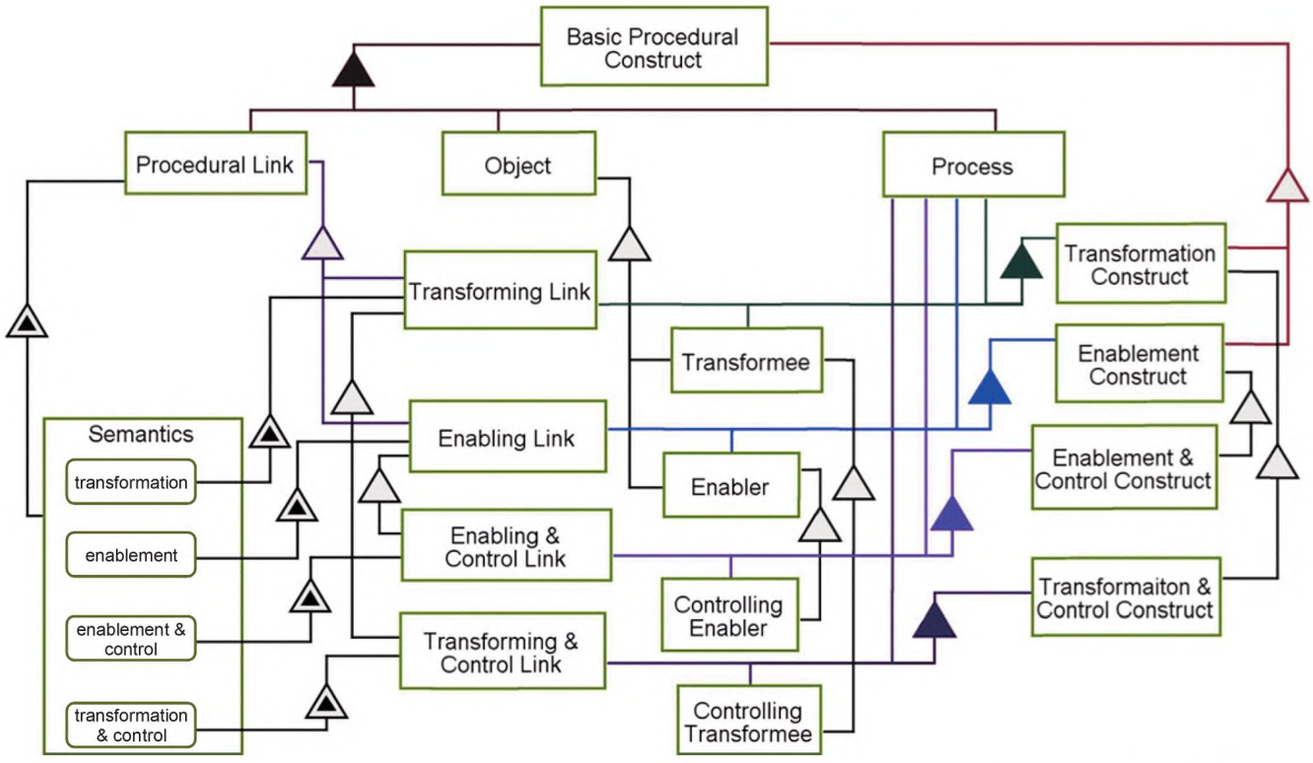
Объект **Generalization-Specialization Construct** (Структура типа «обобщение-специализация») состоит из объектов **Generalization-Specialization Link** (Связь типа «обобщение-специализация»), **General** (Общие сведения) и **Specialization** (Специализация).

Объект **Classification-Instantiation Construct** (Структура типа «классификация-инстанцирование») состоит из объектов **Classification-Instantiation Link** (Связь типа «классификация-конкретизация»), **Class** (Класс) и **Instance** (Экземпляр).

Объект **Tagged Structural Construct** (Тегированная структурная конструкция) содержит объекты **Tagged Structural Link** (Тегированная структурная связь) и 2 объекта **Things** (Сущности).

Рисунок С.13, лист 2

На рисунке С.14 представлена OPM-модель объекта **Basic Procedural Construct** (Базовая процедурная структура).



Объект **Basic Procedural Construct** (Базовая процедурная структура) содержит объекты **Object** (Объект), **Process** (Процесс) и **Procedural Link** (Процедурная связь).

Объект **Procedural Link** (Процедурная связь) представляет объект **Semantics** (Семантика).

Объект **Semantics** (Семантика) для объекта **Procedural Link** (Процедурная связь) может иметь состояния **transformation** (преобразование), **enablement** (внедрение), **transformation & control** (преобразование & управление) и **enablement & control** (внедрение & управление).

Объект типа **Transformee** и объект **Enabler** (Средство реализации) являются объектами **Objects** (Объекты).

Объект **Controlling Transformee** является объектом типа **Transformee.**

Объект **Controlling Enabler** (Контролирующее средство реализации) является объектом **Enabler** (Средство реализации).

Рисунок С.14, лист 1 — OPM-модель объекта **Basic Procedural Construct**

Объекты **Transforming Link** (Преобразующая связь) и **Enabling Link** (Разрешающая связь) являются объектами **Procedural Links** (Процедурные связи).

Объект **Transforming & Control Link** (Преобразующая & управляющая связь) является объектом **Transforming Link** (Преобразующая связь).

Объект **Enabling & Control Link** (Разрешающая & управляющая связь) является объектом **Enabling Link** (Разрешающая связь).

Объект **Transforming Link** (Преобразующая связь) представляет состояние **transformation** (преобразование) объекта **Semantics** (Семантика) для объекта **Procedural Link** (Процедурная связь).

Объект **Enabling Link** (Разрешающая связь) представляет состояние **enablement** (внедрение) объекта **Semantics** (Семантика) для объекта **Procedural Link** (Процедурная связь).

Объект **Transforming & Control Link** (Преобразующая & управляющая связь) представляет состояние **transformation & control Semantics** (Семантика преобразования & управления) объекта **Procedural Link** (Процедурная связь).

Объект **Enabling & Control Link** (Разрешающая & управляющая связь) представляет состояние **enablement & control Semantics** (семантика разрешения & управления) объекта **Procedural Link** (Процедурная связь).

Объекты **Transformation Construct** (Структура преобразования) и **Enablement Construct** (Структура разрешения) являются объектами **Basic Procedural Constructs** (Базовые процедурные структуры).

Объект **Transformation Construct** (Структура преобразования) состоит из объектов **Transforming Link** (Преобразующая связь), объекта типа **Transformee** и объекта **Process** (Процесс).

Объект **Enablement Construct** (Структура разрешения) состоит из объектов **Enablement Link** (Разрешающая связь), **Enabler** (Средство реализации) и **Process** (Процесс).

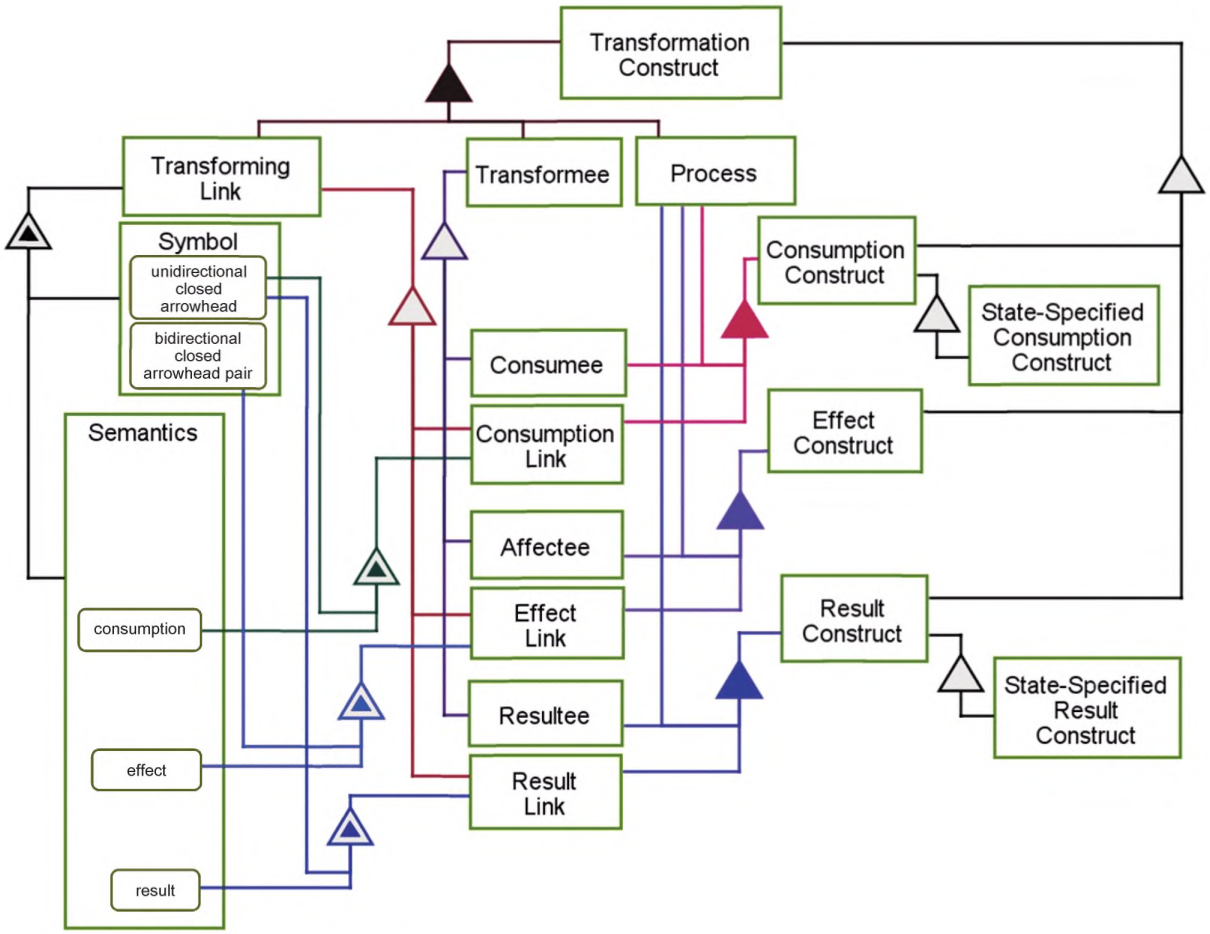
Объект **Transformation & Control Construct** (Структура преобразования & управления) является объектом **Transformation Construct** (Структура преобразования).

Объект **Enablement & Control Construct** (Структура разрешения & управления) является объектом **Enablement Construct** (Структура разрешения).

Объект **Transformation & Control Construct** (Структура преобразования & управления) состоит из объектов **Transforming & Control Link** (Преобразующая & управляющая связь), **Controlling Transformee** и **Process** (Процесс).

Объект **Enablement & Control Construct** (Структура разрешения & управления) состоит из объектов **Enablement & Control Link** (Разрешающая & управляющая связь), **Controlling Enabler** (Контролирующее средство реализации) и **Process** (Процесс).

Рисунок С.14, лист 2



Объект **Transformation Construct** (Преобразующая структура) включает в себя объект типа **Transformee**, объекты **Process** (Процесс) и **Transforming Link** (Преобразующая связь).

Объект **Transforming Link** (Преобразующая связь) представляет объекты **Symbol** (Символ) и **Semantics** (Семантика).

Объект **Symbol** (Символ), связанный с объектом **Transforming Link** (Преобразующая связь), может находиться в состоянии **unidirectional closed arrowhead** (однонаправленная стрелка) или **bidirectional closed arrowhead pair** (двунаправленная стрелка).

Объект **Semantics** (Семантика), связанный с объектом **Transforming Link** (Преобразующая связь), может находиться в состоянии **consumption** (потребительское), **effect** (действующее) или **result** (результирующее).

Объекты **Consumption Link** (Потребительская связь), **Effect Link** (Действующая связь) и **Result Link** (Результирующая связь) являются объектами **Transforming Links** (Преобразующая связь).

Объекты типа **Consumee, Affectee** и **Resultee** являются объектами типа **Transformees**.

Объекты **Consumption Construct** (Потребительская структура), **Result Construct** (Результирующая структура) и **Effect Construct** (Действующая структура) являются объектами **Transformation Constructs** (Преобразующая структура).

Объект **Consumption Construct** (Потребительская структура) содержит объекты **Consumption Link** (Потребительская связь), **Process** (Процесс) и объект типа **Consumee**.

Объект **Effect Construct** (Действующая структура) связан с объектами **Effect Link** (Действующая связь), **Process** (Процесс) и объектом типа **Affectee**.

Объект **Result Construct** (Результирующая структура) связан с объектами **Result Link** (Результирующая связь), **Process** (Процесс) и объектом типа **Resultee**.

Объект **Consumption Link** (Потребительская связь) представлен состоянием **unidirectional closed arrowhead** (однонаправленная стрелка) объекта **Symbol** (Символ), связанного с объектом **Transforming Link** (Преобразующая связь) и состоянием **consumption** (потребительское) объекта **Semantics** (Семантика), связанного с объектом **Transforming Link** (Преобразующая связь).

Объект **Effect Link** (Действующая связь) представлен состоянием **bidirectional closed arrowhead consumption pair** (двунаправленная стрелка) объекта **Transforming Link** (Преобразующая связь) и состоянием **effect** (действующее) объекта **Semantics** (Семантика), связанного с объектом **Transforming Link** (Преобразующая связь).

Рисунок С.15, лист 1 — OPM-модель объекта **Transformation Construct**

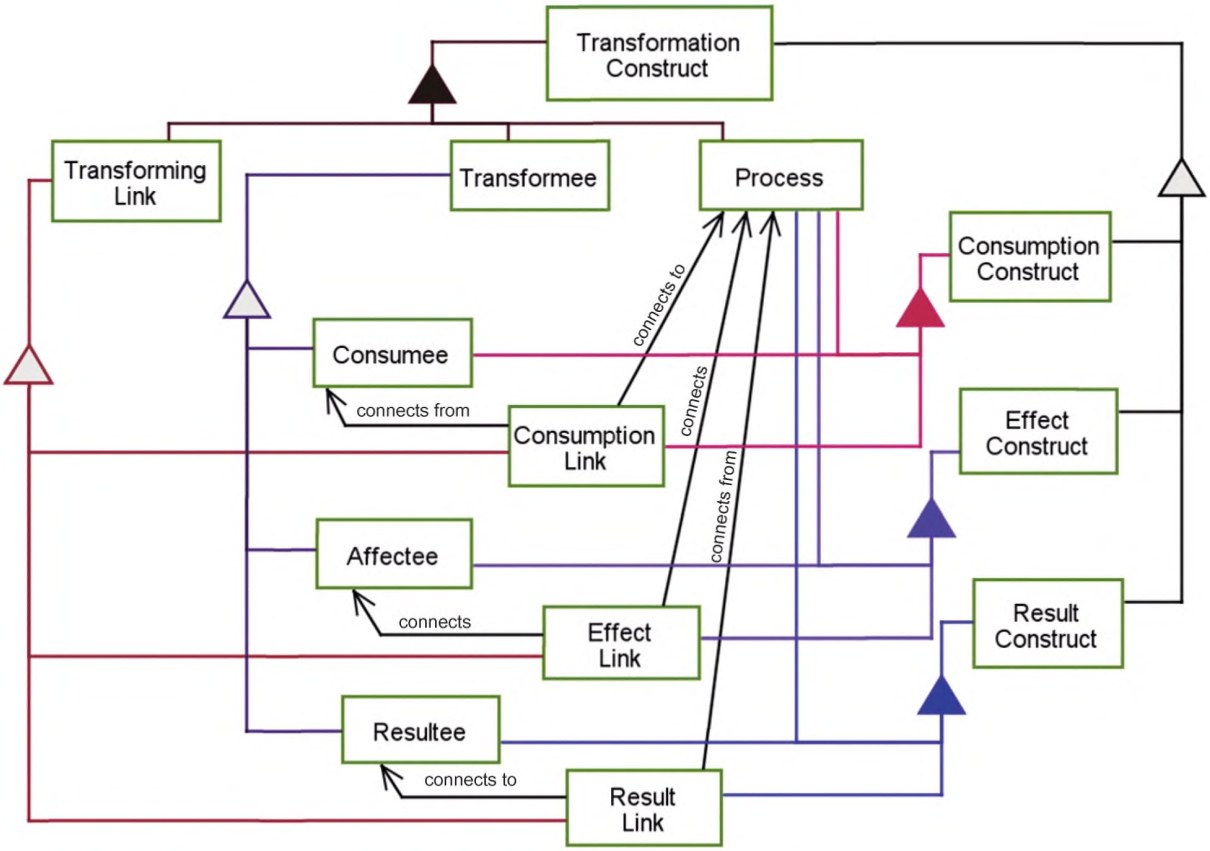
Объект **Result Link** (Результирующая связь) представлен состоянием **unidirectional closed arrowhead** (однонаправленная стрелка) объекта **Symbol** (Символ), связанного с объектом **Transforming Link** (Преобразующая связь) и состоянием **result** (результирующее) объекта **Semantics** (Семантика), связанного с объектом **Transforming Link** (Преобразующая связь).

Объект **State-Specified Consumption Construct** (Определяющая состояние потребительская структура) включает объект **Consumption Construct** (Потребительская структура).

Объект **State-Specified Result Construct** (Определяющая состояние результирующая структура) включает объект **Result Construct** (Результирующая структура).

Рисунок С.15, лист 2

Рисунок С.16 дополняет рисунок С.15, внося в него информацию относительно направленности символов в виде стрелок, соединяющих объекты с процессами. Внесение этой информации в рисунок С.15 могло бы перегрузить диаграмму и создать трудности в ее восприятии.



Объект **Transformation Construct** (Преобразующая структура) включает в себя объект типа **Transformee**, объекты **Process** (Процесс) и **Transforming Link** (Преобразующая связь).

Объекты **Consumption Link** (Потребительская связь), **Effect Link** (Действующая связь) и **Result Link** (Результирующая связь) являются объектами **Transforming Links** (Преобразующая связь).

Объекты **Consumption Construct** (Потребительская структура), **Result Construct** (Результирующая структура) и **Effect Construct** (Действующая структура) являются объектами **Transformation Constructs** (Преобразующая структура).

Объект **Consumption Construct** (Потребительская структура) содержит объекты **Consumption Link** (Потребительская связь), **Process** (Процесс) и объект типа **Consumee**.

Объект **Effect Construct** (Действующая структура) включает в себя объекты **Effect Link** (Действующая связь), **Process** (Процесс) и объект типа **Affectee**.

Объект **Result Construct** (Результирующая структура) связан с объектами **Result Link** (Результирующая связь), **Process** (Процесс) и объект типа **Resultee**.

Объект **Consumption Link** (Потребительская связь) связан с объектом типа **Consumee** и имеет тег **connects from** (соединение от...).

Объект **Consumption Link** связан с объектом **Process** (Процесс) и имеет тег **connects to** (соединение с ...).

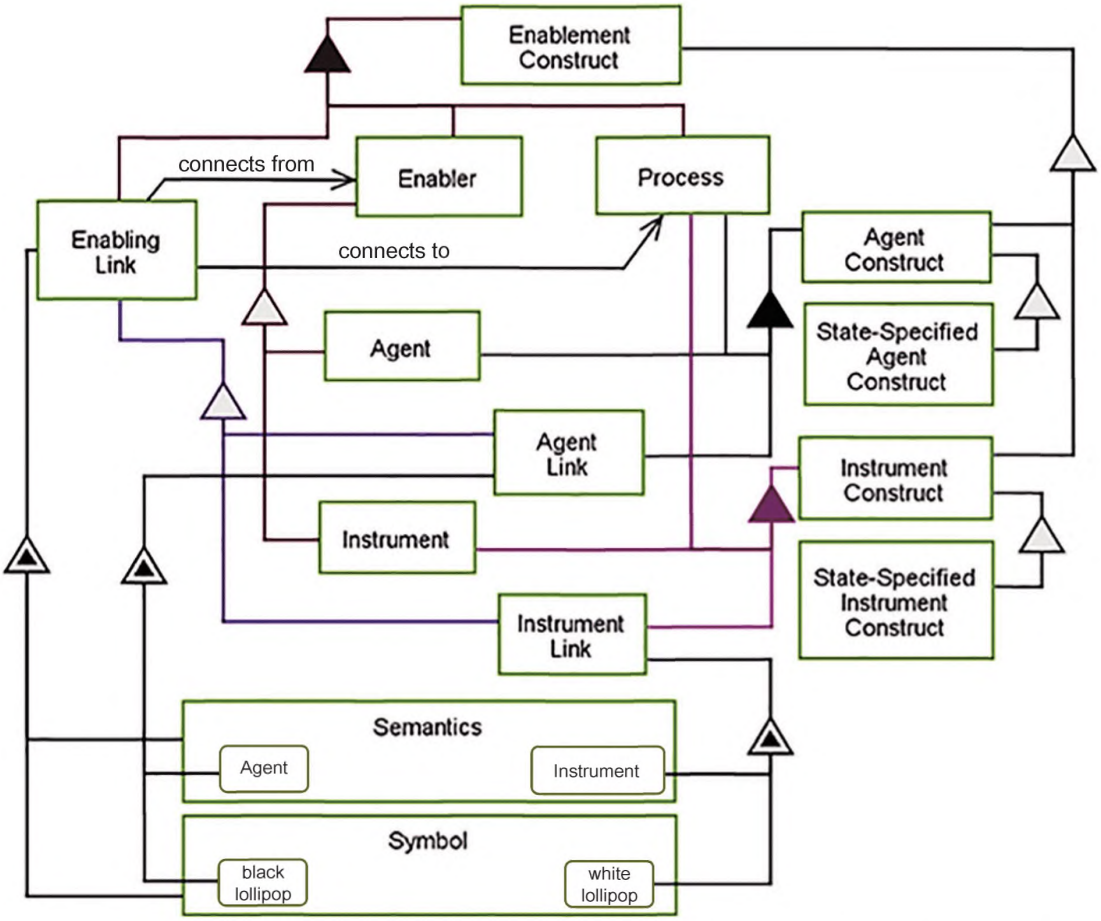
Объект **Effect Link** связан с объектами **Affectee** и **Process** (Процесс) и имеет теги **connects** (соединения).

Объект **Result Link** связан с объектом **Resultee** и имеет тег (соединение с...).

Объект **Result Link** связан с объектом **Process** (Процесс) и имеет тег **connects from** (соединение от...).

Рисунок С.16 — OPM-модель направленности связей объекта **Transformation Construct**

На рисунке С.17 представлена OPM-модель для элемента **Basic Enablement Construct** (Базовая реализующая структура).



Объект **Enablement Construct** (Реализующая структура) состоит из объектов **Enabler** (Средство реализации), **Process** (Процесс) и **Enabling Link** (Разрешающая связь).

Объект **Enabling Link** (Разрешающая связь) представлен объектами **Semantics** (Семантика) и **Symbol** (Символ).

Объект **Enabling Link** (Разрешающая связь) связан с объектом **Enabler** (Средство реализации) и имеет тег **connects from** (соединение от…).

Объект **Enabling Link** (Разрешающая связь) связан с объектом **Process** (Процесс) и имеет тег **connects to** (соединение с…).

Объект **Semantics** (Семантика), связанный с объектом **Enabling Link** (Разрешающая связь), может быть объектом **Agent** (Агент) или **Instrument** (Инструментальное средство).

Объект **Symbol** (Символ), связанный с объектом **Enabling Link** (Разрешающая связь), может иметь состояние **black lollipop** (черный «леденец на палочке») или **white lollipop** (белый «леденец на палочке»).

Объекты **Agent** (Агент) и **Instrument** (Инструментальное средство) являются объектами **Enablers** (Средства реализации).

Объекты **Agent Link** (Агентская связь) и **Instrument Link** (Инструментальная связь) являются объектами **Enabling Links** (Разрешающие связи).

Объект **Agent Link** (Агентская связь) представлен состоянием **agent** (агентское) объекта **Semantics** (Семантика), связанного с объектом **Enabling Link** (Разрешающая связь), и состоянием **black lollipop** (черный «леденец на палочке») объекта **Symbol** (Символ), связанного с объектом **Enabling Link** (Разрешающая связь).

Объект **Instrument Link** (Инструментальная связь) представлен состоянием **instrument** (инструментальное) объекта **Semantics** (Семантика), связанного с объектом **Enabling Link** (Разрешающая связь), и состоянием (белый «леденец на палочке») объекта **Symbol** (Символ), связанного с объектом **Enabling Link** (Разрешающая связь).

Объекты **Agent Construct** (Агентская структура) и **Instrument Construct** (Инструментальная структура) являются объектами **Enablement Constructs** (Реализирующие структуры).

Рисунок С.17, лист 1 — OPM-модель объекта **Basic Enablement Construct**

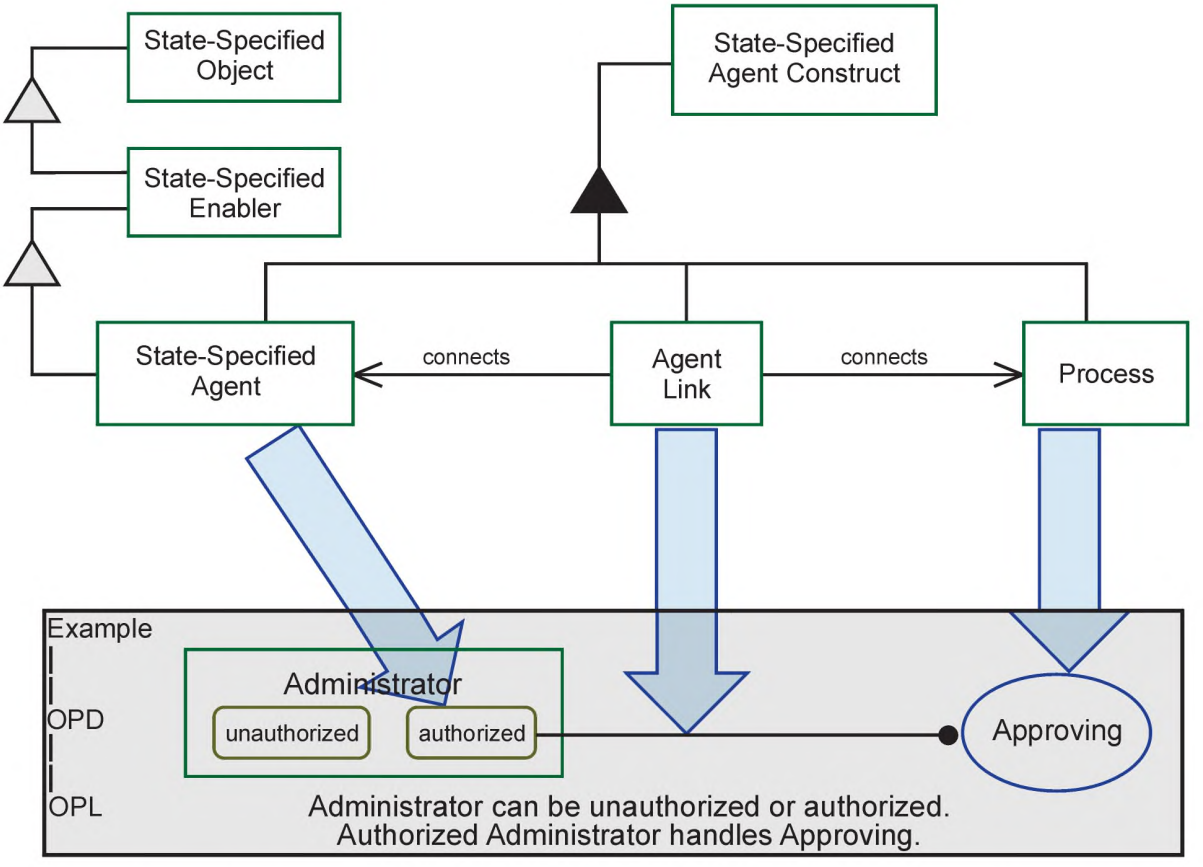
Объект **Agent Construct** содержит объекты **Agent** (Агент), **Process** (Процесс) и **Agent Link** (Агентская связь).

Объект **Instrument** (Инструментальное средство) содержит объекты **Instrument Construct** (Инструментальная структура), **Process** (Процесс) и **Instrument Link** (Инструментальная связь).

Объектом **State-Specified Agent Construct** (Определяющая состояние агента структура) является объект **Agent Construct** (Агентская структура).

Объектом **State-Specified Instrument Construct** (Определяющая состояние инструмента структура) является объектом **Instrument Construct** (Инструментальная структура).

Рисунок С.17, лист 2



Объект **State-Specified Agent Construct** (Определяющая состояние агента структура) содержит объекты **State-Specified Agent Construct** (Определяющий состояние агент), **Process** (Процесс) и **Agent Link** (Агентская связь).

Объектом **State-Specified Agent** (Определяющий состояние агент) является **State-Specified Enabler** (Определяющее состояние средство реализации).

Объектом **State-Specified Enabler** (Определяющее состояние средство реализации) является объект **State-Specified Object** (Определяющий состояние объект).

Объект **Agent Link** (Агентская связь) соединяется с объектами **State-Specified Agent** (Определяющий состояние агент) и **Process** (Процесс) и имеет теги **connects** (соединение).

Example — Пример, Administrator — Администратор, unauthorized — не обладающий соответствующими правами, authorized — обладающий соответствующими правами, Approving — Процесс утверждения, Administrator can be unauthorized or authorized — Администратор может обладать или не обладать соответствующими правами, Authorized Administrator handles Approving — Обладающий соответствующими правами Администратор может работать с процессом утверждения.

Рисунок С.18 — OPM-модель объекта **State-Specified Agent Construct** с приведенным примером

На рисунке С.18 изображены две OPM-модели: сверху — модель, выражающая наиболее существенные связи для объекта **State-Specified Agent Construct** (Определяющая состояние агента структура), и снизу — модель, выражающая соответствующую структуру модели. Первая модель создает метамодель для второй модели. Широкие стрелки на рисунке показывают направление от концептуальных частей структуры к OPD-символам данного примера. Под OPD-диаграммой в данном примере приведен соответствующий OPL-синтаксис.

Для целей обучения аналогичные диаграммы могут выражать связь между OPM-концептуальными структурными моделями и соответствующими прикладными OPM-моделями.

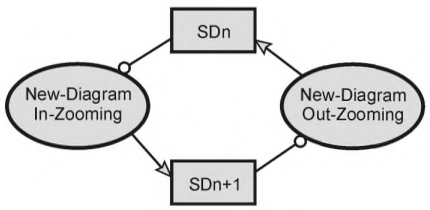
С.5 Детализированные и обобщенные модели

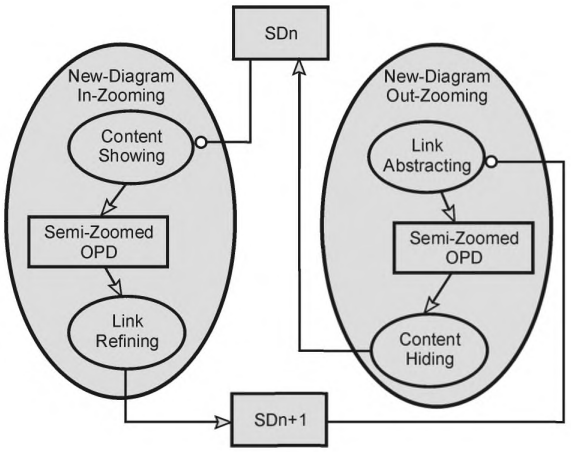
С.5.1 Способы детализации и обобщения OPD-диаграмм

Обе новые диаграммы (детализированные/обобщенные) позволяют создавать новый OPD-контекст из существующего OPD-контекста. Новая детализированная (in-zooming) диаграмма основывается на OPD-диаграмме с относительно небольшим числом деталей; в нее вводятся новые элементы или уточняются существующие элементы предшествующей OPD-диаграммы, которая была применима к определенной сущности с менее детализованной OPD-диаграммой. Новая обобщенная (out-zooming) диаграмма основывается на OPD-диаграмме с относительно большим числом деталей; из нее удаляются введенные или уточненные элементы с целью получения менее детализированной и более абстрактной, чем в предшествующем контексте, сущности.

На новой детализированной диаграмме конкретизируются сущности типа *refineable*, присутствующие на предыдущей OPD-диаграмме (называемой SDn-диаграммой), путем создания новой OPD-диаграммы (называемой SDn+1-диаграммой), путем конкретизации сущностей типа *refineable* — введения дополнительных подпроцессов, связанных с ними объектов и соответствующих связей. Новые детализированные и обобщенные процессы являются взаимно-противоположными.

На рисунке С.19 изображены процессы **New-Diagram In-Zooming** (Новая детализированная диаграмма) и **New-Diagram Out-Zooming** (Новая обобщенная диаграмма) для соответствующих процессов. Модель справа соответствует детализированной модели, изображенной слева и предназначенной для конкретизации двух процессов: одного процесса — создания нового детализированного контекста, второго процесса — для выработки обобщенного контекста. Процесс **New-Diagram In-Zooming** начинается с выполнения процесса **Content Showing** (Отображение контекста), с последующим выполнением процесса **Link Refining** (Детализация связей). Процесс **New-Diagram Out-Zooming** начинается с выполнения процесса **Link Abstracting** (Обобщение связей) — процесса, обратного процессу **Link Refining**, с последующим выполнением процесса **Content Hiding** (Сокрытие контента) — процесса, обратного процессу **Content Showing**.





Процесс **New-Diagram In-Zooming** (Новая детализированная диаграмма) требует объекта **SDn** (SDn-диаграмма).

Процесс **New-Diagram In-Zooming** (Новая детализированная диаграмма) порождает объект **SDn+1** (SDn+1-диаграмма).

Процесс **New-Diagram Out-Zooming** (Новая обобщенная диаграмма) требует объекта **SDn+1** (SDn+1-диаграмма).

Процесс **New-Diagram In-Zooming** (Новая детализированная диаграмма) заменяется на процессы **Content Showing** (Отображение контента) и **Link Refining** (Детализация связей) в указанной последовательности, а также связан с объектом **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма).

Процесс **Content Showing** (Отображение контента) требует объекта **SDn** (SDn-диаграмма).

Процесс **Content Showing** (Отображение контента) создает объект **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма).

Процесс **Link Refining** (Детализация связей) потребляет объект **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма).

Процесс **Link Refining** (Детализация связей) создает объект **SDn+1** (SDn+1-диаграмма).

Процесс **New-Diagram Out-Zooming** (Новая обобщенная диаграмма) заменяется на процесс **Link Abstracting** (Обобщение связей) и **Content Hiding** (Сокрытие контента) в указанной последовательности, а также связан с объектом **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма).

Процесс **Link Abstracting** (Обобщение связей) требует объекта **SDn+1** (SDn+1-диаграмма).

Процесс **Link Abstracting** (Обобщение связей) создает объект **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма).

Процесс **Content Hiding** (Сокрытие контента) потребляет объект **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма).

Процесс **Content Hiding** (Сокрытие контента) создает объект **SDn** (SDn-диаграмма).

Рисунок С.19 — Модели процессов **New-Diagram In-Zooming** и **New-Diagram Out-Zooming**

Объект **Semi-Zoomed OPD** — это промежуточный объект, создаваемый и затем потребяемый в процессе **New Diagram In-Zooming** (Новая детализированная диаграмма) или **New-Diagram Out-Zooming** (Новая обобщенная диаграмма). Объект **Semi-Zoomed OPD** появляется только в контексте процессов **New-Diagram In-Zooming** и **New-Diagram Out-Zooming**.

На рисунке С.20 показаны процессы **New-Diagram In-Zooming** (Новая детализированная диаграмма) и **New-Diagram Out-Zooming** (Новая обобщенная диаграмма) вместе с разворачивающимися объектами **SDn**, **SDn+1** и **Semi-zoomed OPD** (Неполностью преобразованная OPD-диаграмма) с рисунка С.19. Процессы **New-Diagram In-Zooming** и **New-Diagram Out-Zooming** выполняются с использованием конкретного объекта **SDn**, показанного в средней верхней части рисунка С.20, где объект **SDn** — это один из многих других объектов. В данном случае объект **SDn** содержит процесс **P** типа *refineable*, а также четыре объекта, связанных с процессом **P** связями различных видов: объект **C** — связью типа *consumee*, объект **A** — агентской связью, объект **D** — инструментальной связью, объект **B** — связью типа *resultee*.

На детализирующей диаграмме объект **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма) проясняет, что он является промежуточным представлением, создаваемым и потребляемым процессами **New Diagram In-Zooming** (Новая детализированная диаграмма) и **New Diagram Out-Zooming** (Новая обобщенная диаграмма). Объект **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма) является одним и тем же в обоих вариантах.

Процесс **Content Showing** (Отображение контента) является первичным для двух подпроцессов **New-Diagram In-Zooming** (Новая детализированная диаграмма). В процессе **Content Showing** границы процесса **P** охватывают модели подпроцессов **P1**, **P2** и **P3**, а также промежуточный объект **BP**. Результат процесса **Content Showing** состоит в развертывании объекта **Semi-Zoomed OPD**. Как промежуточный объект, определяемый лишь в контексте процесса **New-Diagram In-Zooming**, второй подпроцесс, **Link Refining** (Детализация связей), потребляет его для получения объекта **SDn+1**. В процессе **Link Refining** процедурные связи, приходящие к контуру процесса **P**, разработчиком модели будут перемещаться в сторону соответствующих подпроцессов, поэтому процесс **P1** потребляет объект **C**, а потребительская связь будет перемещаться между процессами **P** и **P1**. Объект агент **A** работает с процессами **P1** и **P2**, поэтому у объекта **SDn+1** появляются две агентские связи: одна — с процессом **P1**, а другая — с процессом **P2**, заменяя связи объекта **SDn** — с объекта **A** на процесс **P**. Процесс **P3** требует объекта **D**, поэтому инструментальная связь будет перемещаться между процессами **P** и **P3**. Наконец, поскольку объект **BP** появляется в результате выполнения процесса **P1**, а процесс **P3** потребляет этот объект, то будут добавляться соответствующие воздействующие потребительские связи, делая объект **BP** для процесса **P** внутренним объектом, который будет проявляться только в контексте процесса **P** — в виде подпроцессов **P1**, **P2** и **P3**. Отметим, что объект **BP** относится к процессу **P**, поскольку объект **Semi-Zoomed OPD** (Неполностью преобразованная OPD-диаграмма) относится к процессу **New-Diagram In-Zooming** (Новая детализированная диаграмма).

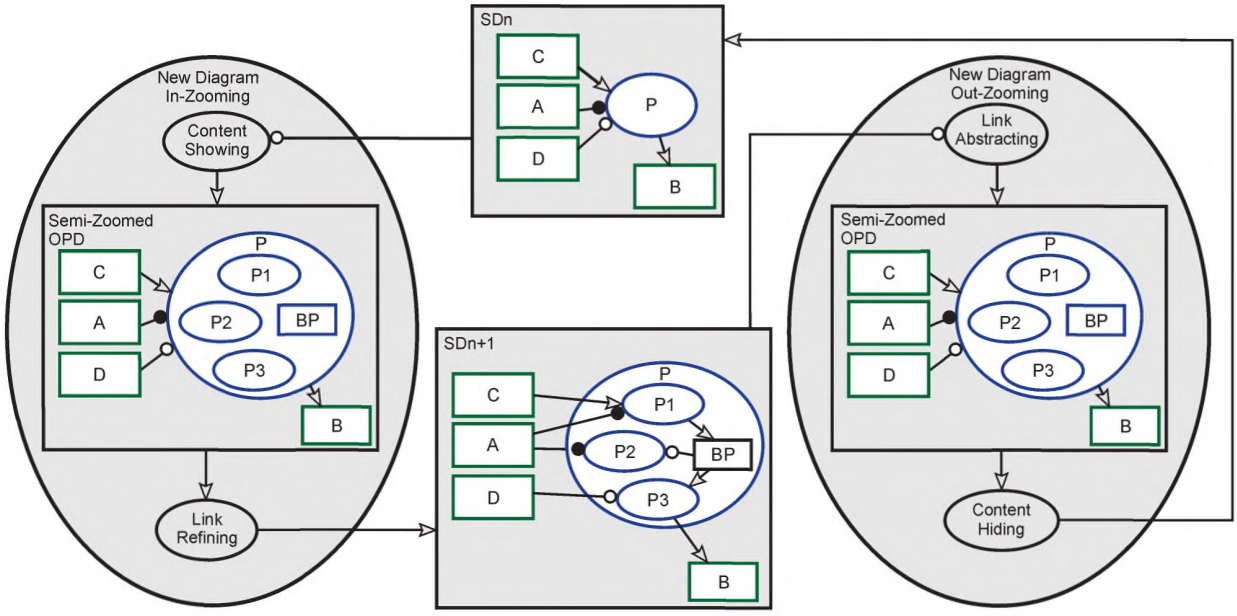


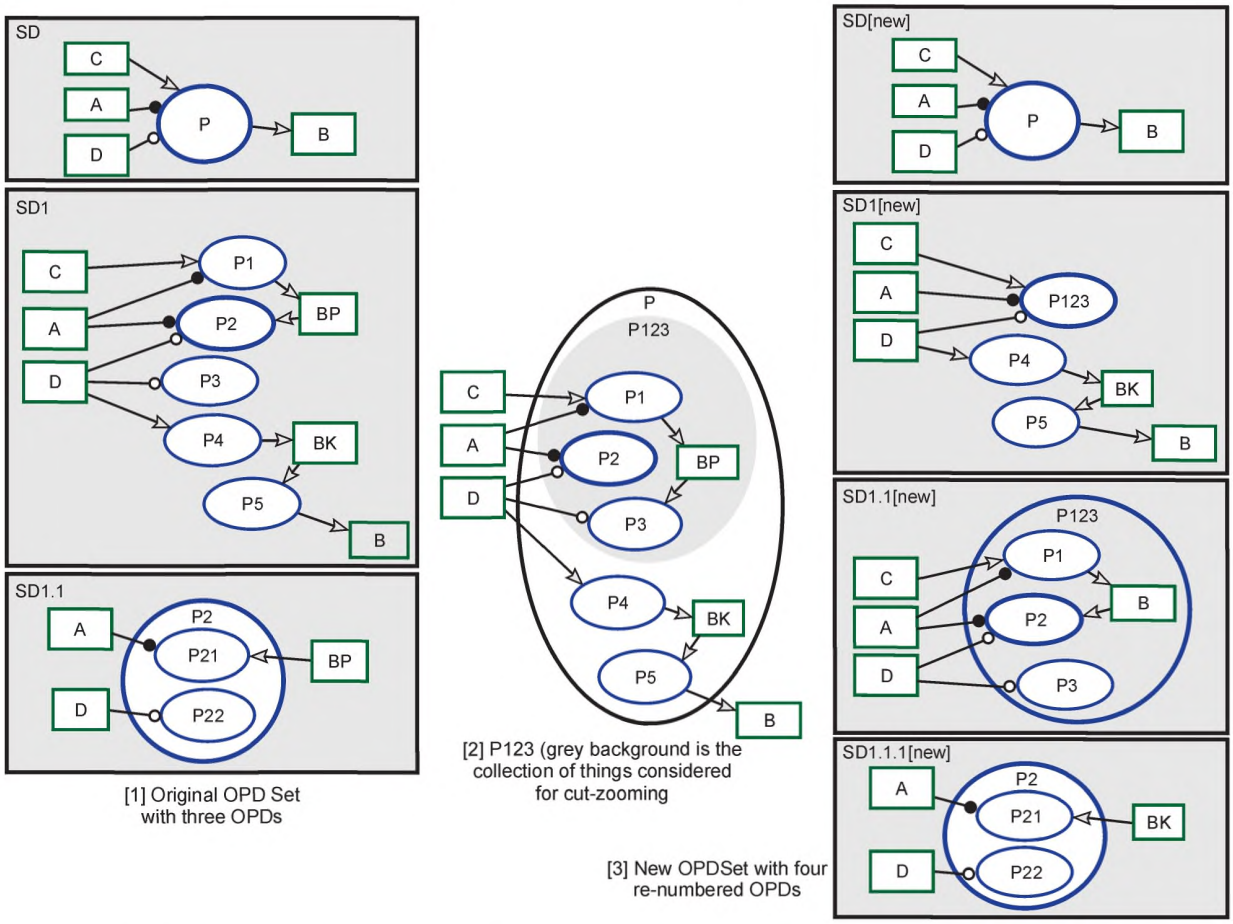
Рисунок С.20 — Описание процессов **New-Diagram In-Zooming** и **New-Diagram Out-Zooming**

С.5.2 Упрощение OPD-диаграммы

Обобщенную диаграмму (In-diagram out-zooming) можно сочетать с новой детализированной диаграммой (new-diagram in-zooming) с целью упрощения уже существующей смоделированной OPD-диаграммы, которую разработчик модели считает чрезмерно сложной. Обобщенная диаграмма, за которой последует новая аналогичная диаграмма, является вариантом, с помощью которого разработчик модели сможет понять, что существующая OPD-диаграмма перегружена различными деталями. Детализированная диаграмма снижает познавательную нагрузку, необходимую для понимания сложной OPD-диаграммы за счет введения новой OPD-диаграммы в набор OPD-диаграмм, появившихся в результате последовательного построения новых обобщенных диаграмм.

На рисунке С.21 приведена детализированная диаграмма, за которой следует новая аналогичная диаграмма. Слева представлен исходный набор из трех OPD-диаграмм: **SD**, **SD1** и **SD1.1**. При этом разработчик модели может посчитать диаграмму **SD1** слишком перегруженной деталями, для снижения детальности которой он выбирает подпроцессы **P1**, **P2** и **P3** вместе с объектом **BP** для их замены на процесс **P123**, используя для этого новую детализированную диаграмму (см. диаграмму посередине рисунка). На правой части этого рисунка изображен новый набор OPD-диаграмм, перенумерованных с учетом новой иерархии. Новая диаграмма **SD1** менее сложная, чем исходная, и имеет пять меньших элементов (три процесса, один объект и две связи удалены и один процесс (**P123**) — добавлен).

Процесс **P123** подвергается новой детализации на новой диаграмме **SD1.1**, и эта новая OPD-диаграмма иерархически вводится в этот процесс, переводя прежнюю диаграмму **SD1.1** в новую диаграмму **SD1.1.1**.



[1] Original OPD Set with three OPDs — [1] Исходный набор OPD-диаграмм, состоящий из трех OPD-диаграмм; [2] P123 (grey background) is the collection of things considered for cut-zooming — [2] Процесс P123 (выделен серым цветом) является совокупностью сущностей, рассматриваемых при детализации; [3] New OPD Set with four re-numbered OPDs — [3] Новый набор OPD-диаграмм, состоящий из четырех перенумерованных OPD-диаграмм.

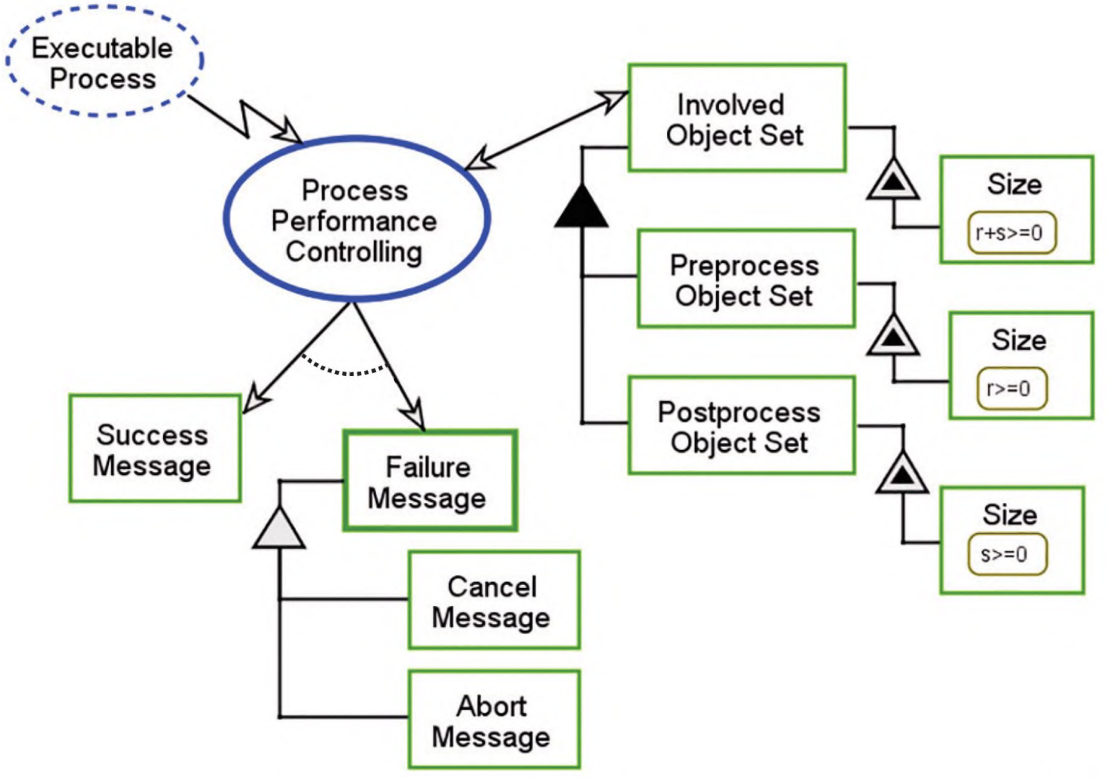
Рисунок С.21 — Упрощенная OPD-диаграмма

Обобщение существующей диаграммы начинается с выбора ТО-набора сущностей для обобщения существующей сложной OPD-диаграммы с целью ее детализации на новой OPD-диаграмме. Предполагая новый одиночный РА-процесс, заменяющий ТО-набор, каждая процедурная связь, доходящая до элемента ТО, должна соединяться с новым РА-процессом и с объектом, который не является ТО-набором. При этом РА-процесс является новым абстрактным процессом, который заменяет элементы ТО-набора, становится новым модельным элементом и обобщением в новой OPD-диаграмме, причем маркировка набора OPD-диаграмм необходима для учета новой OPD-иерархии.

На средней части рисунка С.21 подпроцессы **P1**, **P2** и **P3** вместе с объектом **BP** являются четырьмя элементами ТО, которые содержатся в процессе **P123**. Следствием создания процесса **P123** является исчезновение четырех элементов ТО из новой диаграммы **SD1**. Каждая связь, которая пересекает бело-серую границу среднего графика, теперь будет доходить до границы процесса **P123** на новой диаграмме **SD1**. Объекты, соединяющиеся с границей процесса **P123** на новой диаграмме **SD1**, затем соединяются с соответствующими подпроцессами на новой диаграмме **SD1.1**. Объект **BK** не может быть элементом ТО, поскольку если объект **BK** находится в процессе **P123**, его связи будут создавать две процедурные связи, напрямую соединяющие два процесса: **P4** с **P123**, **P123** с **P5**. OPM-методология не определяет семантику этих связей, а модель будет нарушать спецификацию на соединение каждой процедурной связи (за исключением инициализирующих и исключающих по времени связей) между объектом и процессом.

C.6 OPM-модель контроля реализации процесса

C.6.1 OPM-система контроля осуществления процесса — SD-диаграмма



Объект **Involved Object Set** (Набор рассматриваемых объектов) связан с объектами **Preprocess Object Set** (Набор предварительно обработанных объектов) и **Postprocess Object Set** (Набор апостериорно обработанных объектов).

Объект **Preprocess Object Set** (Набор предварительно обработанных объектов) представляется объектом **Size** (Размер).

Объект **Size** (Размер) объекта **Preprocess Object Set** (Набор предварительно обработанных объектов) имеет состояние **r>=0**.

Объект **Postprocess Object Set** (Набор апостериорно обработанных объектов) представляется объектом **Size** (Объект).

Объект **Size** (Размер) объекта **Postprocess Object Set** (Набор апостериорно обработанных объектов) имеет состояние **s>=0**.

Объект **Involved Object Set** (Набор рассматриваемых объектов) представляется объектом **Size** (Размер).

Объект **Size** (Размер) объекта **Involved Object Set** (Набор рассматриваемых объектов) имеет состояние **r+s>=0**.

Процесс **Process Performance Controlling** (Контроль осуществления процесса) взаимодействует с объектом **Involved Object Set** (Набор рассматриваемых объектов).

Процесс **Executable Process** (Выполняемый процесс) является процессом **environmental** (внешней среды).

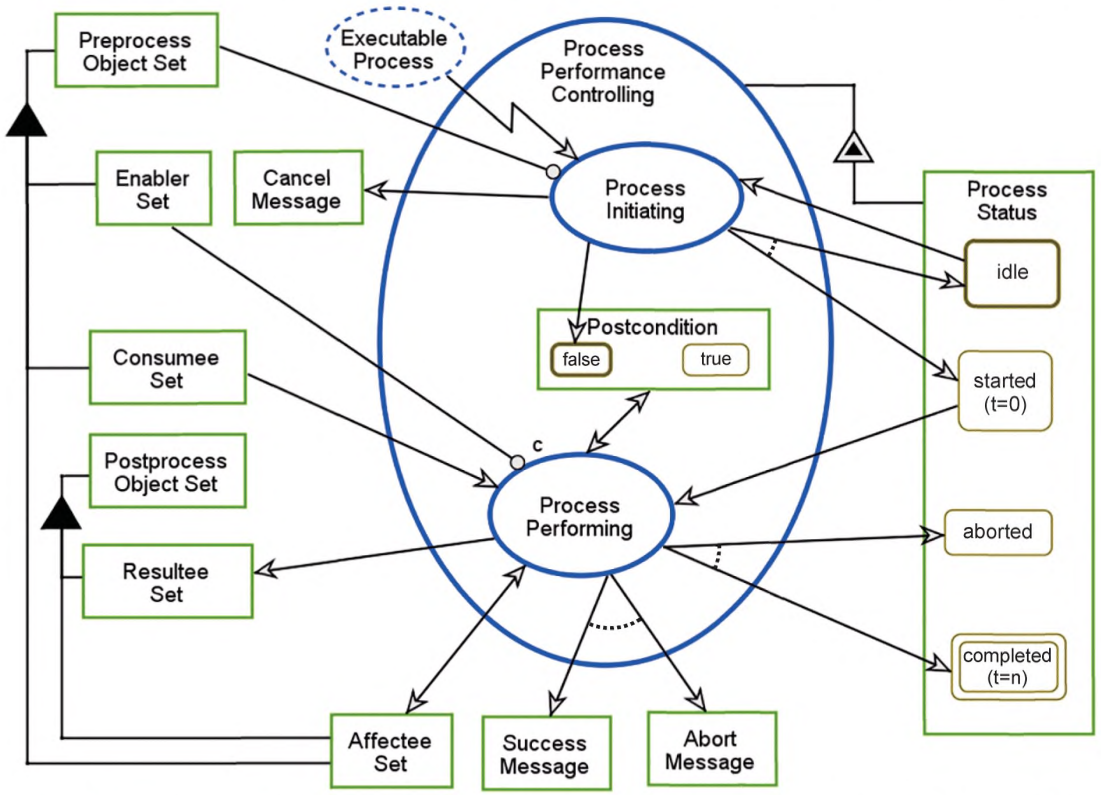
Процесс **Executable Process** (Выполняемый процесс) инициирует процесс **Process Performance Controlling** (Контроль осуществления процесса).

Процесс **Process Performance Controlling** (Контроль осуществления процесса) создает объект **Success Message** (Сообщение об успешном выполнении) или **Failure Message** (Сообщение об отказе).

Объекты **Abort Message** (Сообщение о принудительном прерывании) и **Cancel Message** (Сообщение об отмене) являются объектами **Failure Messages** (Сообщения об отказе).

Рисунок С.22 — Диаграмма системы контроля осуществления процесса — SD-диаграмма

С.6.2 Детализированная диаграмма процесса Process Performance Controlling (диаграмма SD1)



Процесс **Process Performance Controlling** (Процесс осуществления контроля) переходит в процессы **Process Initiating** (Инициализация процесса) и **Process Performing** (Осуществление процесса) в указанной последовательности, а также к объекту **Postcondition** (Постусловие).

Объект **Preprocess Object Set** (Набор предварительно обработанных объектов) содержит объекты **Consumee Set** (Набор объектов типа Consumee), **Affectee Set** (Набор объектов типа Affectee) и **Enabler Set** (Набор средств реализации).

Объект **Postprocess Object Set** (Набор апостериорно обработанных объектов) содержит объекты **Resultee Set** (Набор объектов типа Resultee) и **Affectee Set** (Набор объектов типа Affectee).

Процесс **Executable Process** (Выполняемый процесс) является процессом **environmental** (внешней среды).

Процесс **Executable Process** (Выполняемый процесс) инициирует процесс **Process Initiating** (Инициализация процесса).

Процесс **Process Performance Controlling** представляется объектом **Process Status** (Состояние процесса).

Объект **Process Status** (Состояние процесса) может находиться в состоянии **idle** (ожидание), **started** (начальное) **(t=0)**, **aborted** (прерванное) или **completed** (завершенное) **(t=n)**.

Объект **Process Status** (Состояние процесса) первоначально находится в состоянии **idle** (ожидание), в окончательном состоянии **completed** (завершенный) **(t=n)** или **aborted** (прерванное).

Объект **Postcondition** (Постусловие) может находиться в состоянии **false** (ложное) или **true** (истинное).

Объект **Postcondition** (Постусловие) изначально находится в состоянии **false** (ложное).

Процесс **Process Initiating** (Инициализация процесса) требует объекта **Preprocess Object Set** (Набор предварительно обработанных объектов).

Процесс **Process Initiating** (Инициализация процесса) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **idle** (ожидание) в одно из состояний **idle** (ожидание) или **started** (начальное) **(t=0).**

Процесс **Process Initiating** (Инициализация процесса) переводит в состояние **false** (ложное) объекты **Postcondition** (Постусловие) и **Cancel Message** (Сообщение об отмене).

Процесс **Process Performing** (Выполнение процесса) возникает, если существует объект **Enabler Set** (Набор средств реализации); в противном случае процесс **Process Performing** (Выполнение процесса) пропускается.

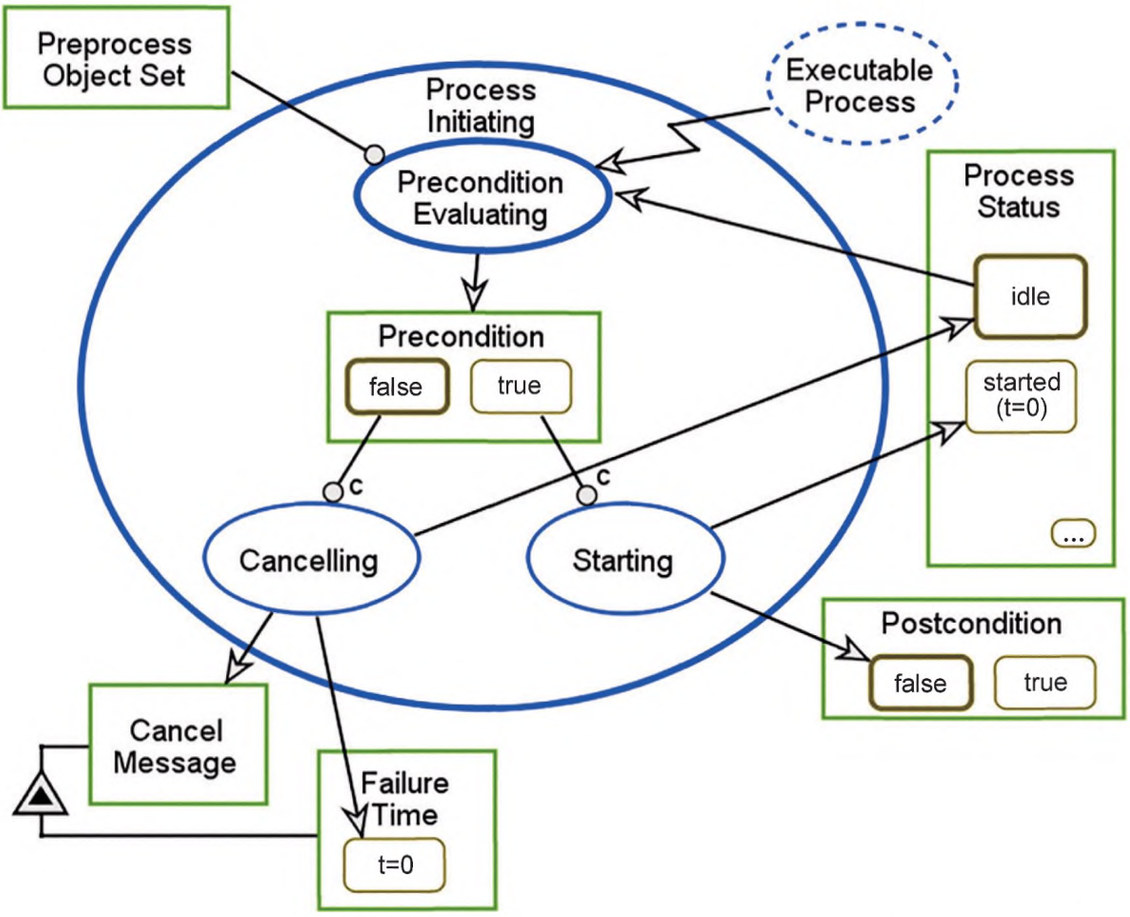
Процесс **Process Performing** (Выполнение процесса) взаимодействует с объектами **Postcondition** (Постусловие) и **Affectee Set** (Набор объектов типа Affectee).

Процесс **Process Performing** (Выполнение процесса) изменяет состояние объекта **Process Status** (Состояние процесса) с состояния **started** (начальное) **(t=0)** на одно из состояний **aborted** (прерванное) или **completed** (завершенное) **(t=n).**

Процесс **Process Performing** (Выполнение процесса) создает объект **Resultee Set** (Набор объектов типа Resultee) и объект **Success Message** (Сообщение об успешном выполнении) или **Abort Message** (Сообщение о принудительном прерывании).

Рисунок С.23 — Детализированная диаграмма процесса **Process Performance Controlling** (SD-диаграмма)

С.6.3 Детализированная диаграмма процесса Process Initiating (диаграмма SD1.1)



Процесс **Process Initiating** (Инициализация процесса), переходящий из диаграммы **SD1** в диаграмму **SD1.1** — в процесс **Precondition Evaluating** (Оценка предварительных условий) и параллельно в процессы **Cancelling** (Отмена) и **Starting** (Начало) в данной последовательности, а также переходит к объекту **Precondition** (Постусловие).

Объект **Process Status** (Состояние процесса) может находиться в состоянии **idle** (ожидание), **started** (начальное) **(t=0)** или в других состояниях.

Объект **Process Status** (Состояние процесса) изначально находится в состоянии **idle** (ожидание).

Объект **Postcondition** (Постусловие) может находиться в состоянии **false** (ложное) или **true** (истинное).

Объект **Postcondition** (Постусловие) изначально находится в состоянии **false** (ложное).

Процесс **Executable Process** (Выполняемый процесс) является процессом **environmental** (внешней среды).

Процесс **Executable Process** (Выполняемый процесс) инициирует процесс **Process Initiating** (Инициализация процесса).

Процесс **Precondition Evaluating** (Оценка предварительных условий) дает объект **Precondition** (Постусловие).

Precondition (Постусловие) может находиться в состоянии **false** (ложное) или **true** (истинное).

Precondition Evaluating (Оценка предварительных условий) требует объекта **Preprocess Object Set** (Набор предварительно обработанных объектов).

Precondition Evaluating (Оценка предварительных условий) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **idle** (ожидание).

Объект **Cancelling** (Отмена) существует, если объект **Precondition** (Постусловие) находится в состоянии **false** (ложное); в противном случае объект **Cancelling** будет пропускаться.

Объект **Cancelling** (Отмена) изменяет состояние объекта **Process Status** (Состояние) на состояние **idle** (ожидание).

Объект **Cancelling** (Отмена) создает объект **Cancel Message** (Сообщение об отмене).

Объект **Cancel Message** (Сообщение об отмене) представляется объектом **Failure Time** (Время отказа).

Объект **Cancelling** (Отмена) устанавливает значение объекта **Failure Time** (Время отказа) на **t=0**.

Объект **Failure Time** (Время отказа), связанный с объектом **Cancel Message** (Сообщение об отмене), равен **t=0**.

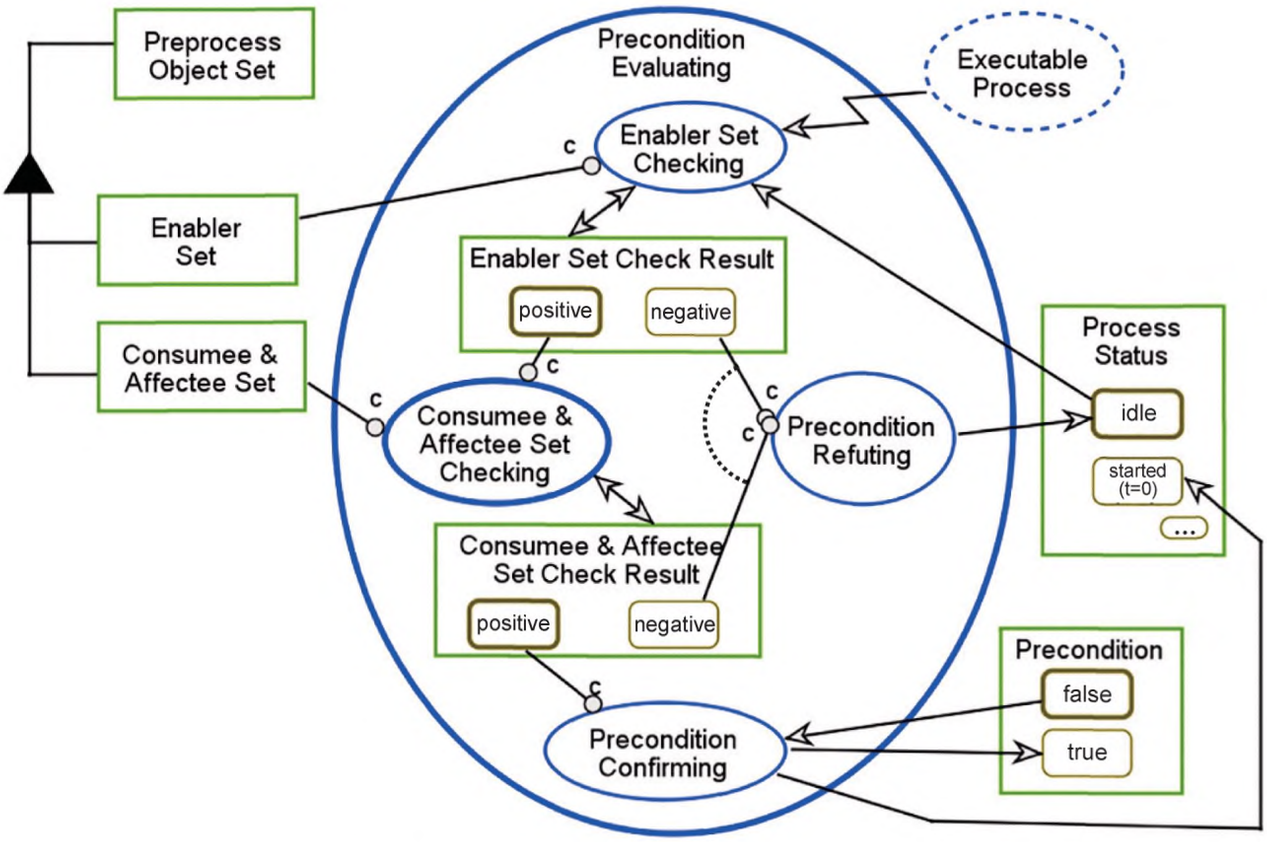
Процесс **Starting** (Начало) возникает, если объект **Precondition** (Предварительные условия) находится в состоянии **true** (истинное); в этом случае объект **Precondition** (Предварительные условия) потребляется; в противном случае объект **Starting** (Начало) будет пропускаться.

Процесс **Starting** (Начало) изменяет состояние процесса **Executable Process** (Выполняемый процесс) на состояние **started** (начальное) **(t=0)**.

Процесс **Starting** (Начало) создает у объекта **Postcondition** (Постусловие) состояние **false** (ложное).

Рисунок С.24 — Детализированная диаграмма процесса **Process Initiating** (SD1.1-диаграмма)

С.6.4 Детализированная диаграмма процесса Precondition Evaluating (диаграмма SD1.1.1)



Процесс **Precondition Evaluating** (Оценка предварительных условий) переходит из диаграммы **SD1.1** в диаграмму **SD1.1.1** — в процессы **Enabler Set Checking** (Проверка набора средств реализации), **Consumee & Affectee Set Checking** (Проверка набора объектов типов Consumee & Affectee), **Precondition Refuting** (Опровержение предварительных условий) и **Precondition Confirming** (Подтверждение предварительных условий) в указанной последовательности, а также переходит к объектам **Enabler Set Check Result** (Результаты проверки набора средств реализации) и **Consumee & Affectee Set Check Result** (Результаты проверки объектов типов Consumee & Affectee).

Объект **Preprocess Object Set** (Набор предварительно обработанных объектов) содержит объекты **Enabler Set** (Набор средств реализации) и **Consumee & Affectee Set** (Набор объектов типов Consumee & Affectee).

Объект **Process Status** (Состояние процесса) может находиться в состоянии **idle** (ожидание), **started** (начальное) **(t=0)** или в других состояниях.

Объект **Process Status** (Состояние процесса) изначально находится в состоянии **idle** (ожидание).

Объект **Precondition** (Предварительные условия) может находиться в состоянии **false** (ложное) или **true** (истинное).

Объект **Precondition** (Предварительные условия) изначально находится в состоянии **false** (ложное).

Процесс **Executable Process** (Выполняемый процесс) инициирует процесс **Enabler Set Checking** (Проверка набора средств реализации).

Процесс **Enabler Set Checking** (Проверка набора средств реализации) требует существования **Enabler Set** (Набор средств реализации); в противном случае процесс **Enabler Set Checking** (Проверка набора средств реализации) будет пропускаться.

Процесс **Enabler Set Checking** (Проверка набора средств реализации) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **idle** (ожидание).

Процесс **Enabler Set Check Result** (Результаты проверки набора средств реализации) может иметь состояние **positive** (положительное) или **negative** (отрицательное).

Процесс **Enabler Set Check Result** (Результаты проверки набора средств реализации) изначально находится в состоянии **positive** (положительное).

Процесс **Enabler Set Checking** (Проверка набора средств реализации) взаимодействует с процессом **Enabler Set Check Result** (Результаты проверки набора средств реализации).

Процесс **Consumee & Affectee Set Checking** (Проверка набора объектов типов Consumee & Affectee) возникает, если процесс **Enabler Set Check Result** (Результаты проверки набора средств реализации) находится в состоянии **positive** (положительное), а объект **Consumee & Affectee Set** (Набор объектов типов Consumee & Affectee) существует; в противном случае процесс **Consumee & Affectee Set Checking** (Проверка набора объектов типов Consumee & Affectee) будет пропускаться.

Процесс **Consumee & Affectee Set Check Result** (Результаты проверки набора объектов типов Consumee & Affectee) может иметь состояние **positive** (положительное) или **negative** (отрицательное).

Рисунок С.25, лист 1 — Детализированная диаграмма процесса **Precondition Evaluating** (SD1.1.1-диаграмма)

Процесс **Consumee & Affectee Set Check Result** (Результаты проверки объектов типов Consumee & Affectee) изначально находится в состоянии **positive** (положительное).

Процесс **Consumee & Affectee Set Checking** (Проверка набора объектов типов Consumee & Affectee) взаимодействует с процессом **Consumee & Affectee Set Check Result** (Результаты проверки набора объектов типов Consumee & Affectee).

Процесс **Precondition Refuting** (Опровержение предварительных условий) требует либо состояния **negative** (отрицательное) процесса **Enabler Set Check Result** (Результаты проверки набора средств реализации), либо состояния **negative** (отрицательное) процесса **Consumee & Affectee Check Result** (Проверка набора объектов типов Consumee & Affectee); в противном случае процесс **Precondition Refuting Precondition Refuting** (Опровержение предварительных условий) будет пропускаться.

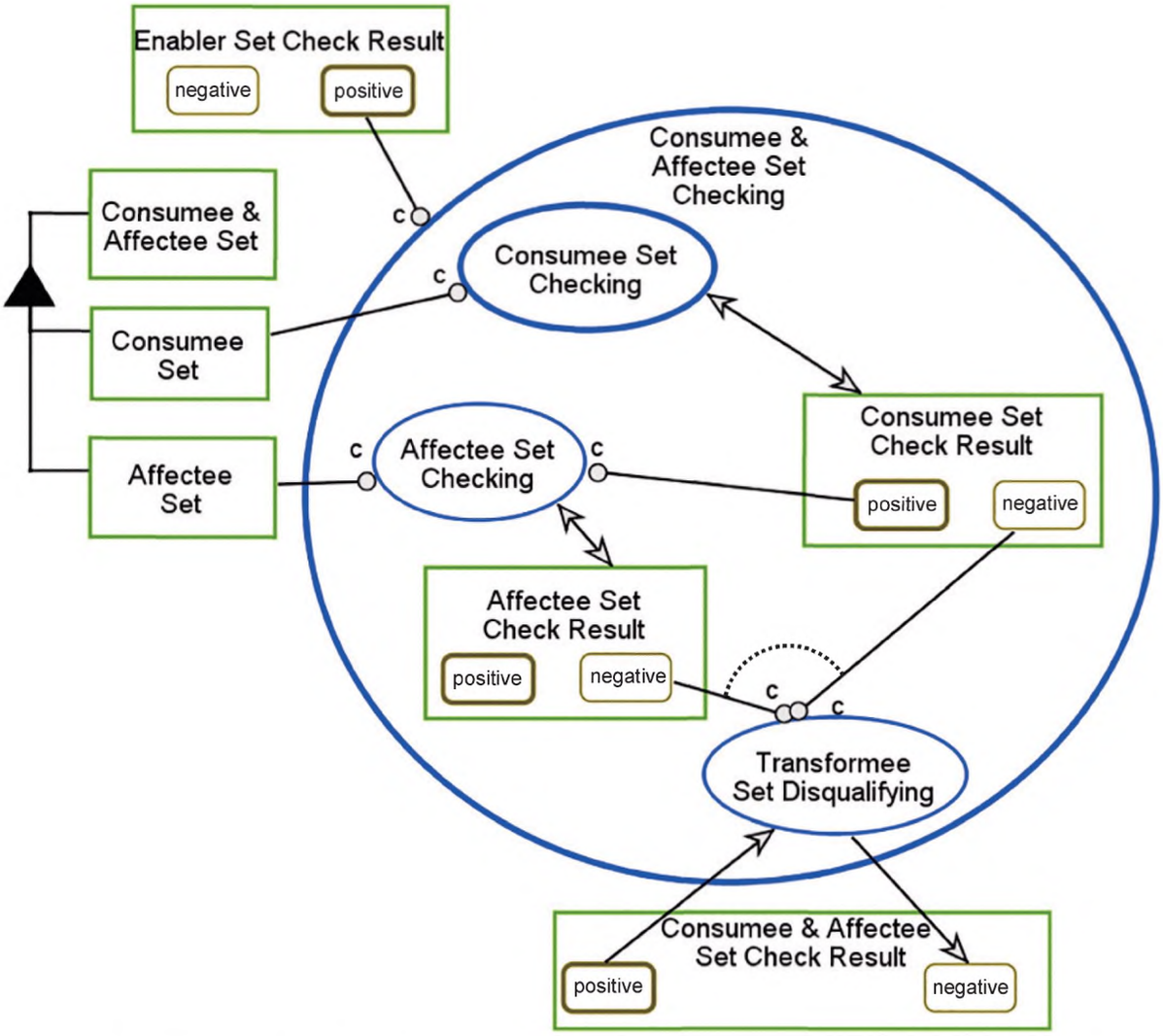
Процесс **Precondition Refuting** (Опровержение предварительных условий) изменяет состояние объекта **Process Status** (Состояние процесса) на состояние **idle** (ожидание).

Процесс **Precondition Confirming** (Подтверждение предварительных условий) возникает, если процесс **Transformee Check Result** (Результаты проверки объектов типа Transformee) находится в состоянии **positive** (положительное); в противном случае процесс **Precondition Confirming** (Подтверждение предварительных условий) будет пропускаться.

Процесс **Precondition Confirming** (Подтверждение предварительных условий) изменяет состояние объекта **Postcondition** (Постусловие) из состояния **false** (ложное) на состояние **true** (истинное), а состояние объекта **Process Status** (Состояние процесса) на состояние **started** (начальное) **(t=0)**.

Рисунок С.25, лист 2

С.6.5 Детализированная диаграмма процесса Consumee & Affectee Set Checking (диаграмма SD1.1.1.1)



Процесс **Consumee & Affectee Set Checking** (Проверка набора объектов типов Consumee & Affectee), переходящий из диаграммы **SD1.1.1** в диаграмму **SD1.1.1.1** — в процессы **Consumee Set Checking** (Проверка набора объектов типа Consumee), **Affectee Set Checking** (Проверка набора объектов типа Affectee) и **Transformee Set Disqualifying** (Исключение набора объектов типа Transformee) в указанной последовательности, а также в объекты **Affectee Set Check Results** (Результаты проверки набора объектов типа Affectee) и **Consumee Set Check Results** (Результаты проверки набора объектов типа Consumee).

Рисунок С.26, лист 1 — Детализированная диаграмма процесса **Consumee & Affectee Set Checking** (SD1.1.1.1)

Объект **Enabler Set Check Result** (Результат проверки набора средств реализации) может находиться в состоянии **negative** (отрицательное) или **positive** (положительное).

Объект **Enabler Set Check Result** (Результат проверки набора средств реализации) изначально находится в состоянии **positive** (положительное).

Процесс **Consumee & Affectee Set Checking** (Проверка набора объектов типов Consumee & Affectee) может находиться в состоянии **negative** (отрицательное) или **positive** (положительное).

Объект **Consumee & Affectee Set Check Result** (Результат проверки набора объектов типов Consumee & Affectee) изначально находится в состоянии **positive** (положительное).

Объект **Consumee & Affectee Set** (Набор объектов типов Consumee & Affectee) связан с объектами **Consumee Set** (Набор объектов типа Consumee) и **Affectee Set** (Набор объектов типа Affectee).

Процесс **Consumee & Affectee Set Checking** (Проверка набора объектов типов Consumee & Affectee) появляется, если объект **Enabler Set Check Result** (Результат проверки набора средств реализации) находится в состоянии **positive** (положительное); в противном случае процесс **Consumee Set Checking** (Проверка набора объектов типа Consumee) будет пропускаться.

Объект **Consumee Set Check Results** (Результаты проверки набора объектов Consumee) может находиться в состоянии **negative** (отрицательное) или **positive** (положительное).

Объект **Consumee Set Check Results** (Результаты проверки набора объектов Consumee) изначально находится в состоянии **positive** (положительное).

Процесс **Consumee Set Checking** (Проверка набора объектов Consumee) возникает, если существует объект **Consumee Set** (Набор объектов типа Consumee); в противном случае процесс **Consumee Set Checking** (Проверка набора объектов типа Consumee) будет пропускаться.

Процесс **Consumee Set Checking** (Проверка набора объектов типа Consumee) взаимодействует с объектом **Consumee Set Check Results** (Результаты проверки набора объектов типа Consumee).

Процесс **Affectee Set Checking** (Проверка набора элементов типа Affectee) возникает, если объект **Consumee Set Check Results** (Результаты проверки набора объектов типа Consumee) находится в состоянии **positive** (положительное), а объект **Affectee Set** (Набор элементов типа Affectee); в противном случае процесс **Affectee Set Checking** (Проверка набора элементов типа Affectee) будет пропускаться.

Процесс **Affectee Set Checking** (Проверка набора элементов типа Affectee) создает объект **Affectee Set Check Results** (Результаты проверки набора объектов типа Affectee).

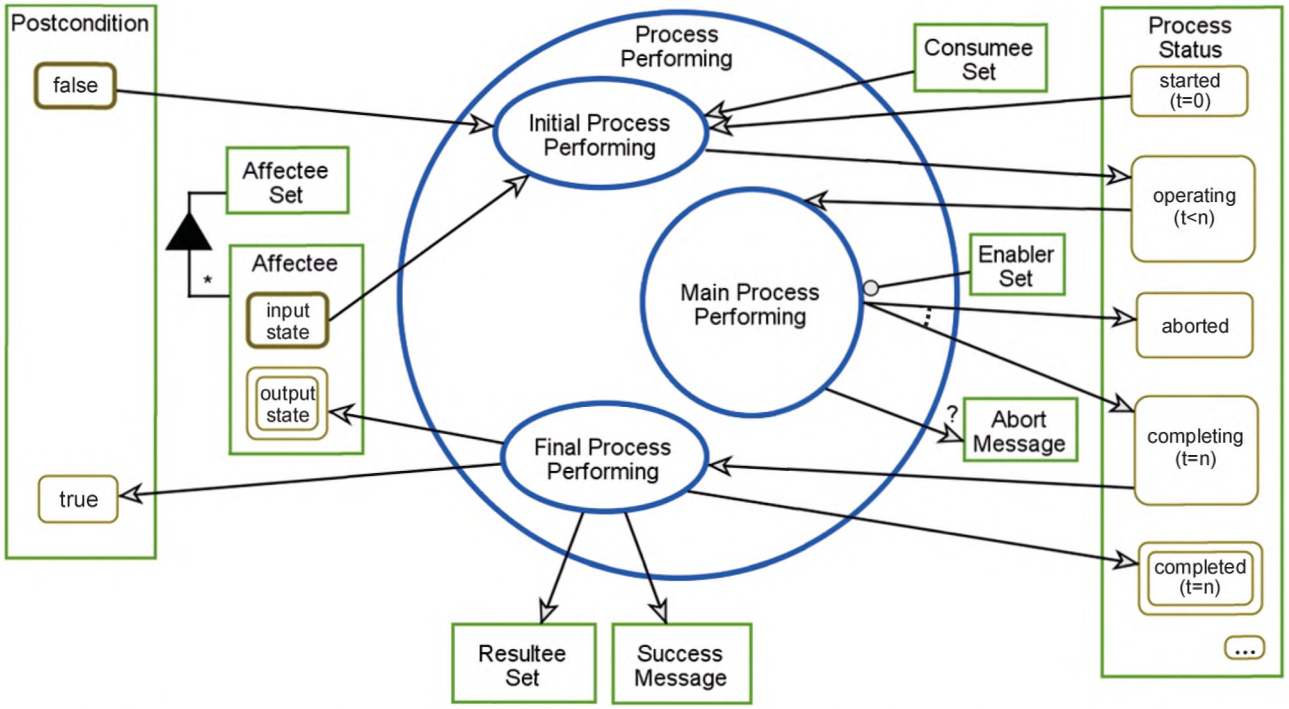
Объект **Affectee Set Check Results** (Результаты проверки набора объектов типа Affectee) может находиться в состоянии **negative** (отрицательное) или **positive** (положительное).

Процесс **Transformee Set Disqualifying** (Исключение набора объектов типа Transformee) появляется, если либо объект **Affectee Set Check Results** (Результаты проверки набора объектов типа Affectee) находится в состоянии **negative** (отрицательное), либо объект **Consumee Set Check Results** (Результаты проверки набора объектов типа Consumee) находится в состоянии **negative** (отрицательное).

Процесс **Transformee Set Disqualifying** (Исключение набора объектов типа Transformee) изменяет состояние объекта **Consumee & Affectee Set Check Result** (Результат проверки набора объектов типов Consumee & Affectee) с состояния **positive** (положительное) на состояние **negative** (отрицательное).

Рисунок С.26, лист 2

С.6.6 Детализированная диаграмма процесса Process Performing (диаграмма SD1.2)



Процесс **Process Performing** (Осуществление процесса), переходящий из диаграммы **SD1** в диаграмму **SD1.2** — в процессы **Initial Process Performing** (Выполнение начального процесса), **Main Process Performing** (Выполнение основного процесса) и **Final Process Performing** (Выполнение конечного процесса) в указанной последовательности.

Объект **Process Status** (Состояние процесса) может находиться в состоянии **idle** (ожидание), **started** (начальное) **(t=0), operating** (рабочее) **(t<n), aborted** (прерванное), **completing** (заканчивающееся) **(t=n), completed** (законченное) **(t=n)** или в других состояниях.

Объект **Process Status** (Состояние процесса) в окончательном состоянии является **completed** (законченным) **(t=n)**.

Объект **Postcondition** (Постусловие) может находиться в состоянии **false** (ложное) или **true** (истинное).

Объект **Postcondition** (Постусловие) изначально находится в состоянии **false** (ложное).

Объект **Affectee Set** (Набор объектов типа Affectee) связан с дополнительными объектами типа **Affectees**.

Объект типа **Affectee** может находиться в состоянии **input state** (входное состояние) или **output state** (выходное состояние).

Объект типа **Affectee** изначально находится в состоянии **input state** (входное состояние) и окончательно в состоянии **output state** (выходное состояние).

Процесс **Initial Process Performing** (Выполнение начального процесса) изменяет состояние объекта Process **Status** (Состояние процесса) из состояния **started** (начальное) **(t=0)** на состояние **operating** (рабочее) **(t<n),** состояние объекта **Postcondition** (Постусловие) — из состояния **false** (ложное), и состояние объекта типа **Affectee** — из состояния **input state** (входное состояние).

Процесс **Initial Process Performing** (Выполнение начального процесса) потребляет объект **Consumee Set** (Набор объектов типа Consumee).

Процесс **Main Process Performing** (Выполнение основного процесса) требует объекта **Enabler Set** (Набор средств реализации).

Процесс **Main Process Performing** (Выполнение основного процесса) создает дополнительный объект **Abort Message** (Прерывание сообщения).

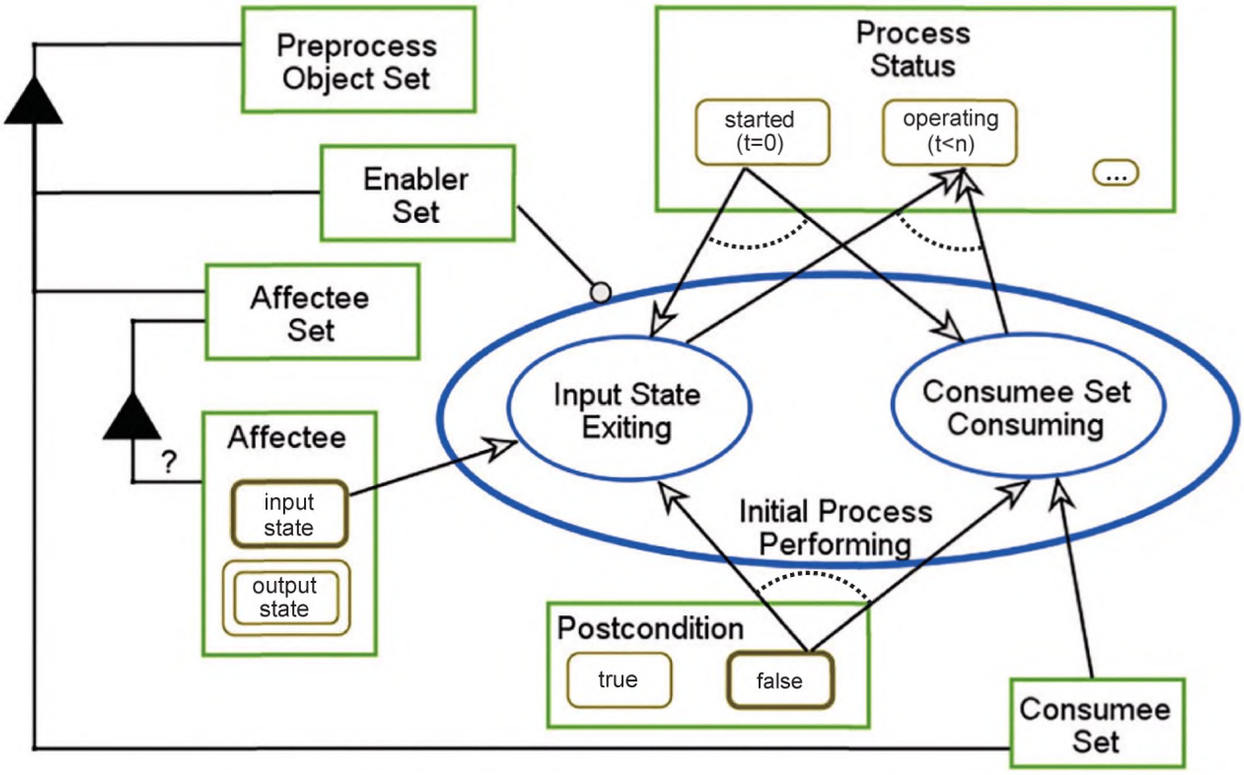
Процесс **Main Process Performing** (Выполнение основного процесса) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **operating** (рабочее) **(t<n)** на состояние **completing** (заканчивающееся) **(t=n)** или **aborted** (прерванное).

Процесс **Final Process Performing** (Выполнение конечного процесса) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **completing** (заканчивающееся) **(t=n)** на состояние **completed** (завершенное) **(t=n)**, состояние объекта **Postcondition** (Постусловие) — на состояние **true** (истинное) и состояние объекта типа **Affectee** — на состояние **output state** (выходное состояние).

Процесс **Final Process Performing** (Выполнение конечного процесса) создает объекты **Success Message** (Сообщение об успешном выполнении) и **Resultee Set** (Набор объектов типа Resultee).

Рисунок С.27 — Детализированная диаграмма процесса **Process Performing** (SD1.2)

С.6.7 Детализированная диаграмма процесса Initial Process Performing (диаграмма SD1.2.1)



Процесс **Initial Process Performing** (Выполнение начального процесса), переходящий из диаграммы **SD1.2** в диаграмму **SD1.2.1** — параллельно в подпроцессы **Input State Exiting** (Существование начального состояния) и **Consumee Set Consuming** (Потребление набора объектов типа Consumee).

Объект **Preprocess Object Set** (Набор предварительно обработанных объектов) связан с объектами **Enabler Set** (Набор средств реализации), **Affectee Set** (Набор объектов типа Affectee) и **Consumee Set** (Набор объектов типа Consumee).

Объект типа **Affectee Set** (Набор объектов типа Affectee) связан с дополнительными объектами типа **Affectees**.

Объект типа **Affectee** может находиться в состоянии **input state** (входное состояние) или **output state** (выходное состояние).

Объект типа **Affectee** изначально находится в состоянии **input state** (входное состояние), и окончательно — в состоянии **output state** (конечное состояние).

Объект **Process Status** (Состояние процесса) может находиться в состоянии **started** (начальное) **(t=0), operating** (рабочее) **(t<0)** или в других состояниях.

Объект **Postcondition** (Постусловие) может находиться в состоянии **false** (ложное) или **true** (истинное).

Объект **Postcondition** (Постусловие) изначально находится в состоянии **false** (ложное).

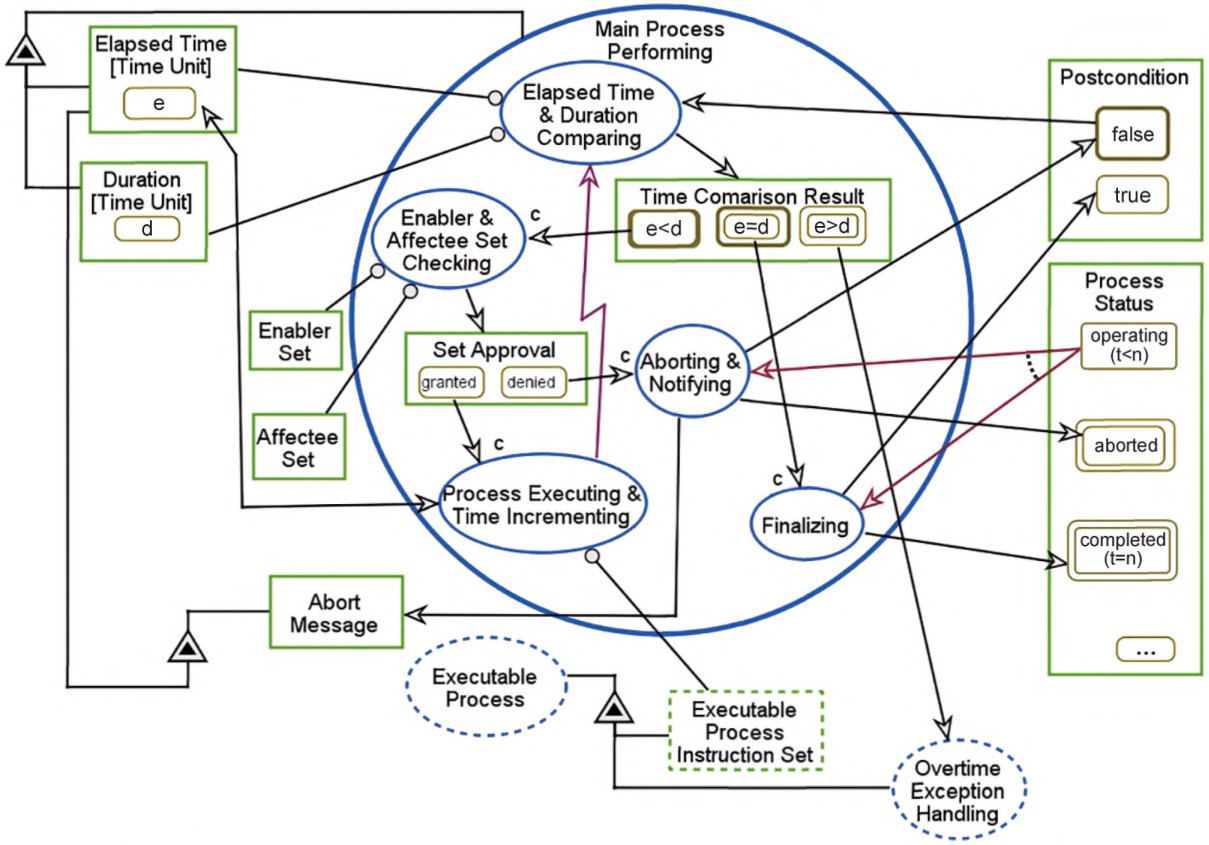
Процесс **Initial Process Performing** (Выполнение начального процесса) требует объекта **Enabler Set** (Набор средств реализации).

Подпроцесс **Input State Exiting** (Существование входного состояния) изменяет состояние объекта **Affectee** из состояния **input state** (входное состояние).

Подпроцесс **Consumee Set Consuming** (Потребление набора объектов типа Consumee) или подпроцесс **Input State Exiting** (Существование входного состояния) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **started** (начальное) **(t=0)** на состояние **operating** (рабочее) **(t<n),** а состояние объекта **Postcondition** (Постусловие) — из состояния **false** (ложное).

Рисунок С.28 — Детализированная диаграмма процесса **Initial Process Performing** (SD1.2.1)

С.6.8 Детализированная диаграмма процесса Main Process Performing (диаграмма SD1.2.2)



Процесс **Main Process Performing** (Выполнение основного процесса), переходящий из диаграммы **SD1.2** в диаграмму **SD1.2.2** — в процессы **Elapsed Time & Duration Comparing** (Сравнение истекшего времени & длительности), **Enabler & Affectee Set Checking** (Проверка набора объектов типов Enabler & Affectee), **Aborting & Notifying** (Прерывание & уведомление), **Time Incrementing** (Временное инкрементирование) и **Finalizing** (Завершение) в указанной последовательности, а также в объекты **Time Comparison Result** (Результат сравнения по времени) и **Set Approval** (Получение одобрения).

Процесс **Executable Process** (Исполняемый процесс) представляется объектами **Executable Process Instruction Set** (Набор инструкций для исполняемого процесса) и **Overtime Exception Handling** (Обработка сверхурочных исключений).

Процесс **Executable Process** (Исполняемый процесс), объекты **Executable Process Instruction Set** (Набор инструкций для исполняемого процесса) и **Overtime Exception Handling** (Обработка сверхурочных исключений) являются процессами **environmental** (внешней среды).

Объект **Process Status** (Состояние процесса) может иметь состояние **aborted** (прерванное), **completed** (завершенное) **(t=n), operating** (рабочее) **(t<0)** или другие состояния.

Объект **Process Status** (Состояние процесса) имеет конечное состояние **aborted** (прерванное) или **completed** (завершенное) **(t=n)**.

Объект **Postcondition** (Постусловие) может иметь состояние **false** (ложное) или **true** (истинное).

Объект **Postcondition** (Постусловие) изначально имеет состояние **false** (ложное).

Процесс **Main Process Performing** (Выполнение основного процесса) представляется объектами **Elapsed Time** (Истекшее время), **Time Unit** (Единицы времени) и **Duration** (Продолжительность), **Time Unit** (Единицы времени).

Процесс **Abortion Message** (Сообщение о принудительном прерывании) представлено объектом **Elapsed Time** (Истекшее время), **Time Unit** (Единицы времени).

Объект **Elapsed Time** (Истекшее время), **Time Unit** (Единицы времени) является объектом типа **e**.

Объект **Duration** (Продолжительность), **Time Unit** (Единицы времени) является объектом типа **d**.

Процесс **Elapsed Time & Duration Comparing** (Сравнение истекшего времени & длительности) требует объектов **Elapsed Time** (Истекшее время), **Time Unit** (Единицы времени) и **Duration** (Продолжительность), **Time Unit** (Единицы времени).

Процесс **Elapsed Time & Duration Comparing** (Сравнение истекшего времени & длительности) изменяет состояние объекта **Postcondition** (Постусловие) из состояния **false** (ложное).

Рисунок С.29, лист 1 — Детализированная диаграмма процесса **Main Process Performing** (SD1.2.2)

Процесс **Elapsed Time & Duration Comparing** (Сравнение истекшего времени & длительности) создает объект **Time Comparison Result** (Результат сравнения по времени).

Объект **Time Comparison Result** (Результат сравнения по времени) может иметь значения **e<d, e=d** или **e>d**.

Объект **Time Comparison Result** (Результат сравнения по времени) изначально находится в состоянии **e<d** или **e=d** и окончательно — в состоянии **finally e=d** или **e>d**.

Процесс **Enabler & Affectee Set Checking** (Проверка набора объектов типов Enabler & Affectee) требует объектов **Enabler Set** (Набор средств реализации) и **Affectee Set** (Набор объектов типа Affectee).

Процесс **Enabler & Affectee Set Checking** (Проверка набора объектов типов Enabler & Affectee) возникает, если объект **Time Comparison Result** (Результат сравнения по времени) составляет **e<d;** в этом случае данный процесс будет потреблять указанный объект; в противном случае процесс **Enabler & Affectee Set Checking** будет пропускаться.

Процесс **Enabler & Affectee Set Checking** (Проверка набора объектов типов Enabler & Affectee) требует объекта **Enabler Set** (Набор средств реализации).

Процесс **Enabler & Affectee Set Checking** (Проверка набора объектов типов Enabler & Affectee) создает объект **Set Approval** (Получение одобрения).

Объект **Set Approval** (Получение одобрения) может иметь состояние **granted** (одобрено) или **denied** (отказано).

Процесс **Aborting & Notifying** (Прерывание & уведомление) протекает, если объект **Set Approval** (Получение одобрения) находится в состоянии **denied** (отказано); в этом случае данный процесс будет потреблять указанный объект; в противном случае процесс **Aborting & Notifying** будет пропускаться.

Процесс **Aborting & Notifying** (Прерывание & уведомление) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **operating** (рабочее) **(t<n)** на состояние **aborted** (прерванное), а состояние объекта **Postcondition** (Постусловие) — на состояние **false** (ложное).

Процесс **Aborting & Notifying** (Прерывание & уведомление) создает объект **Abort Message** (Сообщение о принудительном прерывании).

Процесс **Finalizing** (Завершение) возникает, если объект **Time Comparison Result** (Результат сравнения по времени) есть **e=d**; в этом случае данный процесс будет потреблять указанный объект; в противном случае процесс **Finalizing** будет пропускаться.

Процесс **Finalizing** (Завершение) изменяет состояние объекта **Process Status** (Состояние процесса) из состояния **operating** (рабочее) **(t<n)** на состояние **completed** (завершение) **(t=n)**, а состояние объеста **Postcondition** (Постусловие) — на состояние **true** (истинное).

Процесс **Process Executing & Time Incrementing** (Выполнение процесса & инкрементирование по времени) требует объекта **Executable Process Instruction** (Набор инструкций для исполняемого процесса).

Процесс **Process Executing & Time Incrementing** (Выполнение процесса & инкрементирование по времени) возникает, если объект **Set Approval** (Получение одобрения) находится в состоянии **granted** (одобрено); в этом случае данный процесс будет потреблять указанный объект; в противном случае процесс **Process Executing & Time Incrementing** (Выполнение процесса & инкрементирование по времени) будет пропускаться.

Процесс **Time Incrementing** (Инкрементирование по времени) потребляет объекты **Sets are OK?**

Процесс **Time Incrementing** (Инкрементирование по времени) создает **elt=1..ext** объекта **Elapsed Time** (Истекшее время), **Time Unit** (Единицы времени).

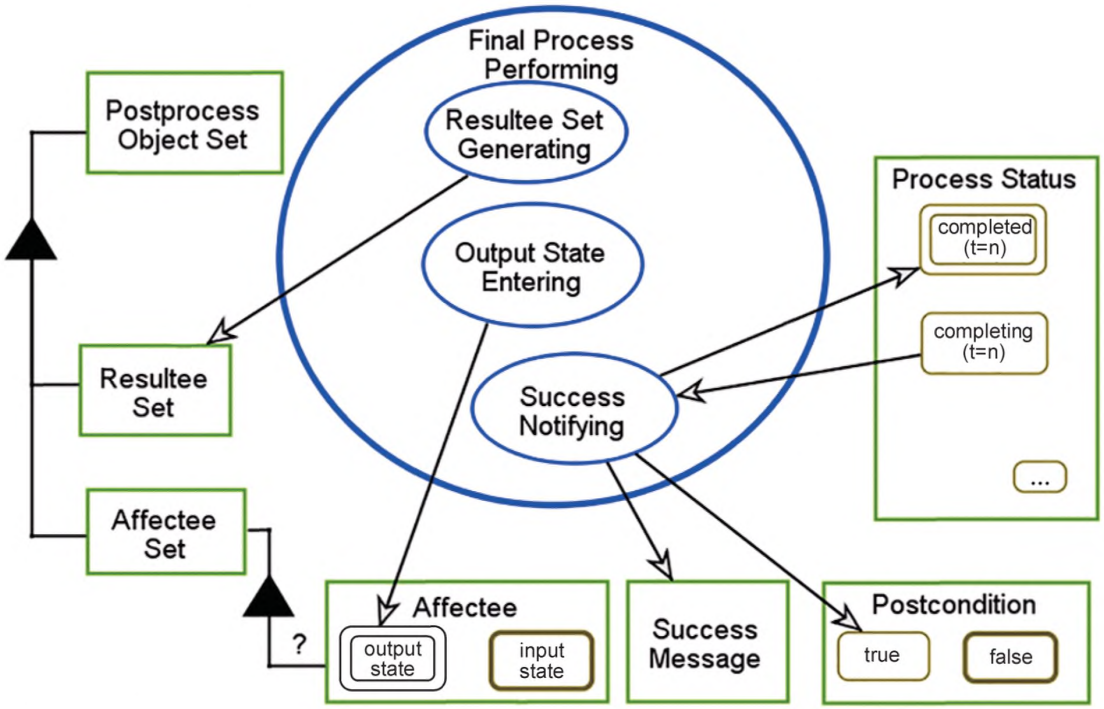
Процесс **Process Executing & Time Incrementing** (Выполнение процесса & инкрементирование по времени) изменяет значение **e** объекта **Elapsed Time** (Истекшее время) **Time Unit** (Единицы времени).

Процесс **Process Executing & Time Incrementing** (Выполнение процесса & инкрементирование по времени) инициирует процесс **Elapsed Time & Duration Comparing** (Сравнение истекшего времени & длительности).

Процесс **Overtime Exception Handling** (Обработка сверхурочных исключений) потребляет **e>d** объекта **Time Comparison Result** (Результат сравнения по времени).

Рисунок С.29, лист 2

С.6.9 Детализированная диаграмма процесса Final Process Performing (диаграмма SD1.2.3)



Процесс **Final Process Performing** (Выполнение конечного процесса), переходящий из диаграммы **SD1.2** в диаграмму **SD1.2.3** — в подпроцессы **Resultee Set Generating** (Формирование набора объектов Resultee), **Output State Entering** (Ввод выходного состояния) и **Success Notifying** (Оповещение об успехе) в указанной последовательности.

Объект **Postprocess Object Set** (Набор апостериорно обработанных объектов) связан с объектами **Resultee Set** (Набор объектов типа Resultee) и **Affectee Set** (Набор объектов типа Affectee).

Объект **Affectee Set** (Набор объектов типа Affectee) связан с дополнительными объектами типа **Affectees**.

Объект **Affectee** может находиться в состоянии **input state** (входное состояние) или **output state** (выходное состояние).

Объект типа **Affectee** изначально находится в состоянии **input state** (входное состояние) и окончательно в состоянии **output state** (выходное состояние).

Объект **Process Status** (Состояние процесса) может находиться в состоянии **completed** (законченное) **(t=n), completing** (завершающееся) **(t=n)** или в других состояниях.

Объект **Process Status** находится в конечном состоянии **completed** (законченное) **(t=n)**.

Объект **Postcondition** (Постусловие) может находиться в состоянии **false** (ложное) или **true** (истинное).

Объект **Postcondition** изначально находится в состоянии **false** (ложное).

Подпроцесс **Resultee Set Generating** (Формирование набора объектов Resultee) создает объект **Resultee Set** (Набор объектов Resultee).

Подпроцесс **Output State Entering** (Ввод выходного состояния) изменяет состояние объекта типа **Affectee** из состояния **output state** (выходное состояние).

Подпроцесс **Success Notifying** (Оповещение об успехе) изменяет состояние объекта **Postcondition** (Постусловие) на состояние **true** (истинное).

Подпроцесс **Success Notifying** (Оповещение об успехе) создает объект **Success Message** (Сообщение об успешном выполнении).

Рисунок С.30 — Детализированная диаграмма процесса **Final Process Performing** (SD1.2.3)

Приложение D
(справочное)

Динамические свойства и моделирование в OPM-методологии

D.1 Реализуемость OPM-методологии

OPM-модель обеспечивает ее реализуемость, т. е. способность моделировать систему путем реализации этой модели посредством ее динамического воспроизведения в надлежащим образом спроектированной программной среде.

D.2 Изменения и воздействия в OPM-методологии

Изменение объекта — это изменение состояния этого объекта. Более конкретно, изменение объекта отражается в замене его текущего состояния на другое состояние. Единственное, что может вызывать это изменение — это процесс, который вызывает его, принимая на входе данный объект в определенном состоянии и получая на выходе объект с другим состоянием. Таким образом, изменение объекта означает изменение его состояния, в котором объект находился.

Структурированные объекты (объекты с внутренними состояниями) могут подвергаться воздействиям, т. е. их состояния могут изменяться. Этот механизм изменения характеризует тесную, неразрывную связь между объектами и процессами. Изменение состояния является следствием воздействия процесса на объект.

По этой причине воздействие может быть определено как изменение состояния объекта, вызванное процессом.

Хотя термины «изменение» и «воздействие» являются почти синонимами, все же существует достаточно тонкое различие в их использовании. Термин «воздействие» (effect) используется для обозначения того, что процесс делает с объектом, а термин «изменение» (change) — того, что происходит с объектом в результате возникновения процесса. Приведенное выше определение воздействия уточняется далее с обозначением входных и выходных связей.

D.3 Существование и преобразования в OPM-методологии

Изменение является только одной из возможностей того, что может произойти с объектом при воздействии на него какого-либо процесса, который влияет на объект для его изменения, однако он может оказывать и более радикальное воздействие, например, формировать или потреблять объекты. Термин «преобразование» (transformation) охватывает три дополнительных вида воздействия на объект: формирование (construction), воздействие (effect) и потребление (consumption).

Термин «формирование» является синонимом для терминов «создание», «образование» или «получение». Термин «воздействие» является синонимом для терминов «изменение» или «переключение», а термин «потребление» является синонимом для терминов «ликвидация», «прекращение», «уничтожение» или «разрушение». При воздействии процесса на объект для изменения его состояния на другое состояние объект будет продолжать существовать и сохранять свою идентичность, которую он имел до возникновения процесса. Формирование и потребление изменяют сам факт существования объекта и, следовательно, вызывают более глубокие преобразования, чем воздействие.

Когда процесс формирует объект (образует, создает или получает его), это будет означать появление нового объекта, который не существовал ранее, поскольку он претерпел радикальные изменения. Это преобразование сделало объект выделенным, идентифицируемым и значимым в системе. После этого объект потребует обработки и обозначения как новой, самостоятельной сущности.

Когда процесс потребляет какой-либо объект (уничтожает или удаляет его), это будет означать, что объект, который ранее существовал, был идентифицируемым и значимым в системе, претерпел радикальные изменения, т. е. он перестал существовать и идентифицироваться в системе.

D.4 Принцип временного OPM-планирования

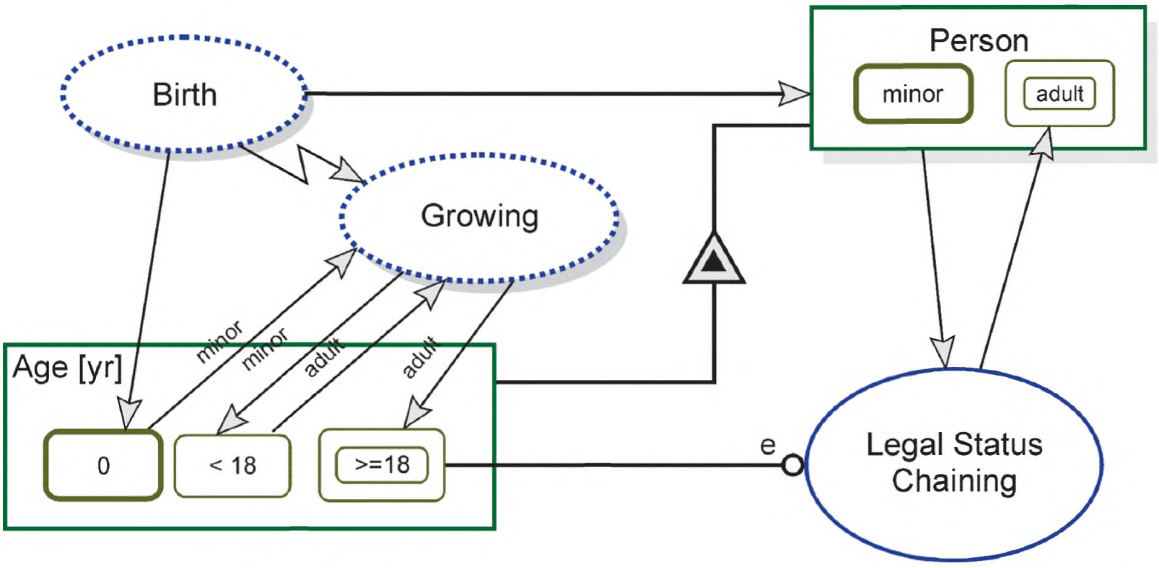
По умолчанию планирование по времени выполнения в рамках детализированного процесса начинается с верхней части диаграммы и заканчивается в нижней части той же диаграммы (при отсутствии признаков относительно отклонений от временного плана-графика). Такими признаками могут быть особые внутренние OPM-события в рамках процесса, которые могут вызвать различные контуры, а также процесс, чьим именем является или имя оканчивается на **Exception Exiting**. Вне зависимости от его графического положения, если этот процесс будет инициализирован, то контекст будет обрабатываться, т. е. детализированный процесс, который встроен в данный процесс, будет выходить своевременно и безусловно.

Самая верхняя точка эллипса процесса служит в качестве опорной точки, поэтому процесс, точка отсчета которого выше своего однотипного процесса (процессов), будет начинаться раньше. Если опорные точки для двух или более процессов находятся на одной и той же высоте (в пределах нескольких графических блоков, например, пикселей — в пределах допусков), то эти процессы будут начинаться одновременно и выполняться параллельно.

D.5 Приуроченные по времени события

Событиями, представляемыми до сих пор, были объекты или состояния: события происходили, когда конкретный объект начинал существовать или входить в определенное состояние. В противоположность этому приуроченные по времени события будут зависеть от наступления определенного момента времени в системе (см. ниже).

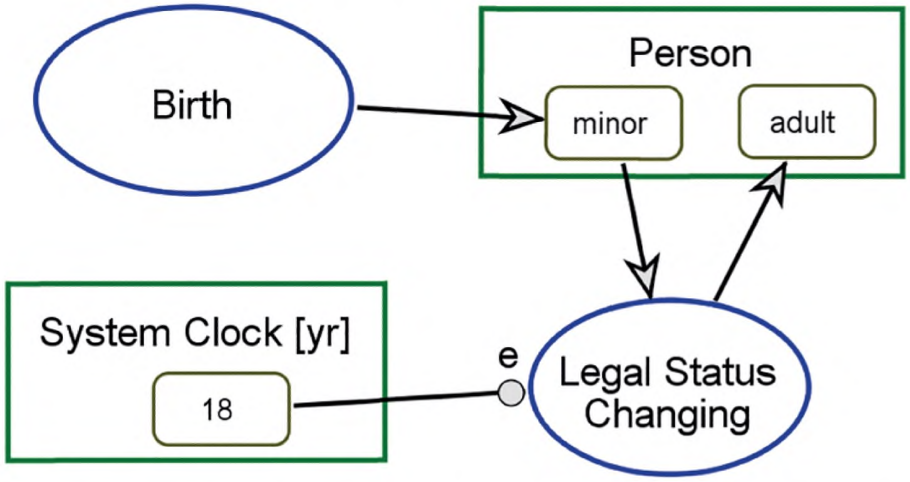
Состояние-событие может представлять собой событие-время (см. рисунок D.1).



Birth — рождение; Growing — рост; Age [yr] — возраст (лет); minor — малолетнее; adult — взрослое; Person — физическое лицо; Legal Status Changing — изменение юридического статуса

Рисунок D.1 — Юридическая модель системы, изменяющаяся от малолетства до взрослого состояния (до возраста 18 лет)

На рисунке D.2 показано системное временное событие, инициирующее изменение юридического статуса.



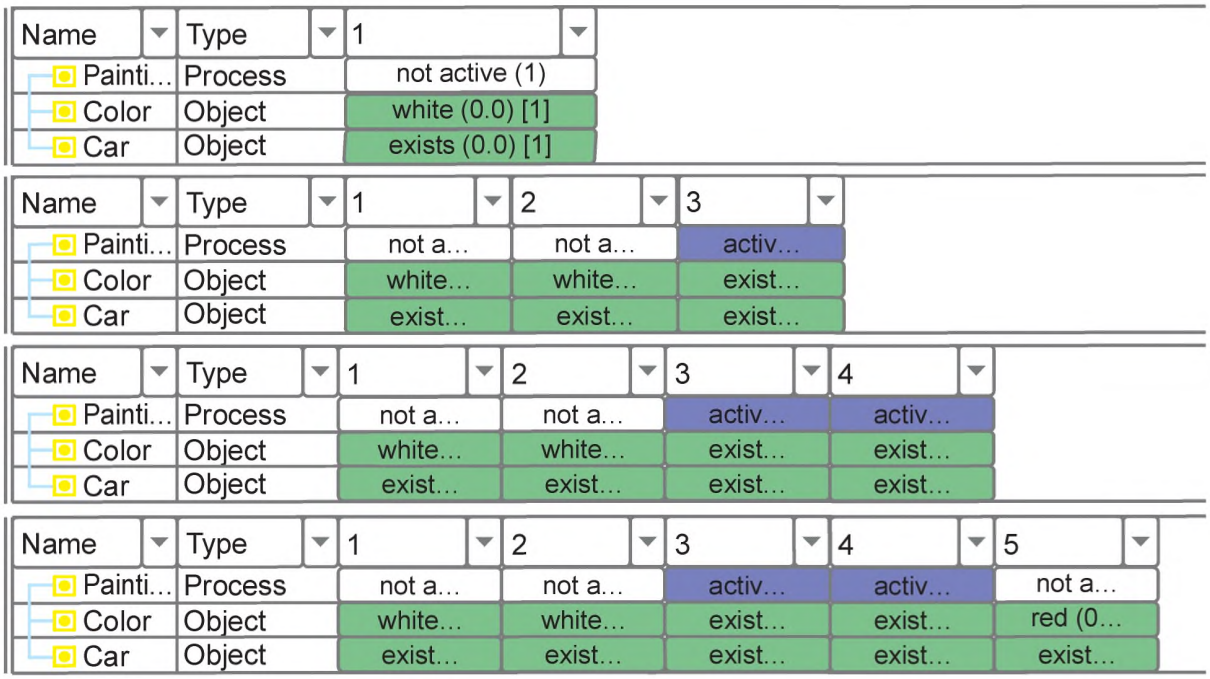
Birth — рождение; minor — малолетнее; adult — взрослое; Person — физическое лицо; Legal Status Changing — изменение юридического статуса; Sysnem Clock [yr] — системное время

Рисунок D.2 — Системное временное событие, инициализирующее изменение юридического статуса

D.6 Предыстория объекта и диаграмма протекания процесса

В любой момент времени объект может находиться в одном из своих состояний или может происходить переход между двумя состояниями.

Диаграмма выполнения процесса (lifespan diagram) — это диаграмма, показывающая в любой момент времени на протяжении всего срока службы системы вид объектов, существующих в системе; состояние каждого объекта и активные в настоящее время процессы.

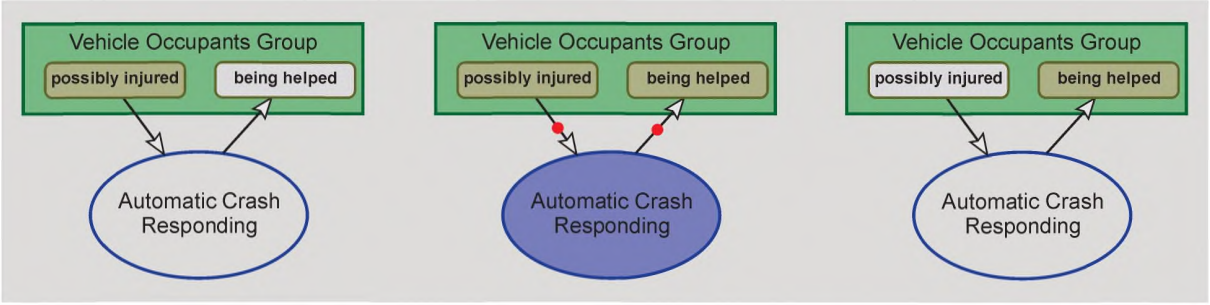


Name — имя; Type — тип; Painting — окраска; Process — процесс; Color — цвет; Object — объект; Car — автомобиль; not active — не активный; active — активный; exist — существует; white — белый; red — красный

Рисунок D.3 — Пример четырех диаграмм выполнения процесса окраски автомобилей

Эти четыре диаграммы выполнения процесса окраски автомобилей (см. рисунок D.3) отображают предысторию системы окраски в виде временных процессов в виде вертикальных столбцов, облегчающих их просмотр. На первой диаграмме отображается только первый период времени. Процесс окраски не активен; автомобиль окрашивается в белый цвет.

На второй диаграмме отображаются первые три периода времени. В третьем периоде процесс окраски уже не является активным, и цвет автомобиля уже не белый. То же самое происходит и в четвертом периоде, как это показано на третьей диаграмме. Наконец, в пятом периоде (см. нижнюю диаграмму), процесс окраски уже не является активным, и автомобиль окрашивается в красный цвет.



Vehicle Occupants Group — Группа пассажиров автотранспортного средства; possibly injured — возможно, получили травмы; being helped — была оказана помощь; Automatic Crash Responding — Автоматическое реагирование на аварию

Рисунок D.4 — Исполнение OPM-модели процесса автоматического реагирования на столкновение

На рисунке D.4 представлены три OPCAT-снимка с экрана, показывающие три стадии исполнения OPM-модели. На левом снимке приведена система до возникновения процесса **Automatic Crash Responding** (Автоматическое реагирование на аварию). На этом этапе объект **Vehicle Occupants Group** (Группа пассажиров автотранспортного средства) находится в своем исходном состоянии, возможно, когда пассажиры находятся в состоянии possibly injured (получили травмы); это состояние выделено сплошным коричневым цветом.

Средний снимок демонстрирует указанный процесс в действии (выделен сплошным синим цветом). В течение того времени, когда процесс **Automatic Crash Responding** активен (т. е. когда он исполняется), объект **Vehicle Occupants Group** находится в переходном периоде от его исходного состояния, когда, возможно, пассажиры находятся в состоянии **possibly injured** (возможного получения травмы), в свое конечное состояние **being helped** (оказание помощи); это состояние выделено светло-коричневым цветом.

Наблюдая за этой анимацией в действии, исходная окраска будет постепенно исчезать, тогда как окраска выходного состояния будет постепенно нарастать, а две красные точки будут перемещаться по паре линий связи, обозначающих «контроль» системы (или где система находится в каждый момент времени). Одна красная точка перемещается от входного состояния к воздействующему процессу, тогда как вторая точка перемещается от этого процесса вдоль выходной связи к выходному состоянию.

И, наконец, на снимке справа показана система после того, как процесс **Automatic Crash Responding** (Автоматическое реагирование на аварию) завершается. На данном этапе объект **Vehicle Occupants Group** будет находиться в состоянии **being helped** (была оказана помощь).

Анимационное (динамическое) исполнение модели системы имеет ряд преимуществ. Во-первых, оно представляет собой динамическую визуализацию, которая помогает как разработчику модели, так и целевой группе с течением времени отслеживать и понимать поведение системы. Во-вторых, как отладчик языка программирования оно облегчает проверку динамики системы и обнаружение логических ошибок проектирования в потоке сигналов контроля исполнения, поэтому настоятельно рекомендуется часто анимировать модель системы во время ее создания.

D.7 Продолжительность процесса

Единицей системного времени по умолчанию является единица времени, используемая по умолчанию для указания всех видов длительности всех системных процессов (если не существует различных единиц времени для конкретных процессов; в этом случае последние должны иметь приоритеты перед единицей системного времени).

Компактный способ выражения соответствующих значений свойства на OPD-диаграмме связан с использованием связей типа «представление-характеризация» и «специализация». Если предположить, что следующее присуще соответствующим свойствам процесса, то в примере 1 выражены два способа графического конфигурирования свойств:

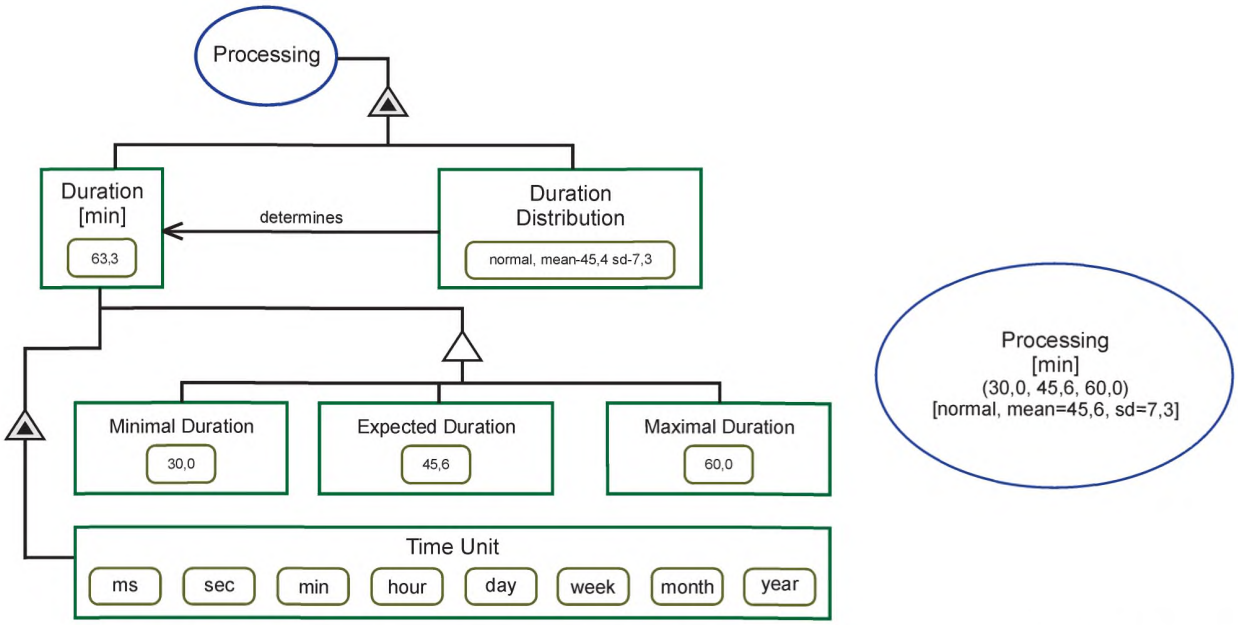
- единица измерения времени;
- параметры продолжительности времени, которым может быть одно из следующих:
  - три значения, характеризующие минимальную, ожидаемую и максимальную продолжительности соответственно,
  - два значения, характеризующие минимальную и максимальную продолжительности соответственно или
  - одно значение, характеризующее как минимальную, так и максимальную продолжительности; а также
  - наименование распределения по продолжительностям и его один или несколько параметров.

Ниже указаны возможные стандартные распределения и их параметр (параметры):

- Нормальное распределение, среднее значение = xx; sd = yy;
- Равномерное распределение, a = xx, b = yy;
- Экспоненциальное распределение, λ = xx.

Примечание — Единица измерения времени в секундах используется по умолчанию; ее часто опускают.

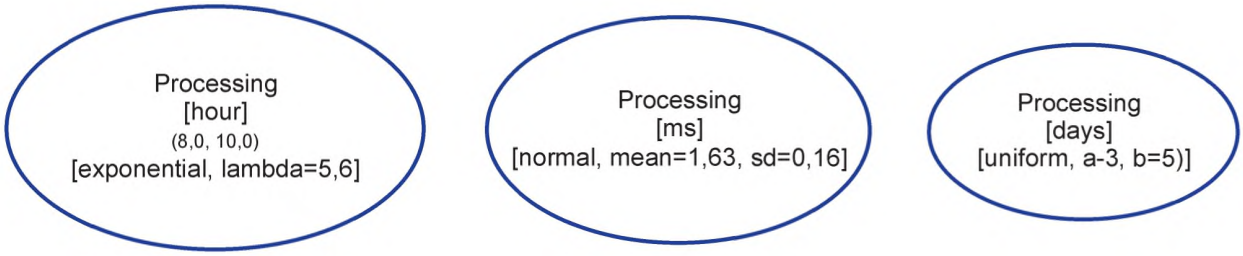
Пример 1 — На рисунке D.5 приведена метамодель для объекта Processing Duration (Продолжительность процесса) вместе со значениями свойств. Слева изображена полная метамодель, а справа иллюстрируется компактный способ регистрации всех данных, за исключением (фактического) значения Duration (Продолжительность), которое относится к свойству времени исполнения. Объект Duration Distribution (Распределение продолжительностей) в данном примере относится к нормальному распределению со средним значением 45,6 мин. и стандартным отклонением 7,3 мин.



Процесс **Processing** (Обработка) представлен значениями **30,0**, **45,6** и **60,0 min** для объектов **Minimal Duration** (Минимальная продолжительность), **Expected Duration** (Ожидаемая продолжительность) и **Maximal Duration** (Максимальная продолжительность) соответственно и состоянием **normal** (нормальное) для объекта **Duration Distribution** (Распределение продолжительности) со значениями параметров **mean** (среднее значение) **=45,6** и **sd** (стандартное отклонение)**=70,0**.

Рисунок D.5 — Объект **Processing Duration** со значениями свойств

Пример 2 — На рисунке D.6 приведены примеры продолжительности процесса.



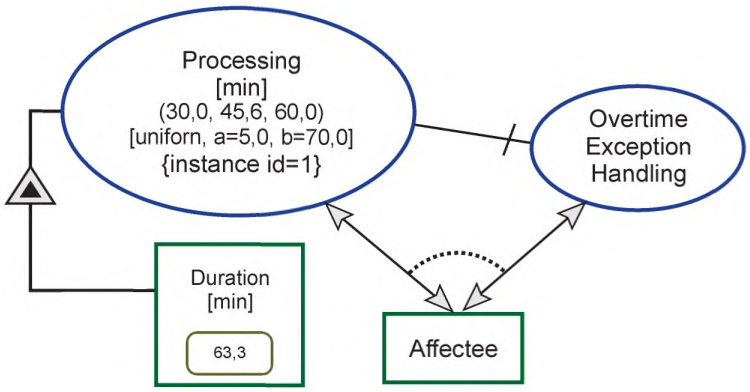
Процесс **Processing** (Обработка) представлен значениями **8,0 h** и **10,0 h** объектов **Minimal Duration** (Минимальная продолжительность) и **Maximal Duration** (Максимальная продолжительность) соответственно и состоянием **exponential** (экспоненционное) объекта **Duration Distribution** (Распределение продолжительностей) с параметром **lambda=5,6**.

Процесс **Processing** (Обработка) представлен состоянием **normal** (нормальное) объекта **Duration Distribution** (Распределение продолжительностей) с параметрами **mean** (среднее значение)**=1,63** и **sd** (стандартное отклонение)**=0,16 ms**.

Процесс **Processing** (Обработка) представлен состоянием **uniform** (равномерное) объекта **Duration Distribution** (Распределение продолжительностей) с параметрами **a=3** and **b=5 days**.

Рисунок D.6 — Примеры продолжительностей процессов

Пример 3 — На рисунке D.7 для процесса Processing (Обработка) {экземпляр id=1} объект Duration (Продолжительность) имеет значение 63,3 min, поэтому существует процесс Overtime Exception Handling.



Процесс **Processing** (Обработка) представляется значениями **30,0**, **45,6** и **60,0 min** для объектов **Minimal Duration** (Минимальная продолжительность), **Expected Duration** (Ожидаемая продолжительность) и **Maximal Duration** (Максимальная продолжительность) соответственно и состоянием **uniform** (равномерное) для объекта **Duration Distribution** (Распределение продолжительностей) с параметрами **a=5,0 и b=70,0**.

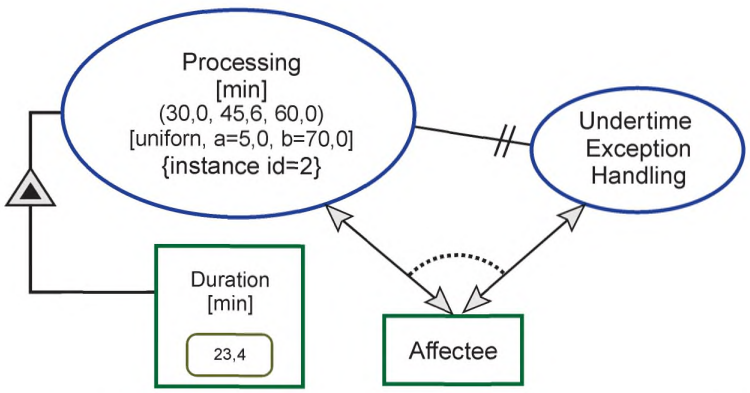
Процесс **Processing** (Обработка) или **Overtime Exception Handling** взаимодействует с объектом типа **Affectee.**

Процесс **Overtime Exception Handling** возникает, если продолжительность процесса **Processing** (Обработка) будет превышать значение **60,0 min**.

Процесс **Overtime Exception Handling** взаимодействует с объектом типа **Affectee**.

Рисунок D.7 — Пример процесса **Overtime exception**

Пример 4 — На рисунке D.8 процесс Processing (Обработка) {экземпляр id=2} и объект Duration (Продолжительность) имеет значение 23,4 min, поэтому возникает процесс Undertime Exception Handling.



Процесс **Processing** (Обработка) представляется значениями **30,0**, **45,6** и **60,0 min** для объектов **Minimal Duration** (Минимальная продолжительность), **Expected Duration** (Ожидаемая продолжительность) и **Maximal Duration** (Максимальная продолжительность) соответственно и состоянием **uniform** (равномерное) для объекта **Duration Distribution** (Распределение продолжительностей), с параметрами **a=5,0** и **b=70,0**.

Процесс **Processing** (Обработка) или **Undertime Exception Handling** взаимодействует с объектом типа **Affectee.** Процесс **Undertime Exception Handling** возникает, если продолжительность процесса **Processing** (Обработка) составляет меньше **60,0 min.** Процесс **Undertime Exception Handling** взаимодействует с объектом типа **Affectee**.

Рисунок D.8 — Пример процесса **Undertime Exception Handling**

Библиография

[1] ИСО/МЭК 14977:1996 Информационные технологии. Синтаксический метаязык. Расширенная нормальная форма Бэкуса (BNF)[1]

[2] Стандарт ИСО/ТК 184/ПК 5 Техническое задание: Исследовательская группа для изучения OPM-методологии для моделирования стандартов, 2009

[3] Стандарт ИСО/ТК 184/ПК 5 N1070 Исследовательская группа для изучения объектно-процессуальной методологии. Промежуточный отчет, 2010

[4] Стандарт ИСО/ТК 184/ПК 5 N1111 Исследовательская группа для изучения объектно-процессуальной методологии. Заключительный отчет, 2011

[5] Bibliowicz A. A. Графическая, основанная на грамматике формальная проверка объектно-процессуальной диаграммы, M. Sc. Thesis, Technion, Израиль, 2008

[6] Bibliowicz A., Dori D. A. Графическая, основанная на грамматике формальная проверка объектно-процессуальной диаграммы. Soft. Syst. Model. 2012, 11 (2) с. 287—302

[7] Crawley E. F., Malmqvist J., Östlund S., Brodeur D. R. Переосмысливание инженерного образования: The CDIO Approach. Springer, 2007

[8] Dori D. Объектно-процессуальная методология. Целостная системная парадигма. Springer Verlag, Berlin, 2002

[9] Dori D. Термины из изображений для двухканальной обработки. Бимодальное графически-текстовое представление сложных систем. Commun. ACM. 2008, 51 (5) с. 47—52

[10] Dori D., Feldman R., Sturm A. От концептуальных моделей к схемам. Объектно-процессуальный метод формирования хранилищ данных, Inf. Syst. 2008, 33 (6) с. 567—593

[11] Dori D. Объектно-процессуальный анализ: Поддержание баланса между структурой системы и поведением. Journal of Logic and Computation. 1995, 5 (2) с. 227—249

[12] Dori D. Объектно-процессуальная методология. Целостная системная парадигма. Springer Verlag, с предисловием Edward Crawley, Берлин, Heidelberg, Нью-Йорк, 2002

[13] Dori D., Reinhartz-Berger I., Sturm A. LNCS 2813, с. 570—572, 2003

[14] Dori D. Международный журнал сверхбольших баз данных. 2004, 13 (2) с. 120—147

[15] Estefan J. Обзор методологий технических наук, основанных на системном проектировании с использованием моделей (MBSE) 2. Дифференциация методологий процессов, методов и моделей жизненного цикла. Jet Propuls. 2008, 25 с. 1—70. Доступно по адресу: http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf

[16] Grobshtein Y., Dori D. Формирование SysML-представлений из OPM-модели. Проектирование и оценка. Syst. Eng. 2011, 14 (3) с. 327—340

[17] Myersdorf D., Dori D. R & D-пространство и его циклы обратной связи. Объектно-процессуальный анализ. R & D Manag. 1997, 27 (4) с. 333—344

[18] Oliver D.W., Andary J.F., Frisch H. Основанные на моделях инженерные системы. В «Справочнике по инженерно-техническим системам и управлению», с. 1361—1400, 2009

[19] Osorio C.A., Dori D., Sussman J. COIM: Объектно-процессуальный метод анализа сложных архитектур сложных взаимосвязанных крупномасштабных социо-технических систем. Syst. Eng. 2011, 14 (3)

[20] Peleg M., Dori D. Проблема кратности модели. Эксперименты с методами спецификации в режиме реального времени. IEEE Trans. Softw. Eng. 2000, 26 (8) с. 742—759

[21] Peleg M.J. and D. A. Методология выявления и моделирования исключений. (4), с. 736—747, 2009

[22] OPCAT. Лаборатория моделирования корпоративных компьютерных систем, Technion, Хайфа, Израиль, http://esml.iem.technion.ac.il/opm/

[23] Ramos A.L., Ferreira J.V., Barceló J. LITHE: Гибкая методология программирования для ориентированного на человека и основанного на модели системного проектирования. IEEE Trans. Syst. Man Cybern. A Syst. Hum. 2012

[24] Reichwein A., Paredis C. Обзор концепций архитектуры и языков моделирования для основанных на моделях систем для технических наук. Труды ASME 2011 Международной конференции по техническому проектированию, компьютерам и информации, с. 1—9, 2011

[25] Reinhartz-Berger I., Dori D. A: Мета-модель, отражающая объектно-процессуальную методологию. Строительные блоки для систем моделирования, B: Анализ бизнес-систем с онтологиями, (Green P., Rosemann M., eds.). Idea Group, Hershey: 2005, pp. 130—73

[26] Sharon A., de Weck O., Dori D. Модель-ориентированное проектирование матричной структуры: Получение DSM из объектно-процессуальной модели. Syst. Eng. 2012, pp. 1—14

[1] Доступно по адресу: http://isotc.iso.org/livelink/livelink/fetch/2000/2489/lttf_Home/PubliclyAvailableStandards.htm.

[27] Somekh J., Choder M., Dori D. Концептуальные, основанные на модели биологические системы: Картирование знаний и выявление недочетов в mRNA-цикле транскрипции. PLoS ONE. 2012 Dec. 20, 7 (12), стр. e51430. DOI:10.1371/journal.pone.0051430

[28] Soffer P., Golany B., Dori D. ERP-моделирование: Комплексный подход. Inf. Syst. 2003, 28 (6), с. 673—690

[29] Sturm A., Dori D., Shehory O. Объектно-процессуальный язык моделирования многоагентных систем. IEEE Trans. Syst. Man Cybern. C. 2010, 40 (2) pp. 227—241

[30] Sturm A., Dori D., Shehory O. Основанный на приложениях подход к доменному анализу и его объектно-процессуальная методология реализация. Int. J. Softw. Eng. Knowl. Eng. 2009 February, 19 с. 1

[31] Yaroker Y., Perelman V., Dori D. OPM-концептуальная, основанная на моделях среда исполнения: Реализация и оценка. Syst. Eng. 2013, 16 (4) стр. 381—390

УДК 658.52.011.56:006.354 ОКС 25.040

Ключевые слова: системы промышленной автоматизации, интеграция, объектно-процессуальная методология

Редактор *А.Е. Петросян*
Корректор *Е.Р. Ароян*
Компьютерная верстка *Ю.В. Половой*

Сдано в набор 07.12.2016. Подписано в печать 27.01.2017. Формат 60 × 84 1/8. Гарнитура Ариал.
Усл. печ. л. 19,53.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Набрано в ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
www.jurisizdat.ru y-book@mail.ru

Издано во ФГУП «СТАНДАРТИНФОРМ», 123995, Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru