Business Process Model and Notation (BPMN)

Version 2.0

Графический язык моделирования бизнес-процессов BPMN

Версия 2.0

Введение

Данный документ представляет собой субъективный перевод спецификации языка моделирования бизнес-процессов BPMN (Business Process Management Notation).

Нотация BPMN была разработана организацией «Business Process Management Initiative (BPMI)», и поддерживается группой компаний «Object Management Group. Текущая версия BPMN – 1.1. Оригинальная спецификация изготовлена группой компаний «Object Management Group».

Нотация BPMN положена в основу системы управления бизнес-процессами «ELMA», разработанной компанией «EleWise», а также ряда других приложений для бизнес-среды.

Настоящий документ создан компанией «EleWise» и предназначен для внутреннего использования, а также для общего ознакомления всех желающих с принципами построения моделей бизнес-процессов на графическом языке BPMN.

Ссылки

- Сайт системы управления бизнес-процессами «ELMA»
  elma-bpm..ru
- Сайт компании «EleWise»
  www.elewise.ru
- Сайт BPMN с оригинальными спецификациями от «OMG» (на англ. языке)
  bpmn.org

Содержание

1. Область действия документа

Разработчиком стандарта **BPMN** (**BusinessProcessModelandNotation**) является рабочая группа OMG. Основной целью языка **BPMN** является обеспечение абсолютно доступной нотацией для описания бизнес-процессов всех бизнес-пользователей: от бизнес-аналитиков, создающих схемы процессов, и разработчиков, ответственных за внедрение технологий выполнения бизнес-процессов, до руководителей и обычных пользователей, управляющих этими бизнес-процессами и отслеживающих их выполнение. Таким образом, **BPMN** нацелен на устранение расхождения между моделями бизнес-процессов и их реализацией.

Другой, не менее важной целью разработки **BPMN**, явилось то, что языки XML (например, **WSBPEL** - WebServicesBusinessProcessExecutionLanguage), разработанные для исполнения бизнес-процессов, теперь могут быть визуализированы в графической нотации, понятной обычным бизнес-пользователям.

В данном документе объединены лучшие практические наработки в области бизнес-моделирования. Это было сделано для выбора нотации и семантики диаграмм **Взаимодействия** (**Collaboration**), **Процессов** (**Process**) и **Хореографии** (**Choreography**). Предназначение **BPMN** – стандартизировать модель бизнес-процесса и нотацию перед лицом множества различных нотаций моделирования и точек зрения. Таким образом, благодаря **BPMN**,бизнес-пользователи, внедренцы, заказчики и поставщики получают простые средства доступа к информации о процессе.

Члены группы OMG произвели оценку и проверили на практике различные нотации, после чего объединили лучшие идеи, предлагаемые этими противоречивыми нотациями, в единую стандартизированную нотацию. Были рассмотрены следующие нотации и методики моделирования: UMLActivityDiagram, UMLEDOCBusinessProcesses, IDEF, ebXMLBPSS, Activity-DecisionFlow (ADF) Diagram, RosettaNet, LOVeM и Event-ProcessChains (EPCs).

2. Соответствие требованиям спецификации

Программное обеспечение может претендовать на звание совместимого с **BPMN 2.0** или соответствующего **BPMN 2.0** только в том случае, если оно полностью согласовывается с применяемыми точками контроля установленных требований, описанных в документе. Программное обеспечение, лишь частично соответствующее применяемыми точками контроля установленных требований, может быть заявлено только как «основанное на **BPMN**», однако, не может являться совместимым с данной спецификацией. Данная спецификация выделяет четыре типа соответствия: **Соответствие требованиям моделирования Процесса** (**ProcessModelingConformance**), **Соответствие требованиям исполнения Процесса** (**ProcessExecutionConformance**), **Соответствие требованиям исполнения Процессов BPEL** (**BPELProcessExecutionConformance**), **Соответствие требованиям моделирования Хореографии** (**ChoreographyModelingConformance**).

Для удовлетворения **Требованиям Моделирования Процесса** (**ProcessModelingConformance**) НЕ ОБЯЗАТЕЛЬНО учитывать правила соответствия **Требованиям Моделирования Хореографии** (**ChoreographyModelingConformance**), и наоборот. Аналогично, соблюдение правил соответствия **Требованиям Исполнения Процесса** НЕ является ОБЯЗАТЕЛЬНЫМ условием для соответствия **Требованиям Моделирования Процесса** или **Требованиям Моделирования Хореографии**.

Для соответствия **Требованиям Моделирования Процесса** (**ProcessModelingConformance**) ДОЛЖНЫ БЫТЬ выполнены все требования, описанные в подразделе 2.1 данного документа. Для Соответствия **Требованиям Исполнения Процесса** (**ProcessExecutionConformance**) ДОЛЖНЫ БЫТЬ удовлетворены все требования, описанные в подразделе 2.2. Для соответствия **Требованиям Применения Исполнительной Семантики Процесса BPEL** (**BPELProcessExecutionSemanticsConformancetype**) ДОЛЖНЫ БЫТЬ удовлетворены все требования, описанные в подразделе 2.3. Для соответствия **Требованиям Моделирования Хореографии** (**ChoreographyConformance**) ДОЛЖНЫ БЫТЬ удовлетворены все требования, описанные в подразделе 2.4. Таким образом, реализация может считаться выполненной в соответствии со всеми предъявляемыми требованиями в том случае, если она произведена с учетом требований, описанных в подразделах 2.1, 2.2, 2.3, и 2.4 данного документа.

2.1. Соответствие Требованиям Моделирования Процесса (ProcessModelingConformance)

В следующих восьми подразделах описываются правила соответствия **Требованиям Моделирования Процесса** (**ProcessModelingConformance**).

2.1.1. Типы Процессов BPMN

Для обеспечения соответствия **Требованиям Моделирования Процессов** НЕОБХОДИМА возможность поддержки следующих пакетов **BPMN**:

- Основные элементы **BPMN**, включающие элементы пакетов: *Инфраструктура* (*Infrastructure*), *Фундамент* (*Foundation*), *Общий* (*Common*) и *Сервис* (*Service*)(см. Главу 8).
- Диаграммы Процессов, в состав которых входят элементы пакетов: **Процесс** (**Process**), **Действия** (**Activities**), *Данные* (*Data*) и *Пользовательские действия* (*HumanInteraction*) (см. Главу 10).
- Диаграммы **Взаимодействий** (**Collaboration**), в состав которых входят **Пулы** (**Pools**) и **Потоки сообщений** (**MessageFlow**) (см. Главу 9).
- Диаграммы **Обмена сообщениями** (**Conversation**) , в состав которых входят **Пулы** (**Pools**), элементы **Обмен сообщениями** (**Conversations**) и **Соединители** элементов **Обмен сообщениями** (**ConversationLinks**) (см. Главу 9).

Альтернативой полному соответствию требованиям моделирования Процесса (**ProcessModelingConformance**) являются три подкласса соответствия:

- **Подкласс для Описания**(**Descriptive**)
- **Аналитический подкласс**(**Analytic**)
- **Подкласс Общего Выполнения** (**CommonExecutable**)

Подкласс для Описания касается отображаемых элементов и атрибутов, используемых в моделях высокого уровня. Это особенно удобно для аналитиков, привыкших использовать инструменты BPA для составления блок-схем.

Аналитический подкласс содержит все подклассы для описания и в общей сложности половину всех конструкций Класса **Соответствия Требованиям Моделирования Процесса** (**ProcessModelingConformance** Class). За основу был взят объединенный опыт использования BPMN, а также анализ ориентированных на пользователя образцов DoDAF (DepartmentofDefenseArchitectureFramework) и результатов плановой стандартизации данной среды.

Как **Подкласс для Описания**, так и **Аналитический подкласс** ориентированы на отображаемые элементы и содержат минимальное количество поддерживающих атрибутов/элементов.

Подкласс Общего Выполнения ориентирован на то, что требуется моделям выполняемых процессов.

Элементы и атрибуты, не относящиеся к вышеописанным подклассам, содержатся в классе **Соответствия Требованиям Моделирования Процесса** (**ProcessModelingConformance** class)

Элементы, входящие в состав каждого из подклассов, описаны в следующем разделе.

2.1.2. Элементы Процесса BPMN

Соответствие **Требованиям моделирования Процесса** (**ProcessModelingConformance**) включает элементы, входящие в состав диаграмм **Взаимодействия** и **Процесса**, а именно: **Задачи** всех типов, *встроенные* **Подпроцессы**, **Действия** типа **Вызов**, **Шлюзы** всех типов, **События** всех типов (**Стартовые**, **Промежуточные**, **Конечные**), **Дорожки**, *Участников*, **Объекты данных** (также **Входные** и **Выходные данные**), **Сообщения**, **Группы**, **Текстовые аннотации**, **Потоки операций** (также *условные* **Потоки операций** и **Потоки операций** *по умолчанию*), **Потоки сообщений**, **Обмен сообщениями** (ограниченный

до группировки **Потоков сообщений**, *ассоциативных корреляций*), *Корреляцию*, **Ассоциацию** (также **Ассоциацию компенсации**). В этот набор также входят маркеры (маркеры **Цикла**, **Многоэкземплярные**, **Транзакции**, **Компенсации**), используемые для дифференцировки **Задач** и *встроенных* **Подпроцессов**.

Примечание: Такие элементы моделирования **Хореографии**, как **Задача Хореографии** и **Подхореография**, в этот перечень не входят.

Для того чтобы инструмент моделирования мог декларировать поддержку подкласса, он ДОЛЖЕН подходить под следующие критерии:

- В нем ДОЛЖНЫ поддерживаться все элементы подкласса.
- В нем ДОЛЖНЫ поддерживаться все перечисленные атрибуты каждого из элементов подкласса.
- В целом, если атрибут не упоминается в подклассе и НЕ является ОБЯЗАТЕЛЬНЫМ для отображения в схеме, то он не входит в данный подкласс. Исключения для этого правила описаны в примечании.

Подкласс соответствия для Описания

Таблица 2.1 содержит информацию о **Подклассе** соответствия для **Описания**.

Таблица 2.1 – Элементы и Атрибуты Подкласса Соответствия для Описания

Элемент	Атрибуты
participant (pool)	id, name, processRef
laneSet	id, lane with name, childLaneSet, flowElementRef
sequenceFlow (unconditional)	id, name, sourceRef, targetRef
messageFlow	id, name, sourceRef, targetRef
exclusiveGateway	id, name
parallelGateway	id, name
task (None)	id, name
userTask	id, name
serviceTask	id, name
subProcess (expanded)	id, name, flowElement
subProcess (collapsed)	id, name, flowElement
CallActivity	id, name, calledElement
DataObject	id, name
TextAnnotation	id, text
association/dataAssociationa	id, name, sourceRef, targetRef, associationDirection[b]
dataStoreReference	id, name, dataStoreRef
startEvent (None)	id, name
endEvent (None)	id, name

messageStartEvent	id, name, messageEventDefinition
messageEndEvent	id, name, messageEventDefinition
timerStartEvent	id, name, timerEventDefinition
terminateEndEvent	id, name, terminateEventDefinition
documentationc	text
Group	id, categoryRef

a) **Ассоциация данных** является АБСТРАКТНОЙ, т.е. **Ассоциация** с **Входными** и **Выходными данными** появляется в сериализации XML. У этих типов Ассоциаций есть ОБЯЗАТЕЛЬНЫЕ атрибуты [`sourceRef` и `targetRef`], относящиеся к элементам `itemAwareElements`. Для согласования с метамоделью возникает необходимость в дополнительных элементах: `ioSpecification`, `inputSet`, `outputSet`, **DataInput**, **DataOutput**. Когда в редакторе **BPMN** отображается **Ассоциация данных**, относящаяся к **Действию** или **Событию**, должна сформироваться невидимая подконструкция. Другими словами, было бы необходимо изменять метамодель для того, чтобы сделать атрибуты `sourceRef` и `targetRef` опциональными или допустить использование ссылки на элемент, не относящийся к `itemAwareElements` (например, **Действие** или **Событие**).

b) Атрибут `associationDirection`, не указанный для **Ассоциации данных**.

c) `Documentation` не отображается. Он является атрибутом практически всех элементов.

Аналитический подкласс соответствия

Аналитический подкласс соответствия включает в себя все элементы Подкласса соответствия для **Описания**, а также элементы, приведенные в таблице 2.2.

Таблица 2.2 – Элементы и Атрибуты Аналитического Подкласса Соответствия

Элемент	Атрибуты
sequenceFlow (conditional)	id, name, sourceRef, targetRef, conditionExpression[a]
sequenceFlow (default)	id, name, sourceRef, targetRef, default[b]
sendTask	id, name
receiveTask	id, name
LoopingActivity	standardLoopCharacteristics
MultiInstanceActivity	multiInstanceLoopCharacteristics
exclusiveGateway	Adddefaultattribute
inclusiveGateway	id, name, eventGatewayType
eventBasedGateway	id, name, eventGatewayType
Link catch/throw Intermediate Event	Id, name, linkEventDefinition
signalStartEvent	id, name, signalEventDefinition
signalEndEvent	id, name, signalEventDefinition

CatchingmessageIntermediateEvent	id, name, messageEventDefinition
ThrowingmessageIntermediateEvent	id, name, messageEventDefinition
BoundarymessageIntermediateEvent	id, name, attachedToRef, messageEventDefinition
Non-interrupting Boundary message Intermediate Event	id, name, attachedToRef, cancelActivity=false, messageEventDefinition
CatchingtimerIntermediateEvent	id, name, timerEventDefinition
BoundarytimerIntermediateEvent	id, name, attachedToRef, timerEventDefinition
Non-interrupting Boundary timer Intermediate Event	id, name, attachedToRef, cancelActivity=false, timerEventDefinition
BoundaryerrorIntermediateEvent	id, name, attachedToRef, errorEventDefinition
errorEndEvent	id, name, errorEventDefinition
Non-interrupting Boundary escalation Intermediate Event	id, name, attachedToRef, cancelActivity=false, escalationEventDefinition
ThrowingescalationIntermediateEvent	id, name, escalationEventDefinition
escalationEndEvent	id, name, escalationEventDefinition
CatchingsignalIntermediateEvent	id, name, signalEventDefinition
ThrowingsignalIntermediateEvent	id, name, signalEventDefinition
BoundarysignalIntermediateEvent	id, name, attachedToRef, signalEventDefinition
Non-interrupting Boundary signal Intermediate Event	id, name, attachedToRef, cancelActivity=false, signalEventDefinition
conditionalStartEvent	id, name, conditionalEventDefinition
CatchingconditionalIntermediateEvent	id, name, conditionalEventDefinition
BoundaryconditionalIntermediateEvent	id, name, conditionalEventDefinition
Non-interrupting Boundary conditional Intermediate Event	id, name, cancelActivity=false, conditionalEventDefinition
messagec	id, name, add messageRef attribute to messageFlow

a) Значение `ConditionExpression`, используемого только для **Потока операций**, исходящего из **Шлюза**, МОЖЕТ БЫТЬ равно нулю.
b) Атрибут `Default` является атрибутом `sourceRef` **Шлюза** (эксклюзивного или неэксклюзивного).
c) Обратите внимание, что атрибут `messageRef`, являющийся атрибутом различных **Событий** типа Сообщение, является опциональным и не входит в данный подкласс.

Подкласс Соответствия Общему выполнению

Данный подкласс соответствия предназначен для инструментов моделирования, выпускающих выполняемые модели.

- Языком определения типа данных ДОЛЖЕН БЫТЬ XMLSchema.

- Языком определения интерфейсов Web-сервисов ДОЛЖЕН БЫТЬ WSDL.
- Языком доступа к данным ДОЛЖЕН БЫТЬ XPath.

Элементы подкласса соответствия **Общему выполнения** приведены в таблице 2.3, а сопутствующие классы отображены в таблице 2.4.

Таблица 2.3 – Элементы и Атрибуты Подкласса Соответствия Общему Выполнению

Элемент	Атрибуты
sequenceFlow (unconditional)	id, (name), sourceRef[a], targetRef[b]
sequenceFlow (conditional)	id, name, sourceRef, targetRef, conditionExpression[c]
sequenceFlow (default)	id, name, sourceRef, targetRef, default[d]
subProcess (expanded)	id, name, flowElement, loopCharacteristics, boundaryEventRefs
exclusiveGateway	id, name, gatewayDirection (только сходящийся и расходящийся), default
parallelGateway	id, name, gatewayDirection (только сходящийся и расходящийся)
startEvent (None)	id, name
endEvent (None)	id, name
eventBasedGateway	id, name, gatewayDirection, eventGatewayType
userTask	id, name, renderings, implementation, resources, ioSpecification, dataInputAssociations, dataOutputAssociations, loopCharacteristics, boundaryEventRefs
sigserviceTask	id, name, implementation, operationRef, ioSpecification, dataInputAssociations, dataOutputAssociations, loopCharacteristics, boundaryEventRefs
callActivity	id, name, calledElement, ioSpecification, dataInputAssociations, dataOutputAssociations, loopCharacteristics, boundaryEventRefs
dataObject	id, name, isCollection, itemSubjectRef
textAnnotation	id, text
dataAssociation	id, name, sourceRef, targetRef, assignment
messageStartEvent	id, name, messageEventDefinition (either ref or contained), dataOutput, dataOutputAssociations
messageEndEvent	id, name, messageEventDefinition, (либо ссылка, либо наличие), dataInput, dataInputAssociations

terminateEndEvent	(Триггер завершения в комбинации с одним из конечных событий)
CatchingmessageIntermediateEvent	id, name, messageEventDefinition (либо ссылка, либо наличие), dataOutput, dataOutputAssociations
ThrowingmessageIntermediateEvent	id, name, messageEventDefinition (либо ссылка, либо наличие), dataInput, dataInputAssociations
CatchingtimerIntermediateEvent	id, name, timerEventDefinition (наличие)
daryerrorIntermediateEvent	id, name, attachedToRef, errorEventDefinition, (ссылка или наличие), dataOutput, dataOutputAssociations

a) Множественные исходящие соединения допускаются только для **Шлюзов**, в которых маршруты сходятся.
b) Множественные исходящие соединения допускаются только для **Шлюзов**, в которых маршруты расходятся.
c) Значение `ConditionExpression`, используемого только в направленных от **Шлюзов Потоках операций**, МОЖЕТ БЫТЬ равно нулю.
d) Атрибут `Default` является атрибутом `sourceRef` **Шлюза** (эксклюзивного или неэксклюзивного).

Таблица 2.4 – Сопутствующие Классы Подкласса Соответствия Общему Выполнению

Элемент	Атрибуты
StandardLoopCharacteristics	id, loopCondition
MultiInstanceLoopCharacteristics	id, isSequential, loopDataInput, inputDataItem
Rendering	
Resource	id, name
ResourceRole	id, resourceRef, resourceAssignmentExpression
InputOutputSpecification	id, dataInputs, dataOutputs
DataInput	id, name, isCollection, itemSubjectRef
DataOutput	id, name, isCollection, itemSubjectRef
ItemDefinition	id, structureorimporta
Operation	id, name, inMessageRef, outMessageRef, errorRefs
Message	id, name, structureRef
Error	id, structureRef
Assignment	id, from, tob
MessageEventDefinition	id, messageRef, operationRef

TerminateEventDefinition	id
TimerEventDefinition	id, timeDate

a) Структуру ДОЛЖЕН определять составной тип XSD (XSD ComplexType).
b) Структуру ДОЛЖЕН определять составной тип XSD (XSD ComplexType).

2.1.3. Внешний вид

Ключевым элементом **BPMN** является выбор форм и иконок, соответствующих описанным в данной спецификации элементам. Намерением разработчиков спецификации явилось создание стандартного визуального языка, легко распознаваемого и понятного всем разработчикам моделей процессов. Создание и отображение Диаграмм **Процессов BPMN** подразумевает ОБЯЗАТЕЛЬНОЕ использование графических элементов, форм и маркеров, описанных в данном документе.

Примечание: Если не указано конкретного применения, правила выбора необходимого размера, цвета, линий графических элементов, а также расположения в них текста являются достаточно гибкими (см. подраздел 7.4).

При моделировании диаграммы **BPMN** допускаются следующие расширения:

- Для определенных графических элементов МОГУТ БЫТЬ добавлены новые маркеры (индикаторы). Такие маркеры или индикаторы могут использоваться для выделения конкретного атрибута элемента BPMN либо для отображения нового подтипа соответствующего понятия.
- На Диаграмму в качестве вариации Артефакта МОЖЕТ БЫТЬ добавлена новая форма, однако, эта новая форма НЕ ДОЛЖНА противоречить форме, определенной для какого-либо другого элемента **BPMN** или маркера.
- Графические элементы МОГУТ БЫТЬ выделены цветом. Использование какого-то цвета МОЖЕТ соответствовать определенной семантике, используемой для пополнения информации, которую, согласно данному документу, передают графические элементы.
- Стили линий, используемые для отображения графических элементов, МОГУТ БЫТЬ изменены, однако, они НЕ ДОЛЖНЫ противоречить каким-либо другим стилям линий, УКАЗАННЫМ в данной спецификации.
- Расширение НЕ ДОЛЖНО изменять установленной формы любого графического элемента или маркера (например, квадрат не может быть заменен треугольником, закругленные края не могут стать острыми и т.д.)

2.1.4. Соответствие структуры

Создание и отображение диаграмм **BPMN** ДОЛЖНО БЫТЬ осуществлено в соответствии с техническими требованиям и ограничениям, при этом следует обращать внимание на соединения и другие взаимодействия между графическими элементами на схеме. Там, где разрешенные или требуемые соединения являются условными и основываются на атрибутах соответствующего элемента, ДОЛЖНО БЫТЬ гарантировано сообщение между соединениями и значениями этих атрибутов.

Примечание: В целом, к таким соединениям и взаимоотношениям применяется особая семантическая интерпретация, которая определяет взаимоотношения между понятиями процесса, представленными графическими элементами. Условные взаимоотношения основываются на атрибутах, которые представляют собой различные типы поведения. Именно поэтому соответствие структуры гарантирует корректную интерпретацию диаграммы как спецификации процесса. На протяжении всего документа в различных параграфах описываются конструктивные характеристики. В тексте они отображаются при помощи маркера абзаца ●, например: ● **ЗАДАЧА** МОЖЕТ являться целью Потока операций и иметь множество *входящих* Потоков операций. *Входящий* Поток операций МОЖЕТ поступать по альтернативному и/или параллельному маршрутам.

2.1.5. Семантика Процесса

В данной спецификации приводится несколько понятий семантики, используемых для определения **Процесса**; при этом они ассоциируются с графическими элементами, маркерами и соединениями. Поскольку при применении диаграмма **BPMN** интерпретируется как семантическая спецификация данного **Процесса**, такая интерпретация ДОЛЖНА согласовываться с описанной в данном документе семантической интерпретацией. Другими словами, подгонка под требования соответствия моделированию **Процесса BPMN** (**BPMNProcessModelingConformance**) означает следование семантикам, окружающим элементы диаграммы и описанным в Главе 10.

Примечание: Моделирование в согласии с требованиями соответствия моделированию **Процесса BPMN** (**BPMNProcessModelingConformance**) не подразумевает поддержку исполнительной семантики **BPMN**, описанной в Главе 13.

2.1.6. Атрибуты и ассоциации

Данная спецификация выделяет несколько атрибутов и свойств семантических элементов, представленных посредством графических элементов, маркеров и соединений. Некоторые из них используются лишь для наглядности и обозначены соответствующим образом, другие являются обязательными для отображения. Некоторые атрибуты являются обязательными, однако, могут отображаться по желанию или не отображаются вообще, а некоторые атрибуты являются опциональными. Для адекватного использования каждого обязательного атрибута или свойства ДОЛЖЕН БЫТЬ определен механизм, с помощью которого можно отображать значения атрибутов или свойств. Такой механизм ДОЛЖЕН позволять пользователям создавать или просматривать эти значения для каждого элемента **BPMN**, имеющего атрибут или свойство. Если способ отображения конкретного атрибута или свойства является ОБЯЗАТЕЛЬНЫМ, ДОЛЖНО использоваться только такое отображение. Если способ отображения конкретного атрибута или свойства является опциональным, то при моделировании МОЖЕТ БЫТЬ использован как графический, так и другой способ отображения. Если атрибут или свойство отображается графически, то ДОЛЖНЫ БЫТЬ определены требования для этого. Если же атрибуту или свойству не соответствует никакое графическое представление, то при моделировании МОЖЕТ БЫТЬ использован как графический, так и другой способ отображения. Если используется какое-либо графическое представление, оно НЕ ДОЛЖНО противоречить установленному графическому представлению какого-либо другого элемента **BPMN**.

2.1.7. Расширенные и опциональные элементы

Для адекватного моделирования НЕ ТРЕБУЕТСЯ поддержки элементов или атрибутов, которые в данном документе определены как ненормативные (non-normative) и информативные (informative). Для любой характеристики, подходящей под определение «опциональной», в документе указано:

- То, каким образом такая характеристика отображается на диаграмме.
- Отображается ли характеристика вообще.
- Поддерживается ли данная характеристика.

Для адекватного моделирования сопровождение любых характеристик, поддержка которых опциональна, НЕ является ОБЯЗАТЕЛЬНОЙ. Если при моделировании осуществляется поддержка опциональной характеристики, такая поддержка ДОЛЖНА соответствовать указанным в документе требованиям. Для соответствия требованиям ДОЛЖНЫ поддерживаться любые опциональные характеристики, для которых опциональность заключается в том, как они ДОЛЖНЫ отображаться и ДОЛЖНЫ ли они отображаться вообще.

2.1.8. Перенос визуальной модели

Одной их основных целей создания данного документа является обеспечение форматом обмена данных, используемым для обмена понятиями **BPMN** (как моделями одной предметной области, так и диаграммами) между различными инструментами моделирования. При моделировании должны поддерживаться метамодели типов Процессов, описанные в разделе 13.1. Это позволяет сделать диаграммы процесса переносимыми,

благодаря чему пользователи могут переносить понятия, используемые в среде одного производителя, в среду другого производителя.

2.2. Соответствие исполнению Процесса

В следующих двух подразделах описаны правила **Соответствия Требованиям Исполнения Процесса** (**ProcessExecutionConformance**).

2.2.1. Семантика исполнения

Семантика исполнения **BPMN** полностью формализована с помощью данного документа. Инструмент моделирования, желающий соответствовать **Требованиям Исполнения BPMN** (**BPMN ExecutionConformance**), ДОЛЖЕН полностью поддерживать и уметь интерпретировать операционную семантику и жизненный цикл **Действия**, описанный в подразделе 14.2.2. Непригодные для использования элементы, информация о которых содержится в Главе 14, МОГУТ БЫТЬ опущены при моделировании в соответствии с требованиями исполнения **BPMN**. Такое моделирование ДОЛЖНО полностью поддерживать и уметь интерпретировать базовую метамодель.

Примечание: Инструмент моделирования, желающий соответствовать **Требованиям Исполнения BPMN** (**BPMN ExecutionConformance**), не обязан поддерживать и уметь интерпретировать модели **Хореографии** или соответствовать **Требованиям Моделирования Процесса** (**ProcessModelingConformance**). Точнее, инструмент моделирования не обязан поддерживать графический синтаксис или семантику, описанные в данном документе. В нем МОГУТ быть использованы различные графические элементы, формы и маркеры, а не только те, что указаны в данной спецификации.

2.2.2. Импорт диаграмм Процессов

Инструмент моделирования, желающий соответствовать требованиям исполнения **BPMN** (**BPMNExecutionConformance**), ДОЛЖЕН поддерживать импорт диаграмм **Процессов BPMN**, включая **Взаимодействие** (см. таблицу 10.1).

2.3. Соответствие Требованиям Исполнения Процессов BPEL

Для удовлетворения Требованиям Исполнения Процессов (ProcessExecutionConformance), определяющим соответствие **BPMN** языку WS-BPEL (см. подраздел 15.1), может быть необходимо соответствие **Требованиям Исполнения Процессов BPEL**.

Примечание: Инструмент моделирования, желающий соответствовать **Требованиям Исполнения Процессов BPEL** (**BPELProcessExecutionConformance**), ДОЛЖЕН полностью соответствовать **Требованиям Исполнения Процессов BPMN** (**ProcessExecutionConformance**). Инструмент моделирования, желающий соответствовать **Требованиям Исполнения Процессов BPEL** (**BPELProcessExecutionConformance**), не обязательно должен поддерживать и уметь интерпретировать модели **Хореографии**. Также инструмент моделирования, желающий соответствовать **Требованиям Исполнения Процессов BPEL** (**BPELProcessExecutionConformance**), не обязательно должен соответствовать **Требованиями Моделирования Процессов** (**ProcessModelingConformance**).

2.4. Соответствие Требованиям Моделирования Хореографии

В следующих пяти подразделах описаны **Требования Соответствия Хореографии** (**ChoreographyConformance**).

2.4.1. Типы Хореографий BPMN

При моделировании, удовлетворяющем требованиям соответствия **Хореографии** (**ChoreographyConformance**), ДОЛЖНЫ поддерживаться следующие пакеты **BPMN**:

- Основные элементы **BPMN**, включая те, что указаны в *Инфраструктуре* (*Infrastructure*), *Фундаменте* (*Foundation*), *Общем* (*Common*) и *Сервисе* (*Service*) (см. Главу 8).
- Диаграммы **Хореографии**, содержащие элементы **Хореографии**, и пакеты **Хореографии** (см. Главу 11).
- Диаграммы **Взаимодействия**, содержащие **Пулы** и **Потоки сообщений** (см. Главу 9).

2.4.2. Элементы Хореографии BPMN

Список элементов, удовлетворяющих **Требованиям Соответствия Хореографии** (**ChoreographyConformance**), включает **Сообщения**, **Задачу Хореографии**, **Глобальную Задачу Хореографии**, **Подхореографию** (скрытую и развернутую), некоторые типы **Стартовых событий** (например, **Неопределенный**, **Таймер**, **Условие**, **Сигнал** и **Множественный**), некоторые типы **Промежуточных событий** (например, **Неопределенный**, присоединенное к границе **Действия Сообщение**, **Условие**, **Сигнал**, **Множественный**, Связь и т.д.) и некоторые типы **Конечных событий** (**Неопределенный** и **Завершение**), а также **Шлюзы**. Также, для включения **Хореографии** в состав **Взаимодействия** первая должна поддерживать использование **Пулов** и **Потоков сообщений**.

2.4.3. Общий вид

При создании и отображении диаграмм **Хореографий BPMN** ДОЛЖНЫ использоваться графические элементы, формы и маркеры, указанные в спецификации **BPMN**. Для ознакомления с правилами использования текста, цвета и линий в моделировании диаграмм **Хореографий** см. раздел 7.4.

2.4.4. Семантика Хореографии

Инструмент моделирования, желающий удовлетворять **Требованиям Соответствия Хореографии** (**ChoreographyConformance**), должен полностью поддерживать и уметь интерпретировать графическую и исполнительную семантику элементов диаграммы **Хореографии**, а также типы диаграмм **Хореографии**.

2.4.5. Перенос визуальной модели

При моделировании **Хореографии** должен поддерживаться импорт/экспорт типов диаграмм **Хореографии** и **Взаимодействия**, благодаря чему **Хореография** отображается в рамках **Взаимодействия** (см. описание в разделе 9.4). Это позволяет сделать диаграммы **Хореографий** переносимыми, благодаря чему пользователи могут переносить понятия, используемые в среде одного производителя, в среду другого производителя.

2.5. Обзор типов соответствий BPMN

Таблица 2.5 содержит общую информацию о требованиях соответствия **BPMN** (**BPMN**Conformance).

Таблица 2.5 – Типы соответствия BPMN

Категория	**Соответствие Моделированию Процессов (ProcessModelingConformance)**	**Соответствие Исполнению Процессов (ProcessExecutionConformance)**	**Соответствие Исполнению Процессов BPEL (BPEL ProcessExecutionConformance)**	**Соответствие Хореографии (ChoreographyConformance)**
Визуальное представление типов Диаграмм BPMN	Типы диаграмм Процессов и Хореографий, в которых отображается взаимодействие между типами	Нет данных	Нет данных	Типы диаграмм Хореографий и Взаимодействия, в которых отображается взаимодействие между типами

	диаграмм Процессов.			диаграмм Хореографий.
Элементы Диаграммы BPMN, поддержка которых необходима	Все типы Задач, встроенные Подпроцессы, Действия типа Вызов, все типы Событий, все типы Шлюзов, Пулы, Дорожки, Объекты данных (включая Входные и Выходные данные), Сообщения, Группы, Артефакты, маркеры Задач и Подпроцессов, Потоки операций, Ассоциации, Потоки сообщений.	Нет данных	Нет данных	Сообщения, Задачи Хореографии, Глобальные Задачи Хореографии, Подхореографии (скрытые и развернутые), некоторые типы Стартовых, Промежуточных и Конечных событий, Шлюзы, Пулы, Message Flow.
Импорт/Экспорт типов диаграмм	Да, для диаграмм Процессов и Взаимодействия, в которых Процесс отображается в рамках Взаимодействия.	Да, для диаграмм Процессов	Да, для диаграмм Процессов	Да, для диаграмм Хореографии и Взаимодействия, в которых Хореография отображается в рамках Взаимодействия.
Поддержка Графического синтаксиса и семантики	Диаграммы Процессов и Взаимодействия, в которых Процесс отображается в рамках Взаимодействия.	Нет данных	Нет данных	Диаграммы Хореографии и Взаимодействия, в которых Хореография отображается в рамках Взаимодействия.
Поддержка Исполнительной Семантики	Нет данных	Да, для диаграмм Процессов	Да, для диаграмм Процессов	Исполнительная семантика Хореографии.

3. Нормативные ссылки

3.1. Нормативные

RFC-2119

- Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, IETF RFC 2119, March 1997 http://www.ietf.org/rfc/rfc2119.txt

3.2. Ненормативные

Activity Service

- Additional Structuring Mechanism for the OTS specification, OMG, June 1999
  http://www.omg.org
- J2EE Activity Service for Extended Transactions (JSR 95), JCP
  http://www.jcp.org/jsr/detail/95.jsp

BPEL4People

- WS-BPEL Extension for People (BPEL4People) 1.0, June 2007
- http://www.active-endpoints.com/active-bpel-for-people.htm
- http://www.active-endpoints.com/active-bpel-for-people.htmhttp://www.adobe.com/devnet/livecycle/articles/bpel4people_overview.html
- http://dev2dev.bea.com/arch2arch/
- http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/
- http://www.oracle.com/technology/tech/standards/bpel4people/
- https://www.sdn.sap.com/irj/sdn/bpel4people

BusinessProcessDefinitionMetamodel

- OMG, May 2008,
  http://www.omg.org/docs/dtc/08-05-07.pdf

Business Process Modeling

- Jean-Jacques Dubray, "A Novel Approach for Modeling Business Process Definitions," 2002
  http://www.ebpml.org/ebpml2.2.doc

Dublin Core Meta Data

- Dublin Core Metadata Element Set, Dublin Core Metadata Initiative
  http://dublincore.org/documents/dces/

ebXML BPSS

- Jean-Jacques Dubray, "A new model for ebXML BPSS Multi-party Collaborations and Web Services Choreography," 2002
  http://www.ebpml.org/ebpml.doc

OMG UML

- Unified Modeling Language Specification V2.1.2: Superstructure, OMG, Nov 2007,
  http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF

Open Nested Transactions

- Concepts and Applications of Multilevel Transactions and Open Nested Transactions, Gerhard Weikum, Hans-J. Schek, 1992
  http://citeseer.nj.nec.com/weikum92concepts.html

RDF

- RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft
  http://www.w3.org/TR/rdf-schema/

SOAP 1.2

- SOAP Version 1.2 Part 1: Messaging Framework, W3C Working Draft
  http://www.w3.org/TR/soap12-part1/

- SOAP Version 1.2 Part21: Adjuncts, W3C Working Draft
  http://www.w3.org/TR/soap12-part2/

UDDI

- Universal Description, Discovery and Integration, Ariba, IBM and Microsoft, UDDI.org
  http://www.uddi.org

URI

- Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, IETF RFC 2396, August 1998
  http://www.ietf.org/rfc/rfc2396.txt

WfMC Glossary

- Workflow Management Coalition Terminology and Glossary
  http://www.wfmc.org/wfmc-standards-framework.html

Web Services Transaction

- (WS-Transaction) 1.1, OASIS, 12 July 2007,
  http://www.oasis-open.org/committees/ws-tx/

Workflow Patterns

- Russell, N., terHofstede, A.H.M., van der Aalst W.M.P, &Mulyar, N. (2006). Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcentre.org
  http://www.workflowpatterns.com/

WSBPEL

- Web Services Business Process Execution Language (WSBPEL) 2.0, OASIS Standard, April 2007
  http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

WS-Coordination

- Web Services Coordination (WS-Coordination) 1.1, OASIS Standard, July 2007
  http://www.oasis-open.org/committees/ws-tx/

WSDL

- Web Services Description Language (WSDL) 2.0, W3C Proposed Recommendation, June 2007
  http://www.w3.org/TR/wsdl20/

WS-HumanTask

- Web Services Human Task (WS-HumanTask) 1.0, June 2007
  http://www.active-endpoints.com/active-bpel-for-people.htm
- http://www.adobe.com/devnet/livecycle/articles/bpel4people_overview.html
- http://dev2dev.bea.com/arch2arch/
- http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/
- http://www.oracle.com/technology/tech/standards/bpel4people/
- https://www.sdn.sap.com/irj/sdn/bpel4people

XML 1.0 (Second Edition)

- Extensible Markup Language (XML) 1.0, Second Edition, Tim Bray et al., eds., W3C, 6 October 2000
  http://www.w3.org/TR/REC-xml

XML-Namespaces

- Namespaces in XML, Tim Bray et al., eds., W3C, 14 January 1999
  http://www.w3.org/TR/REC-xml-names

XML-Schema

- XML Schema Part 1: Structures, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, W3C, 2 May 2001
  http://www.w3.org/TR/xmlschema-1//
- XML Schema Part 2: Datatypes, Paul V. Biron and Ashok Malhotra, eds., W3C, 2 May 2001
  http://www.w3.org/TR/xmlschema-2/

XPath

- XML Path Language (XPath) 1.0, James Clark and Steve DeRose, eds., W3C, 16 November 1999
  http://www.w3.org/TR/xpath

XPDL

- Workflow Management Coalition XML Process Definition Language, version 2.0.
  http://www.wfmc.org/wfmc-standards-framework.html

4. Термины и определения

См. приложение C «Словарь».

5. Символы

Данная спецификация не содержит описания символов.

6. Дополнительная информация

6.1. Условные обозначения

Данный подраздел содержит информацию об условных обозначениях, используемых в данном документе. Сюда входят условные обозначения нотации (текста) и нотация компонентов схемы. Сюда также входят определения обозначенного пространства имен.

6.1.1. Типографские и лингвистические знаки и стили

В данной спецификации описаны следующие обозначения:

- Интерпретация ключевых слов «ДОЛЖЕН» («MUST»), «НЕДОЛЖЕН» («MUSTNOT»), «НЕОБХОДИМ» («REQUIRED»), «ДОЛЖЕН»(«SHALL»), «НЕДОЛЖЕН» («MUSTNOT»), «СЛЕДУЕТ» («SHOULD»), «НЕ СЛЕДУЕТ» («SHOULDNOT»), «РЕКОМЕНДУЕТСЯ» («RECOMMENDED»), «МОЖЕТ» («MAY»)и «ПО ЖЕЛАНИЮ» («OPTIONAL») должна быть выполнена в согласии с RFC-2119.
- **Термин** – это слово или фраза с особым значением. Когда указан какой-либо термин, его название выделяется жирным шрифтом.

- Ссылка на другое определение, раздел, спецификацию отображается подчеркнутой и представляет собой действующую ссылку на необходимую область данного документа.
- Ссылка на графический элемент начинается с заглавной буквы, выделяется жирным и отображается шрифтом **Arial** (например, **Подпроцесс**).
- Ссылка на не являющийся графическим элемент или понятие **BPMN** выделяется курсивом и отображается шрифтом *TimesNewRoman* (например, *токен*).
- Ссылка на атрибут или ассоциацию отображается шрифтом `CourierNew` (например, `Expression`).
- Ссылка на элемент, атрибут или конструкцию WSBPEL пишется курсивными строчными буквами, обычно с предшествующим словом «WSBPEL», и отображается шрифтом *`CourierNew`* (например, WSBPEL*`pick`*).
- Ненормативные примеры расположены отдельно и сопровождаются кратким описанием. Код XML и псевдокод выделяются моноширинным шрифтом. Разные цвета шрифта МОГУТ быть использованы для выделения различных компонентов кода XML.
- Количество элементов определяется посредством следующих команд:
  - <none> — только один раз
  - [0..1] — 0 или 1
  - [0..*] — от нуля и более
  - [1..*] — от одного и более
- Атрибуты, разделенные знаком «|» и сгруппированные посредством {and}, являются альтернативными значениями.
  - <value> — значение по умолчанию
  - <type> — тип атрибута

6.1.2. Аббревиатуры

В документе использованы следующие аббревиатуры:

Аббревиатура	Расшифровка
WSBPEL	Web Services Business Process Execution Language (см.WSBPEL).Данная аббревиатура относится к версии 2.0 данной спецификации.
WSDL	Web Service Description Language (scмWSDL).Данная аббревиатура относится кW3CTechnicalNote (от15 марта 2001г), однако, она используется и для поддержки последующих спецификаций WSDL.

6.2. Структура документа

В Главе 7 данного документа обсуждаются рамки спецификации, предоставляется краткое описание элементов, которым соответствуют подразделы документа.

Глава 8 знакомит читателей с Ядром **BPMN**, включающим основные элементы **BPMN**. Эти элементы необходимы для моделирования **Бизнес-процессов**, включая **Взаимодействие**, **Процессы** *оркестровки* и **Хореографию.**

Элементы, необходимые для моделирования **Взаимодействи**я, **Процесса** *оркестровки*, **Обмена сообщениями** и **Хореографии** описаны в Главах 9, 10, 11 и 12 соответственно.

Глава 13 содержит визуальное представление диаграмм моделей **BPMN**. В Главе 14 представлено описание семантик *оркестровки* **Процесса** в **BPMN2.0.** В Главе 15 обсуждается соответствие **BPMN** языку WS-BPEL, который был выведен путем анализа объектов **BPMN** и отношений между ними. Обмен форматами и XSLT трансформация этих форматов описаны в Главе 16.

6.3. Благодарность

Сотрудничающие Организации (RFP Process)

Следующие компании формально сотрудничают с группой OMG:

- Axway
- International Business Machines
- MEGA International
- Oracle
- SAP AG
- Unisys

Поддерживающие Организации(RFP Process)

Следующие организации поддерживают использование данной спецификации, однако, не сотрудничают с группой OMG формально:

- Accenture
- Adaptive
- BizAgi
- Bruce Silver Associates
- Capgemini
- Enterprise Agility
- France Telecom
- IDS Scheer
- Intalio
- Metastorm
- Model Driven Solutions
- Nortel
- Red Hat Software
- Software AG
- TIBCO Software
- Vangent

Члены оперативной группы с правом голоса

- Adaptive
- Axway Software
- BAE SYSTEMS
- BizAgi Ltd.
- CA Inc.
- Camunda Services GmbH
- Cordys
- DICOM
- France Telecom R&D
- Fujitsu
- Global 360, Inc.
- Hewlett-Packard
- iGrafx
- Inferware
- Intalio
- International Business Machines
- KnowGravity Inc.

- Lombardi Software
- MITRE
- U.S. National Institute of Standards and Technology
- No Magic, Inc.
- oose Innovative Informatik GmbH
- Oracle
- PNA Group
- Red Hat
- SAP AG
- Softeam
- Software AG Inc.
- TIBCO
- Trisotech
- Visumpoint

Особая благодарность

Особая благодарность выражается членам основной команды, работавшим над содержанием данного документа:Anurag Aggarwal, Mike Amend, Sylvain Astier, Alistair Barros, Rob Bartel, Mariano Benitez, Conrad Bock, Gary Brown, Justin Brunt, John Bulles, Martin Chapman, Fred Cummins, Rouven Day, MagedElaasar, David Frankel, Denis Gagné, John Hall, Reiner Hille-Doering, Dave Ings, Pablo Irassar, Oliver Kieselbach, Matthias Kloppmann, Jana Koehler, Frank Michael Kraft, Tammo van Lessen, Frank Leymann, Antoine Lonjon, Sumeet Malhotra, FalkoMenge, Jeff Mischkinsky, Dale Moberg, Alex Moffat, Ralf Mueller, SjirNijssen, KarstenPloesser, Pete Rivett, Michael Rowley, Bernd Ruecker, Tom Rutt, Suzette Samoojh, Robert Shapiro, Vishal Saxena, Scott Schanel, Axel Scheithauer, Bruce Silver, MeeraSrinivasan, Antoine Toulme, IvanaTrickovic, Hagen Voelzer, Franz Weber, Andrea Westerinen and Stephen A. White.

Следующие люди также привнесли ценные идеи и обеспечили обратную связь, что помогло усовершенствовать содержимое и качество данного документа: imAmsden, Mariano Belaunde, Peter Carlson, Cory Casanave, Michele Chinosi, Manoj Das, Robert Lario, SumeetMalhotra, Henk de Man, David Marston, Neal McWhorter, Edita Mileviciene, VadimPevzner, Pete Rivett, Jesus Sanchez, Markus Schacher, Sebastian Stein, and Prasad Yendluri.

7. Общее представление

За короткий промежуток времени был сделан большой скачок в разработке языков на основе XML для реализации **Бизнес-процессов** для Web-сервисов. Такие языки, как WSBPEL (Web Services Business Process Execution Language), служат для формального описания **Бизнес-Процессов**. Отличительной особенностью языков такого формата является то, что они были оптимизированы для описания **Процессов** внутри BPM-систем и их взаимодействия с другими. Оптимизация этих языков для приложений сделала их менее удобными для использования людьми (включая разработку **Бизнес-процессов**, управление ими, мониторинг). Для описания **Бизнес-процессов** WSBPEL использует блоки и диаграммы. В нем применены принципы формальных математических моделей (например, pi-calculus1). Все это способствовало формированию основ выполнения **Бизнес-процессов** для обработки комплексных внутренних и внешних B2B взаимоотношений, а также выгодного использования Web-сервисов. Для WSBPEL характерно то, что сложные **Бизнес-процессы** могут быть организованы в формате потенциально сложных, разрозненных и интуитивно непонятных моделей, которые с легкостью обрабатываются программами и приложениями, однако, сложны для понимания бизнес-аналитиками и менеджерами, задачами которых являются разработка, управление и мониторинг **Процессов**. Поэтому языки на основе XML для реализации **Бизнес-процессов** для Web-сервисов не ориентированы на создание удобства использования и способности к взаимодействию на уровне людей.

Людям, занимающимся бизнесом, крайне удобно работать с **Бизнес-процессами**, отображаемыми в виде блок-схем. Множество бизнес-аналитиков проектируют и описывают **Бизнес-процессы** компаний с помощью простых блок-схем, вследствие чего возникла нерешенная техническая проблема, заключающаяся в различии форматов исходных проектов **Бизнес-процессов** и языков исполнения **Бизнес-процессов** (WSBPEL и др.).

Для решения этой проблемы потребовалось создание формального механизма соответствия визуализации **Бизнес-процессов** (нотации) необходимому языку исполнения **Бизнес-процессов** (BPM execution language).

Взаимодействие людей (а не операций приложения) в **Бизнес-процессах** могло быть решено благодаря стандарту **BPMN** (**BusinessProcessModelandNotation**). Использование **BPMN** позволяет создавать множество диаграмм для использования людьми, разрабатывающими **Бизнес-процессы** и управляющими ими. Благодаря **BPMN**, также осуществляется и соотнесение модели **Бизнес-процесса** с языком исполнения (WSBPEL). Таким образом, создание **BPMN** обеспечило появление механизма стандартной визуализации **Бизнес-процессов**, описанного с помощью языка исполнения оптимизированных **Бизнес-процессов**.

Стали понятными изображенные с помощью графической нотации внутренние процедуры компаний, а сами компании получили возможность доступа к стандартной работе с этими процедурами. На данный момент существует множество инструментов и методик моделирования **Бизнес-процессов**. Персонал переходит из одной компании в другую, сами компании объединяются и распадаются, - все это требует от бизнес-аналитика понимания полимасштабных представлений моделей **Бизнес-процессов**, т.е. потенциально разных моделей одного и того же **Процесса** на разных его стадиях (включая разработку, внедрение, выполнение, мониторинг и проведение анализа). Следовательно, использование стандартизированной графической нотации может облегчить понимание **Процесса Взаимодействия** и *транзакций* внутри одной компании или между взаимодействующими компаниями. Результатом этого является гарантированное взаимопонимание между участниками **Бизнес-процессов**, что поможет компаниям быстро подстраиваться под новые внутренние и внешние (B2B) обстоятельства. Для большей читабельности и гибкости в данной нотации продолжены традиции нотаций блок-схем. Семантика исполнения, описанная в ней, полностью формализована. Для создания нотации нового поколения, в которой сочетались бы читабельность, гибкость и способность к расширению, группа OMG использовала опыт использования других нотаций **Бизнес-процессов**, предшествующих **BPMN**.

Благодаря тому, что при разработке **BPMN** была использована B2B концепция построения **Бизнес-процессов**, были расширены возможности нотации в отношении *публичных* и *приватных* **Процессов**, **Хореографии** (**Choreography**), а также *обработки ошибок*, *транзакции* и *компенсаций*.

7.1. Область применения BPMN

В данной спецификации рассматриваются нотация, варианты моделирования **Бизнес-процессов**, а также формат обмена данными, применение которых поможет пользователям успешно осуществлять использование терминов нотации **BPMN** (как в моделях, так и на диаграммах) и их замещение при работе с различными инструментами моделирования. Задача данного документа – показать гибкость использования одних и тех же объектов нотации в различных средах моделирования **Бизнес-процессов**.

В спецификации **BPMN 2.0** содержится более подробное, нежели содержащееся в спецификации **BPMN 1.2**, описание возможностей и областей использования нотации, а именно:

- Формализация семантики исполнения всех существующих элементов **BPMN**,
- Определение механизма изменения, как расширений модели **Бизнес-процесса**, так и для расширений графических элементов,
- Детализация состава и корреляции **События**,
- Расширение определения пользовательских действий,
- Введение элемента **Хореография** (**Choreography**)и описание данной модели.

Данная спецификация также разрешает некоторые несоответствия и неточности спецификации **BPMN 1.2**.

Нотация **BPM** заключает в себе концепцию моделирования, применяемую для **Бизнес-процессов**, что, соответственно, исключает рассмотрение других типов моделирования, осуществляемого различными организациями для ведения деловой деятельности. По этой причине в данной спецификации не затронуты следующие аспекты:

- Определение типов организационных структур и их ресурсов,
- Функциональное моделирование структуры организации,

- Создание моделей данных и информационных моделей,
- Моделирование стратегии,
- Созданиебизнес-правил.

Поскольку все вышеперечисленные аспекты высокоуровнего моделирования прямо или косвенно относятся к **Бизнес-процессам**, взаимосвязь нотации **BPMN** с другими типами высокоуровнего моделирования можно формально определить как **BPMN** и его взаимосвязь с другими спецификациями.

Несмотря на то, что **BPMN** описывает поток данных (**Сообщений**) и связь артефактов с **Действиями**, её нельзя назвать языком потока данных. Также в спецификацию не включена информация о выполнении операций в режиме симуляции деятельности, мониторинге и разворачивании **Бизнес-процесса**.

В **BPMN 2.0** можно найти соответствия с несколькими языками исполнения бизнес-процессов, таких как WS-BPEL 2.0. Данная спецификация содержит информацию о соответствии ряда параметров **BPMN** языку WS-BPEL 2.0. Поиск соответствий данной нотации другим стандартам требует дополнительного изучения.

Для определения типов данных, `выражений` и сервисных операций в данной спецификации также использованы другие стандарты: XMLSchema, XPath, и WSDL соответственно.

7.1.1. Использование BPMN

Моделирование **Бизнес-процессов** предназначено для описания взаимодействия между огромным количеством информации и множеством целевых групп. **BPMN** объединяет возможности различных типов моделирования, что позволяет создавать непрерывные (end-to-end) **Бизнес-процессы**. С помощью элементов **BPMN** пользователи могут легко отделять одну часть диаграммы от другой. Существует три основных типа компонентов (sub-model) модели бизнес-процесса end-to-end:

1. **Процесс** (Оркестровка), включающий:
    a. *Приватные невыполняемые (внутренние)* **Бизнес-процессы***(Private non-executable (internal)* **Business Processes**).
    b. *Приватные выполняемые (внутренние)* **Бизнес-процессы** *(Private executable (internal)* **Business Processes**).
    c. *Публичные* **Процессы** (*Public Processes*).
2. **Хореография (Choreography)**.
3. **Взаимодействие (Collaborations)**, которое может содержать **Процессы** и/или **Хореографии**.
    a. Вид обмена **Сообщениями**.

Приватный (внутренний) Бизнес-процесс (Private (Internal) Business Processes)

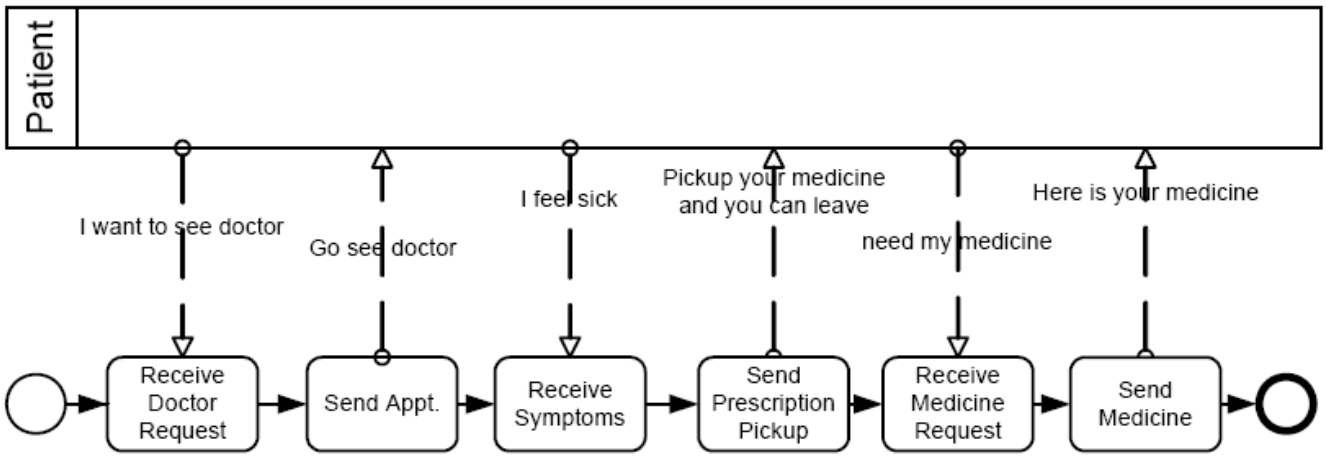
Приватные **Бизнес-процессы** (*Private (internal)* **Business-Process**) относятся к внутренним **Процессам** компании. Такие **Процессы** обычно называют **Процессами** потока операций или **Процессами** BPM (см. фигуру 10.4). Другое их название – *Оркестровка* сервисов (используется в веб-сервисах). *Приватные* **Процессы** могут быть *выполняемыми* и *невыполняемыми*. *Выполняемый* **Процесс** моделируется для исполнения в соответствии с семантикой, описанной в главе 14. Время от времени на определенных этапах выполнения **Процесса** может недоставать информации, необходимой для того, чтобы **Процесс** оставался *выполняемым*. *Невыполняемый* **Процесс** моделируется с целью описания поведения **Процесса** с точки зрения разработчика модели. Таким образом, информация, необходимая для выполнения **Процесса** (например, *условное выражение*), обычно не включается в *невыполняемый* **Процесс**.

В случае, если используются Зоны ответственности (например, **Взаимодействие**, см. ниже), то *приватный* **Бизнес-процесс** не выходит за рамки **Пула**, а Поток операций данного **Процесса**, содержащийся в этом же **Пуле**, не пересекает его границ. Поток **Сообщений**, однако, может выходить за рамки **Пула** для того, чтобы отобразить взаимодействие между отдельно взятыми *приватными* **Бизнес-процессами**.

Фигура 7.1 – Пример приватного Бизнес-процесса

Публичный Процесс

Публичный **Процесс** (*PublicProcess*) служит для отображения взаимодействия между *приватным* **Бизнес-процессом** и другим **Процессом** или Участником (см. фигуру 7.2). В его состав входят лишь те **Действия**, которые обычно используются для отображения сообщения с другими *Участниками*. Все остальные «внутренние» **Действия** *приватного* **Бизнес-процесса** не входят в *публичный* **Процесс**. Таким образом, посредством *публичного* **Процесса** внешний мир может наблюдать за **Потоками сообщений** и порядком, в котором эти **Потоки сообщений** должны взаимодействовать с **Процессом**. *Публичный* **Процесс** может моделироваться как отдельно от **Взаимодействия**, так и с ним, что помогает показать обмен **Сообщениями** между **Действиями** *публичного* **Процесса** и другими *Участниками*. Обратите внимание, что в **BPMN 1.2** *публичный* **Процесс** назван *абстрактным*.

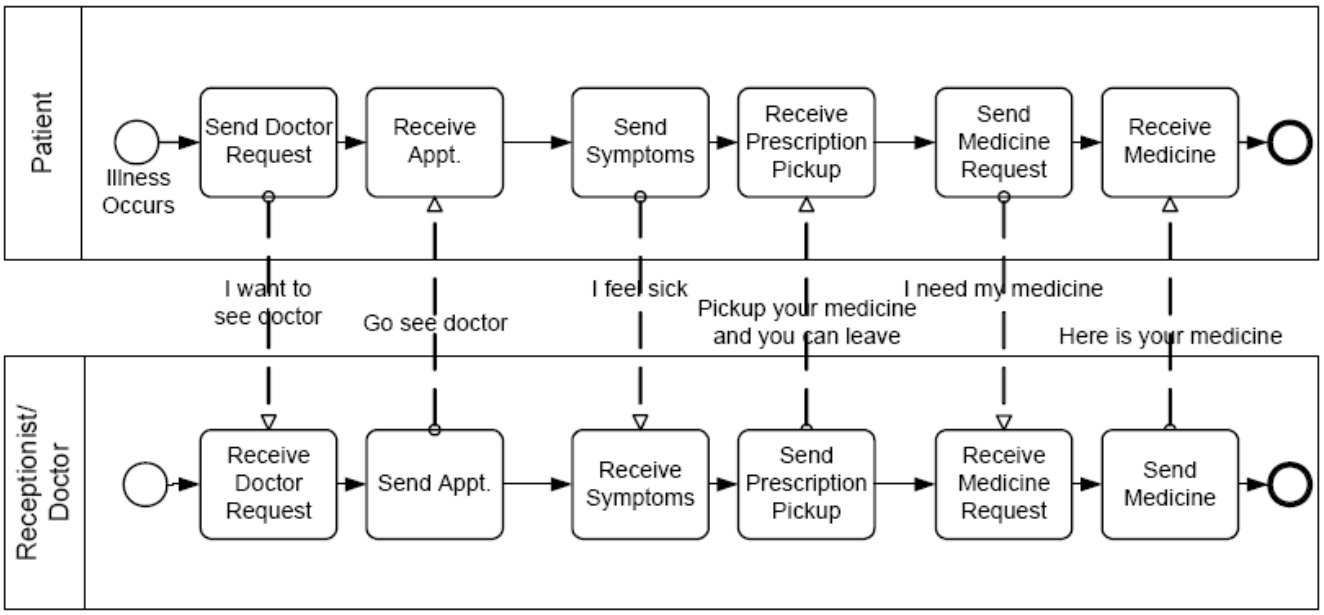


Фигура 7.2 – Пример публичного Процесса

Взаимодействие

С помощью **Взаимодействия (Collaboration)**отображаются взаимоотношения между двумя или более бизнес-сущностями. Обычно в состав **Взаимодействия** входят две или более **Зоны ответственности**, представляющие собой *Участников* **Взаимодействия**. Обмен **Сообщениями** между этими *Участниками* отображается при помощи **Потока сообщений**, соединяющих два **Пула** или объекты в них. Также отображаются и **Сообщения**, входящие в состав данного **Потока сообщений. Взаимодействие** может отображаться в виде двух или более сообщающихся между собой *публичных* **Процессов** (см. фигуру 7.3). *Публичные* **Процессы** и **Действия**, предназначенные для выполнения *Участниками* **Взаимодействия**, могут рассматриваться в качестве точек соприкосновения (touch-points). Соответствующие внутренние (выполняемые) **Процессы** содержат, как правило, намного больше **Действий** и информации, чем публичные **Процессы.** Также **Пул** МОЖЕТ представлять собой черный ящик (blackbox) или, другими словами, быть пустым. **Хореография** МОЖЕТ отображаться в пространстве между **Пулами**, поскольку она разделяет пополам

соединяющие их **Потоки операций**. Во **Взаимодействии** допускается использование любых комбинаций **Пулов**, **Процессов** и **Хореографий**.

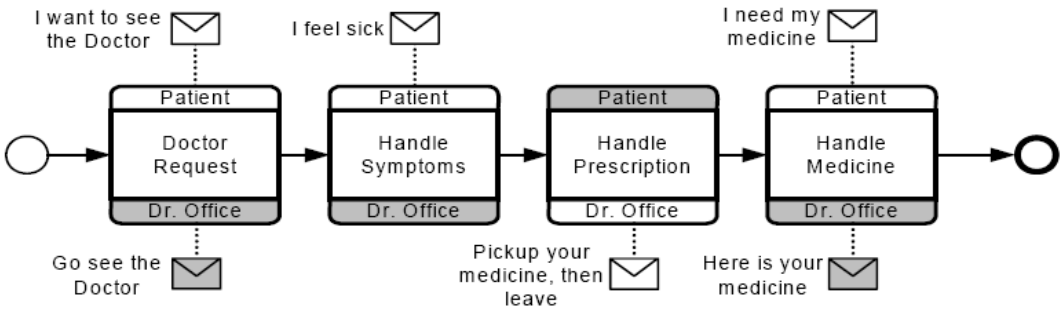


Фигура 7.3 – Пример Процесса Взаимодействия

Хореография (Choreography)

Стандартный **Процесс** существует внутри **Зоны ответственности**, а **Хореография** осуществляется между **Зонами ответственности**.

Хореография (Choreography) похожа на *приватный* **Бизнес-процесс**, т.к. состоит из последовательности **Действий**, **Событий** и **Шлюзов** (см. фигуру 7.4). Отличие **Хореографии** заключается в том, что под **Действиями** в ней подразумеваются взаимоотношения, представляющие собой обмен одним или более **Сообщениями** между двумя или более *Участниками.* Ещё одним отличием **Хореографии** от стандартного **Процесса** является то, что в ней нет центрального контроллера **Процесса**, ответственной за него сущности или наблюдателя.

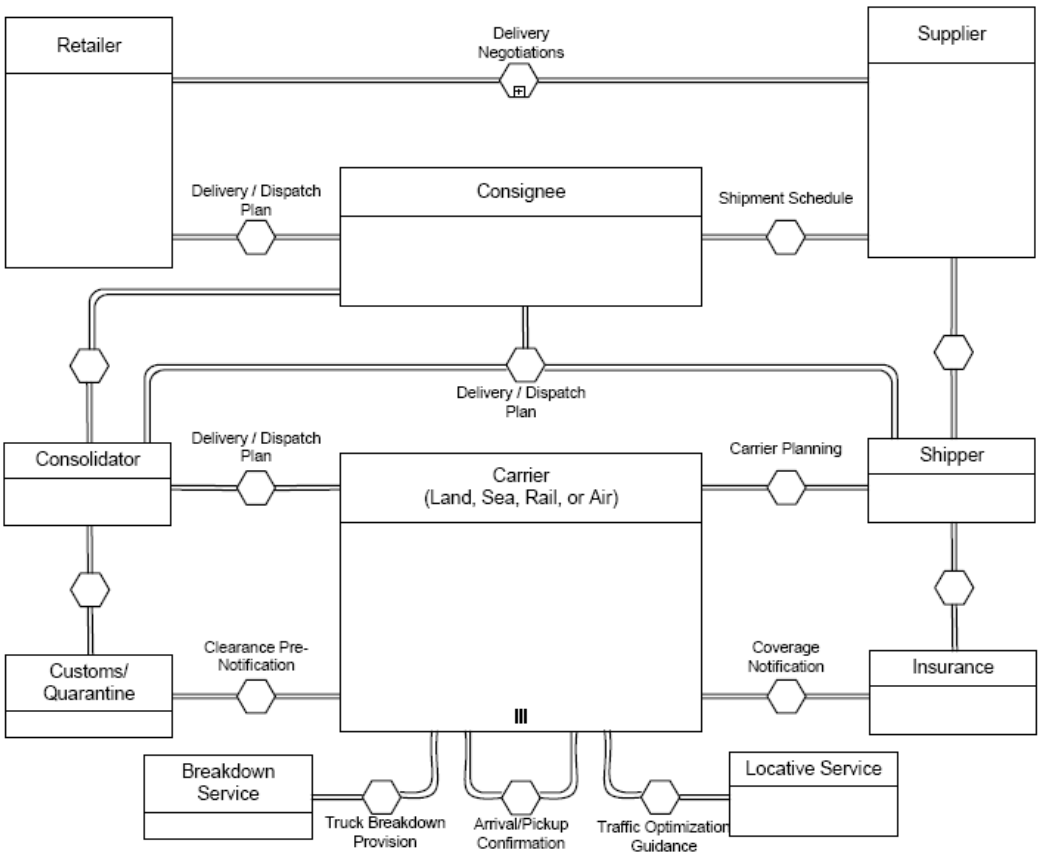


Фигура 7.4 – пример Хореографии

Обмен сообщениями (Conversations)

Диаграмма **Обмена сообщениями (Conversation)** является частным случаем использования диаграммы **Взаимодействия** и его неформального описания. При этом, однако, **Пулы**, входящие в **Обмен сообщениями**, обычно не содержат **Процессов**, а **Хореография (Choreography)**, как правило, не располагается между **Пулами** диаграммы **Обмена сообщениями**. **Обмен сообщениями** представляет собой логическое отношение типа обмен **Сообщениями**. На деле логическое отношение часто касается бизнес-объектов коммерческих объединений (к примеру, заказ товаров, их перевозка и доставка, выписка счета)

Компоненты **Обмен сообщениями** связаны друг с другом и отражают определенные бизнес-сценарии. Рассмотрим пример из логистики. Процесс пополнения запасов товаров включает следующие типичные сценарии: создание заказа, назначение перевозчика товара (сюда входят различные заказы на покупку), карантин, оплата, отвод. Таким образом, на диаграмме **Обмена сообщениями** (например, такой, что отображена на фигуре 7.5) шестиугольниками показан **обмен сообщениями** между *Участниками* (**Пулами**). Это обеспечивает так называемый «вид с высоты птичьего полета» различных **Обменов сообщениями**, относящихся к одной области.



Фигура 7.5 – Пример диаграммы Обмена сообщениями

Диаграмма Точек зрения

Поскольку при помощи диаграммы **BPMN** МОЖЕТ изображаться **Процесс** с несколькими Участниками, то каждый из участников может по-разному понимать диаграмму, т.е. у каждого из них есть собственная точка зрения по поводу того, каким образом они будут участвовать в **Процессе**. Некоторые из **Действий** будут для кого-то из *Участников* внутренними (т.е. будут выполняться под контролем этого *Участника*), а некоторые – внешними. Любой *Участник* по-своему воспринимает то, какими должны быть внутренние и внешние **Действия**. Во время выполнения **Процесса** разница между внешними и внутренними **Действиями** влияет на то, каким образом данный *Участник* определяет статус какого-то из **Действий** или выявляет проблему. Однако

сама диаграмма всегда означает одно и то же. На фигуре 7.3 изображен **Бизнес-процесс**, который может рассматриваться разными *Участниками* по-разному. С одной стороны, есть Пациент (Patient), с другой стороны – офис Доктора (Doctor'soffice). На диаграмме показаны **Действия** обоих *Участников* **Процесса**, однако, когда **Процесс** выполняется, каждый из *Участников* контролирует выполнение лишь собственных **Действий.** Несмотря на то, что для каждого *Участника* важна его точка зрения на **Процесс**, на данный момент в **BPMN** нет никакого графического механизма отображения точек зрения. Разработчику модели или производителю инструмента моделирования предоставляется возможность вводить собственные графически отображаемые реплики о точках зрения на диаграмму.

Понимание поведения диаграммы

На протяжении данного документа обсуждается то, каким образом в **Процессе** задействован **Поток операций.** Для облегчения восприятия материала вводится элемент «токен». *Токен* пересекает **Поток операций** и проходит сквозь элементы **Процесса.** *Токен* является теоритическим понятием и используется для определения поведения выполняемого **Процесса**. Поведение элементов **Процесса** определяется путем описания их взаимодействия с *токеном*, который пересекает **Процесс**. Однако для инструментов моделирования и исполнения, работающих с **BPMN**, использование *токенов* НЕ является ОБЯЗАТЕЛЬНЫМ условием.

Стартовое событие формирует *токен*, который в итоге ДОЛЖЕН завершиться **Конечным Событием** (**Конечное событие** МОЖЕТ БЫТЬ скрытым в случае, если оно не отображается на диаграмме). Маршрут *токена* должен легко отслеживаться на протяжении всей диаграммы **Процесса**, содержащей **Потоки операций**, **Шлюзы** и **Действия.**

Примечание: *токен* не пересекает **Поток сообщений**, поскольку его пересекают **Сообщения** (ясно из названия).

7.2. Элементы BPMN

Важно отметить, что одной из причин создания **BPMN** явилась необходимость построения простого механизма для проектирования как простых, так и сложных моделей **Бизнес-процессов**. Для удовлетворения двух этих противоречащих требований был применен подход систематизации графических элементов нотации по категориям. Результатом явился небольшой перечень категорий нотаций, позволивший людям, работающим с диаграммами **BPMN**, без труда распознавать основные типы элементов и осуществлять корректное чтение схем. Основные категории элементов допускают внутренние вариации,а также добавление информации для удовлетворения требований сложности без внесения значительных изменений в общую структуру диаграммы для легкости её понимания.

Существуют пять основных категорий элементов:

1. Элементы потока (Flow Objects);
2. Данные (Data)
3. Соединяющие элементы (ConnectingObjects);
4. Зоны ответственности (Swimlanes);
5. Артефакты (Artifacts).

Элементы потока являются важнейшими графическими элементами, определяющими ход

Бизнес-процесса. *Элементы потока*, в свою очередь, делятся на:

1. События (Events);
2. Действия (Activities);
3. Шлюзы (Gateways).

Данные на диаграмме могут быть представлены любыми из следующих четырех элементов:

1. Объект данных (Data Objects)
2. Входныеданные (Data Inputs)
3. Выходныеданные (Data Outputs)
4. Хранилища данных (Data Stores)

Выделяют четыре вида соединяющих Элементов потока, связывающихся друг с другом и сдругими элементами:

1. Поток операций (Sequence Flow);
2. Поток сообщений (Message Flow);
3. Ассоциация (Association);
4. Ассоциацияданных (Data Associations).

Существуют два способа группировки основных элементов моделирования с помощью

Зон ответственности:

1. Группировка с помощью Пула (Pool);
2. Группировка с помощью Дорожки (Lane).

Артефакты используются для добавления дополнительной информации о Процессе. Выделяют два типовых Артефакта, что, однако, не запрещает разработчикам моделей **Бизнес-процессов** либо программам моделирования добавлять любое необходимое количество Артефактов. Для широкого круга пользователей, а также для вертикальных рынковсуществует возможность стандартизации более полного перечня Артефактов. На данный момент текущий перечень Артефактов включает в себя следующие элементы:

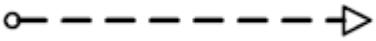
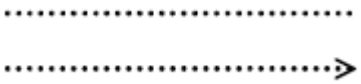
1. Группа (Gruop);
2. Текстовая аннотация (Text Annotation).

7.2.1. Основные графические элементы моделирования

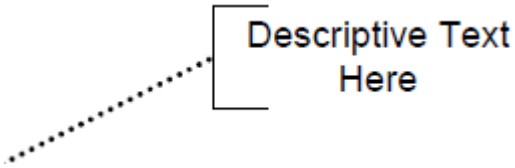
Таблица 7.1 содержит перечень основных графических элементов моделирования, изображенных при помощи графических нотаций.

Таблица 7.1 - Основные элементы моделирования

Элемент	Описание	Нотация
Событие (Event)	Событие – это то, что происходит в течение бизнес-процесса или его Хореографии. Событие оказывает влияние на ход бизнес-процесса и чаще всего имеет причину (триггер) или воздействие (результат). Изображается в виде круга со свободным центром, предназначенным для дифференцировки внутренними маркерами различных триггеров или их результатов. Согласно влиянию Событий на ход бизнес-процесса, выделяют три типа: Стартовое событие (Start), Промежуточное событие (Intermediate) и Конечное событие (End).	
Действие (Activity)	Действие – общий термин, обозначающий работу, выполняемую	

	исполнителем в ходе бизнес-процесса. Действия могут быть либо элементарными, либо неэлементарными (составными). Выделяют следующие виды действий, являющихся частью модели Процесса: Подпроцесс (Sub-Process) и Задача (Task). И Задача, и Подпроцесс изображаются в виде прямоугольников с закругленными углами. Все Действия могут являться элементами, как стандартных Процессов, так и Хореографий.	
Шлюз (Gateway)	Шлюзы используются для контроля расхождений и схождений Потока операций, как в Процессе, так и в Хореографии. Таким образом, данныйтермин подразумевает ветвление, раздвоение, слияние и соединение маршрутов. Внутренние маркеры указывают тип контроля развития бизнес-процесса.	
Поток операций (Sequence Flow)	Поток операций служит для отображения того порядка, в котором организованы действия Процесса или условия Хореографии.	
Поток сообщений (Message Flow)	Поток сообщений служит для отображения обмена сообщениями между двумя участниками, готовыми эти сообщения отсылать и принимать. На диаграмме взаимодействия BPMN два отдельно взятых Пула представляют собой двух участников Процесса (бизнес-сущности или бизнес-роли).	
Ассоциация (Association)	Ассоциация служит для установления связи между информацией или Артефактами (объектами, не относящимися к Элементампотока) и элементами потока. Текстовые объекты, а также графические объекты, не относящиеся к элементам потока, могут соотноситься с элементами потока. При необходимости Ассоциация может указывать направление потока (например, потока данных).	
Пул(Pool)	Пул представляет собой Участника Взаимодействия. Пул также может выступать в качестве Зоны ответственности или	

	графического контейнера, отвечающего за разделение определенного набора действий, относящихся к другим Пулам, что обычно встречается в ситуациях типа «бизнес для бизнеса» (B2B). Внутри Пула МОЖЕТ находиться дополнительная информация по выполняемому Процессу. В случае, если такой информации в Пуле не содержится, то он МОЖЕТ представлять собой «черный ящик».	
Дорожка (Lane)	Дорожка используется для отображения распределения ролей и может быть как вертикальной, так и горизонтальной (также может использоваться для разделения внутреннего пространства Пула). Служит для упорядочивания и категоризации Действий.	
Объект данных (Data Object)	Объект данных предоставляет информацию о том, какие действия необходимо выполнить и/или каков результат этих действий. Может изображаться как в единственном экземпляре, так и в нескольких. Входные и Выходные данные Объекта данных представляют собой одну и ту же информацию о Процессе.	
Сообщение (Message)	Сообщение используется для отображения сущности взаимодействия между двумя Участниками бизнес-процесса (Участники определяются командами business PartnerRole или business PartnerEntity).	
Группа(блок, содержащийгруппу объектов одной категории) (Group)	Группа предназначена для группировки графических элементов, принадлежащих одной и той же категории. Такая группировка не оказывает влияния на Поток операций. На диаграмме бизнес-процесса название категории, к которой принадлежат сгруппированные элементы, отображается в качестве названия группы. Такого рода группировка может использоваться в целях составления документации или при проведении анализа. Графически Группы отображаются так же, как и Категории объектов.	

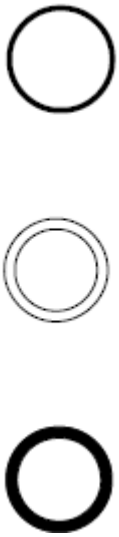
		
Текстовая аннотация (связана с Ассоциацией) (Text Annotation)	Текстовые аннотации являются механизмом, позволяющим разработчику модели бизнес-процесса вводить дополнительную информацию для тех, кто работает с BPMN диаграммами.	

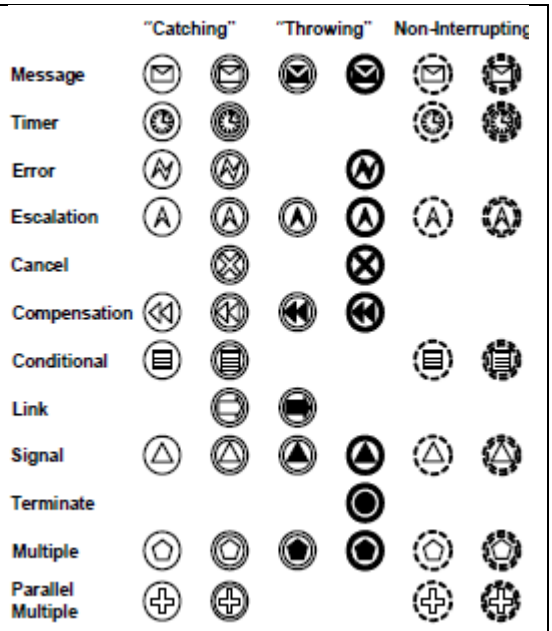
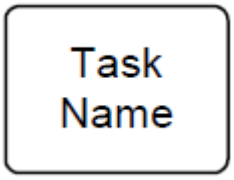
7.2.2. Полный перечень графических элементов диаграмм бизнес-процессов

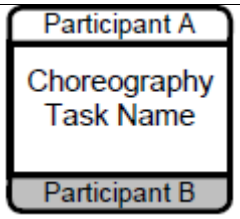
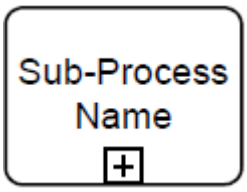
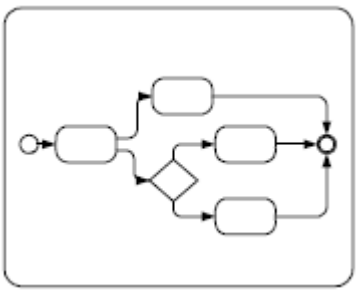
Таблица 7.2 содержит более полный перечень основных графических элементов моделирования **Бизнес-процессов**, изображенных при помощи графических нотаций.

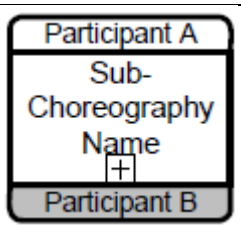
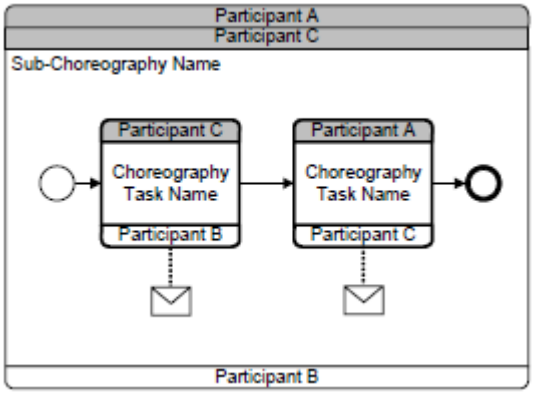
Таблица 7.2 –Полный перечень элементов моделирования BPMN

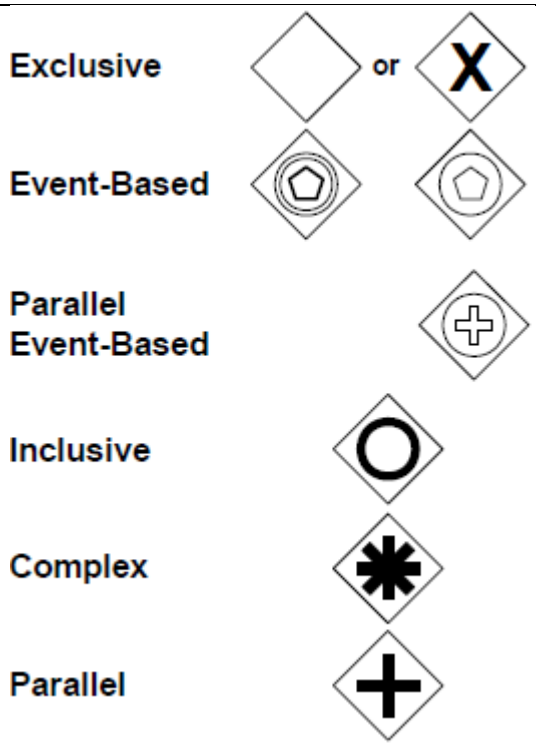
Элемент	Описание	Нотация
Событие (Event)	Событие – это то, что происходит в течение бизнес-процесса или его Хореографии. Событие оказывает влияние на ход бизнес-процесса и чаще всего имеет причину (триггер) или воздействие (результат). Изображается в виде круга со свободным центром, предназначенным для дифференцировки внутренними маркерами различных триггеров или их результатов. Согласно влиянию Событий на ход бизнес-процесса, выделяют три типа: Стартовое событие (Start), Промежуточное событие (Intermediate) и Конечное событие (End).	
Состав потока (Flow Dimension) (например, Стартовое событие, Промежуточное		

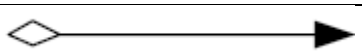
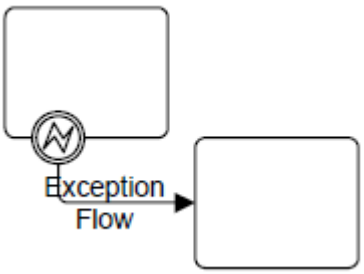
событие, Конечное событие) Стартовое событие Промежуточное событие Конечное событие	Как видно из названия, Стартовое событие указывает на то, в какой точке берет начало тот или иной Процесс или Хореография(**Choreography**). Промежуточное событие происходит на отрезке,ограниченном Стартовым и Конечным Событиями. Промежуточное событие оказывает влияние на ход Процесса или Хореографию, однако, не может являться началом или непосредственным завершением Процесса. Как видно из названия, Конечное событие указывает на то, в какой точке завершится тот или иной Процесс или Хореография.	
Тип (Type Dimension) (например, Неопределенный, Сообщение, Таймер, Ошибка, Отмена, Компенсация, Условие, Связь, Сигнал, Множественный, Завершение)	Стартовые и некоторые Промежуточные события имеют триггеры, определяющие причины происхождения Событий данных типов (см. разделы Стартовое событие и Промежуточное событие далее по тексту). Существует множество причин, инициирующих появление События. Конечные события МОГУТ определять результат, являющийся следствием окончания Потока операций. В отличие от Стартового события, которое лишь обрабатывает триггер, Промежуточное может, как обрабатывать триггеры, так и возбуждать их. Конечное событие лишь определяет результат (инициирует триггер). Маркеры Событий, обрабатывающих триггеры, отображаются без заливки, в то время как маркеры инициирующих триггеры Событий закрашены.	

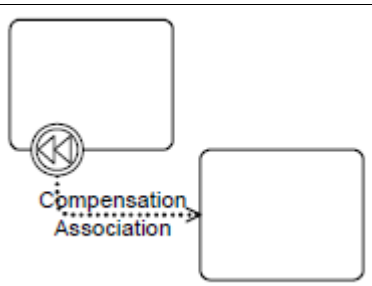
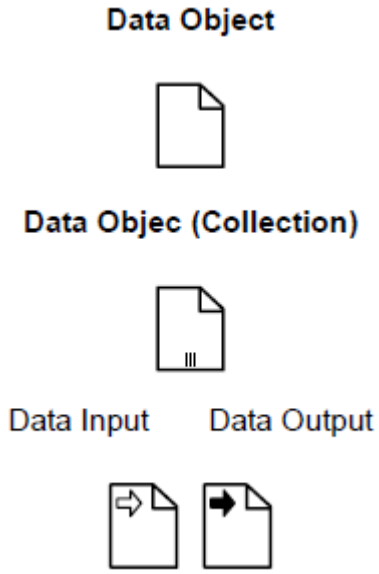
	Кроме того, некоторые типы Событий, используемые в BPMN 1.1 для прерывания хода Действия, в данной редакции могут использоваться для других целей. Такое Событие изображается в виде круга с пунктирными границами (см. ряд Событий справа).	
Действие (Activity)	Действие – общий термин, обозначающий работу, выполняемую исполнителем в ходе бизнес-процесса. Действия могут быть либо элементарными, либо неэлементарными (составными). Выделяют следующие виды действий, являющихся частью модели Процесса: Подпроцесс (Sub-Process) и Задача (Task). И Задача, и Подпроцесс изображаются в виде прямоугольников с закругленными углами. Все Действия могут являться элементами, как стандартных Процессов, так и Хореографий.	
Задача (элементарное действие) (Task)	Задача представляет собой элементарное действие, включенное в состав Процесса. Используется в случае, если Процесс не детализируется далее в данной Модели.	
Задача Хореографии	Задача Хореографии представляет собой элементарное действие в составе Хореографии. Отображает	

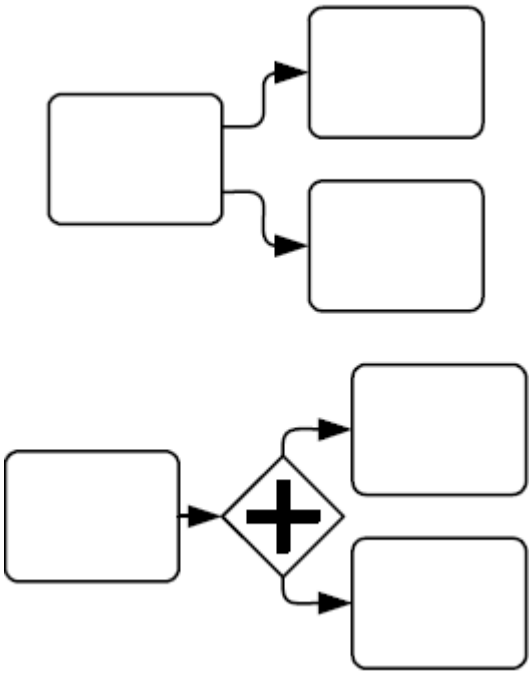
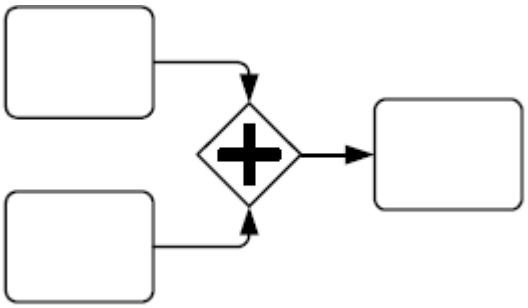
(Choreography Task)	один или несколько случаев обмена сообщениями и подразумевает наличие как минимум двух Участников. Название Задачи Хореографии и имена Участников отображаются в трех разных частях данного графического элемента. Таким образом, графически Задача Хореографии должна быть разделена на дорожки с именами участников (две или более), а также содержать дорожку, предназначенную для названия данной Задачи.	
Процесс/Подпроцесс (неэлементарное действие) (Process/Sub-Process)	Подпроцесс представляет собой комплексное Действие, включенное в состав Процесса. Такой вид действия считается составным, т.к. может быть разбит на составляющие (Процесс, Хореография (**Choreography**)) благодаря использованию поддействий (sub-Activities).	См. следующие 4 фигуры
Свернутый Подпроцесс (Collapsed Sub-Process)	Диаграмма не отображает детали Подпроцесса. Знак «плюс» находится в центре нижней части фигуры, символизирующей Подпроцесс, и указывает на то, что данное действие является Подпроцессом. В данном случае детали Процесса находятся на нижнем уровне.	
Развернутый Подпроцесс (Expanded Sub-Process)	Границы Подпроцесса расширены. Внутри границ просматриваются детали. Важно отметить, что Поток операций не может пересекать границ Подпроцесса.	
Скрытая Подхореография (Collapsed Sub-Choreography)	Диаграмма не отображает детали Подхореографии. Знак «плюс» находится в центре нижней части дорожки с названием Задачи и указывает на то, что данное Действие является Подпроцессом. В данном случае детали Хореографии	

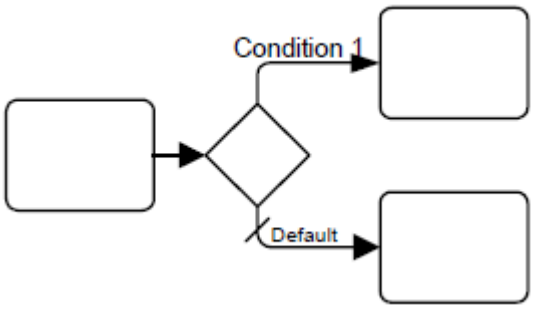
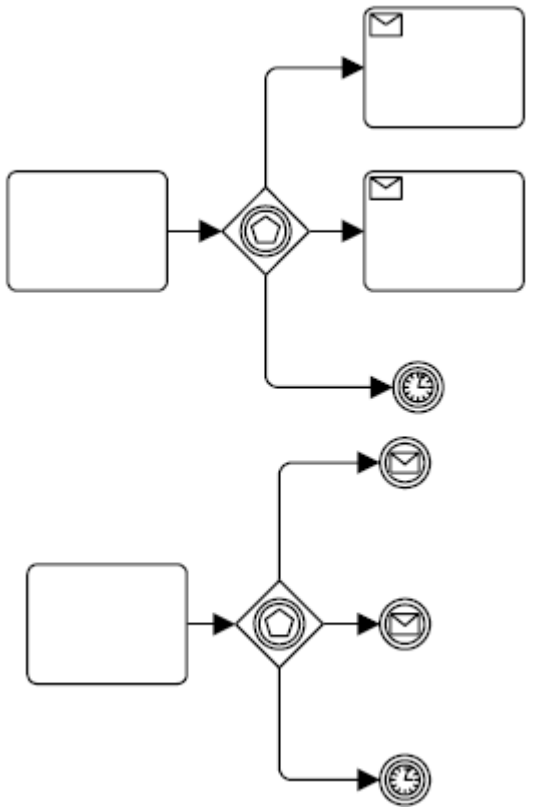
	находятся на нижнем уровне.	
Развернутая Подхореография (Expanded Sub-Choreography)-	Границы Подхореографии расширены. Внутри границ просматриваются детали. Важно отметить, что Поток операций не может пересекать границ Подхореографии.	
Шлюз (Gateway)	Шлюзы используются для контроля расхождений и схождений множественных Потоков операций Процесса и Хореографии. Таким образом, данный термин подразумевает ветвление, раздвоение, слияние и соединение маршрутов. Могут содержать внутренние маркеры, предназначенные для дифференцировки направления потоков.	
Типы Шлюзов (Gateway Control Types)	Шлюзы - фигуры в виде ромба - влияют на потоки. Выделяют следующие типы Шлюзов: • Эксклюзивные условия и объединения. Могут быть исключающими и основываться на событиях. Данный тип Шлюзов может отображаться как с маркером «X», так и без него. • Шлюзы, основанные на Событиях, и Параллельные Шлюзы, основанные на Событиях, инициируют появление нового	

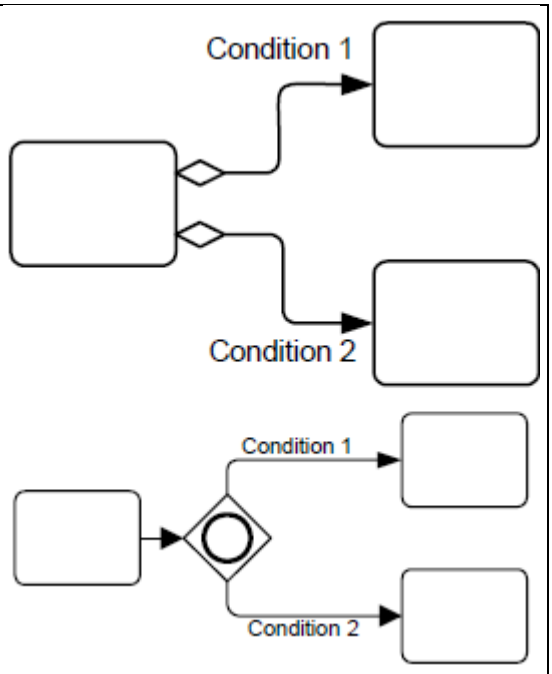
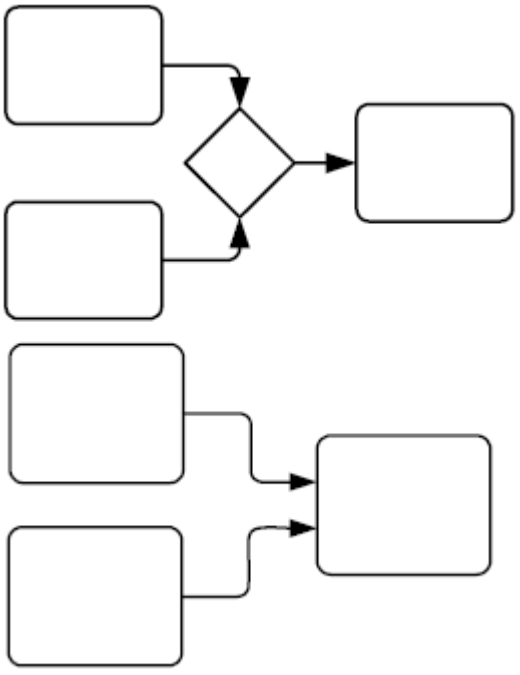
	экземпляра Процесса. • Включающие условия и объединения. • Комплексные Шлюзы, представляющие собой сложные условия и ситуации (например, 3 из 5). • Параллельные Шлюзы, представляющие собой раздвоение и слияние. Шлюзы каждого из типов оказывают влияние, как на входящие, так и на исходящие потоки.	
Поток операций (Sequence Flow)	Поток операций служит для отображения того порядка, в котором выполняются действия Процесса или Хореографии.	См. следующие 7 фигур
Стандартный поток операций (Normal Flow)	Стандартный поток операций относится к потокам, берущим начало от Стартового события и следующим по ходу выполнения Действий.	
Неконтролируемый поток операций (Uncontrolled Flow)	Неконтролируемый поток операций относится либо к потокам, на которые не воздействую никакие условия, либо к потокам, не проходящим через Шлюзы. Простейшими примерами Неконтролируемого потока операций могут послужить отдельно взятый Поток операций, объединяющий два Действия, или составной Поток операций, сходящийся в Действии или расходящийся от него. Для каждого Неконтролируемого потока операций возникает «токен», проходящий от ресурсного объекта до целевого.	
Условный поток операций (Conditional Flow)	Поток операций может зависеть от условных выражений, оценивающихся согласно времени выполнения для того, чтобы определить, будет ли использоваться поток или нет	

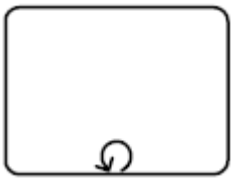
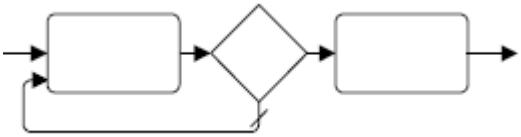
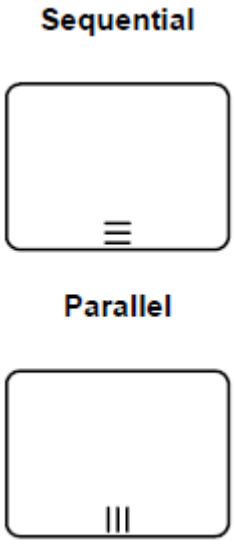
	(например, будет ли токен перемещаться вместе Потоком операций). В случае, если Условный поток операций является исходящим от Действия, то у основания линии изображается небольшой ромбик (см. фигуру справа). Если же Условный поток операций является исходящим от Шлюза, то никакого ромбика у основания линии не будет (см. фигуру ряда выше).	
Поток операций по умолчанию (Default Flow)	Для основанных на данных Эксклюзивных и Неэксклюзивных Условий предназначен лишь один тип потоков – Условный поток операций по умолчанию. Поток операций данного типа используется в том случае, если все остальные исходящие Условные потоки операций не являются верными во время выполнения действия. Для изображения таких Потоков операций используются диагональная черточка, располагающиеся у основания линии (см. фигуру справа).	
Поток исключений (Exception Flow)	Поток исключений встречается за пределами Стандартного потока операций. Основывается на Промежуточных событиях, возникающих в ходе Процесса.	
Поток сообщений (Message Flow)	Поток сообщений используется для отображения потока сообщений между двумя участниками Процесса, готовыми принимать и отсылать сообщения. На диаграмме взаимодействия два отдельно взятых Пула представляют собой двух Участников Процесса (e.g., *PartnerEntities*and/or*PartnerRoles*).	
Компенсирующая ассоциация (Compensation	Компенсирующая ассоциация происходит за рамками Стандартного потока операций. Основой такого рода Ассоциации служит Промежуточноесобытие	

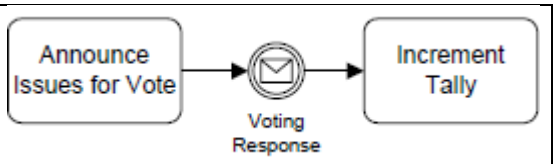
Association)	«Компенсация», инициируемое ошибкой, совершенной в ходе транзакции, либо инициирующим триггер Событием Компенсация. Целью Компенсирующей ассоциации ДОЛЖНО являться компенсирующее действие.	
Объект данных (Data Object)	Однако Объект данных предоставляет информацию о том, какие действия необходимо выполнить и/или каков результат этих действий. Может изображаться как в единственном экземпляре, так и в нескольких. Входные и Выходные данные Объекта данных представляют собой одну и ту же информацию о Процессе.	
Сообщение (Message)	Сообщение используется для отображения сущности взаимодействия между двумя Участниками бизнес-процесса (Участники определяются командами business PartnerRole или business PartnerEntity).	

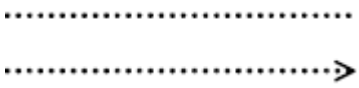
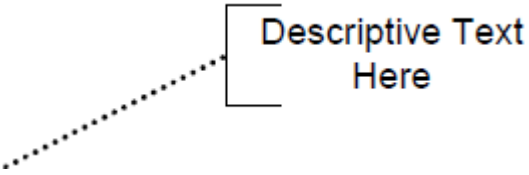
Раздвоение (Fork)	Термин «раздвоение» служит в BPMN для обозначения разделения на два или более параллельных маршрутов (данное явление также называется «И-Разделение»). Раздвоение происходит в том случае, если предпочтение отдается параллельному выполнению действий, нежели последовательному. Существуют два типа Раздвоения: • Множественный исходящий поток операций (см. фигуру справа вверху) Представляет собой Неконтролируемый поток операций, являющийся предпочтительным в большинстве ситуаций. • Параллельный Шлюз (см. фигуру справа ниже). Используется реже, обычно – в сочетании с другими видами Шлюзов.	
Соединение (Join)	Термин «соединение» используется в BPMN для обозначения слияния двух или более параллельных маршрутов в один (данное явление также называется И-Соединение или синхронизация). Параллельный Шлюз предназначается для объединения множественных потоков.	
Условие, Точка ветвления (Decision, Branching Point)	Условиями являются Шлюзы, находящиеся в рамках Процесса или Хореографии, где контрольный поток движется по одному или нескольким альтернативным маршрутам.	См. следующие 5 ячеек
Эксклюзивный шлюз (Exclusive)	Эксклюзивный шлюз представляет собой Точку ветвления, в которой выбор маршрута основывается на условных выражениях (conditional Expressions), хранимых в исходящем Потоке операций. В данном случае может быть выбран лишь один из предложенных маршрутов.	

		
Шлюз, основанный на Событиях (Event-Based)	Данный вид Шлюзов представляет собой Точку ветвления, в которой выбор маршрута основывается на Событии, происходящем в данной точке в ходе Процесса или Хореографии. Отдельно взятое Событие, обычно являющееся получением Сообщения, определяет выбор необходимого маршрута. Также могут использоваться другие типы Событий, например, Событие «Таймер». В данном случае может быть выбран лишь один из предложенных маршрутов. Существуют два пути получения сообщения: через Задачи типа «Получение» (см. фигуру справа вверху) и Промежуточные события «Сообщение» (см. фигуру справа ниже).	
Неэксклюзивный шлюз (Inclusive)	Данный вид Шлюзов представляет собой Точку ветвления, в которой выбор маршрута основывается на условных выражениях, хранимых в Исходящем потоке операций. В некотором смысле, данный вид Шлюзов является группировкой связанных между собой независимых Бинарных Шлюзов (Да/Нет). Т.к. любой из маршрутов является независимым, то МОГУТ использоваться любые сочетания маршрутов (от нуля до максимального количества	

	комбинаций маршрутов). Однако при построении диаграмм необходимо учитывать то, что должен быть выбран хотя бы один маршрут. Для проверки того, что выбран, по меньшей мере, один маршрут, может быть использовано Условие по умолчанию. Существую два вида данного типа Шлюзов. • Первый тип использует совокупность Условных потоков операций. На схеме выделяется при помощи небольших ромбиков (см. фигуру справа вверху). • Второй тип использует Неэксклюзивные Шлюзы (см. фигуру справа ниже).	
Слияние (Merging)	Термин «слияние» используется в BPMN для обозначения исключающего объединения двух или более маршрутов в один (данное явление также называется ИЛИ-Соединение). Эксклюзивный шлюз «Слияние» предназначается для отображения слияния множества потоков. В случае, если все Входящие потоки операций являются альтернативными, то необходимость в Шлюзе отпадает. Это означает, что такое же влияние на ход Процесса оказывает и Неконтролируемый поток операций (см. фигуру справа ниже).	
Цикличность (Looping)	В BPMN существуют два механизма, обеспечивающих цикличность внутри Процесса.	См. следующих две фигуры
Цикличность действия (Activity Looping)	Атрибуты Задач и Подпроцессов указывают на то, будут ли они повторяться или будут выполнены единожды. Существуют два вида	

	циклов: Стандартный и Многоэкземплярный. Графически цикличность отображается в виде небольшого маркера в центре нижней части фигуры.	
Цикличность Потока операций (Sequence Flow Looping)	Циклы могут появляться благодаря присоединению Потока операций к «противоположному» объекту. Объект является противоположным в том случае, если от него направлен Исходящий поток операций, ведущий к ряду других Потоков операций, последний из которых является Входящим потоком операций для исходного объекта.	
Многоэкземплярность (Multiple Instances)	Атрибуты Задач и Подпроцессов указывают на то, будут ли они повторяться или будут выполнены единожды. Три горизонтальные полоски в центре нижней части фигуры указывают на последовательную многоэкземплярность (см. фигуру справа вверху). Три вертикальные полоски в центре нижней части фигуры указывают на параллельную многоэкземплярность (см. фигуру справа ниже).	
Перерыв в Процессе (что-то, способное приостановить	Перерыв в Процессе представляет собой участок Процесса, указывающий, на каком его отрезке произойдет ожидаемая задержка. Для отображения действительного	

Процесс и не подающееся управлению) (Process Break)	хода Процесса используется Промежуточное действие (см. фигуру справа вверху). Необходимо отметить, что Артефакт Перерыва в Процессе по желанию разработчика модели или программы моделирования может быть отнесен к Событиям, что подчеркнет расположение задержки внутри потока.	
Транзакция (Transaction)	Транзакция представляет собой Подпроцесс, поддерживаемый особым протоколом, гарантирующим то, что между всеми участвующими сторонами заключено соглашение о том, что действие следует либо завершить, либо отклонить. Графические элементы действия указывают на то, является ли действие соглашением. Граница, выполненная двойной линией, указывает на то, что данный Подпроцесс является Транзакцией.	
Вложенный/Встроенный Подпроцесс (Nested/EmbeddedSub-Process (Inline Block))	Вложенный (или встроенный) Подпроцесс представляет собой действие, имеющее тот же набор данных, что и родительский Процесс. Данный тип Подпроцесса является противоположным независимому Подпроцессу, который может быть использован заново и на который ссылается родительский Процесс. При использовании Потока операций данные должны передаваться основному, а не вложенному Подпроцессу.	На диаграмме данный вид Подпроцесса не имеет никаких особых маркеров

Группа (блок, содержащий группу объектов одной категории) (Group)	Группа предназначена для группировки графических элементов, принадлежащих одной и той же категории. Такая группировка не оказывает влияния на Поток операций. На диаграмме бизнес-процесса название категории, к которой принадлежат сгруппированные элементы, отображается в качестве названия группы. Такого рода группировка может использоваться в целях составления документации или при проведении анализа. Графически Группы отображаются так же, как и Категории объектов.	
Соединитель страниц (Off-Page Connector)	На диаграмме данный графический элемент отображается там, где на предыдущей странице заканчивается Поток операций, а затем - где он возобновляется на следующей странице. В качестве соединителя страниц может использоваться Промежуточное событие «Связь». Предназначен в основном для печати.	
Ассоциация (Association)	Ассоциация служит для установления связи между информацией или Артефактами (объектами, не относящимися к Элементам потока) и элементами потока. Текстовые объекты, а также графические объекты, не относящиеся к элементам потока, могут соотноситься с элементами потока. При необходимости Ассоциация может указывать направление потока (например, потока данных).	
Текстовая аннотация (связана с Ассоциацией) (Text Annotation)	Текстовые аннотации являются механизмом, позволяющим разработчику модели бизнес-процесса вводить дополнительную информацию для тех, кто работает с BPMN диаграммами.	

Пул(Pool)	Пул представляет собой Участника Взаимодействия. Пул также может выступать в качестве Зоны ответственности или графического контейнера, отвечающего за разделение определенного набора действий, относящихся к другим Пулам, что обычно встречается в ситуациях типа «бизнес для бизнеса» (B2B). Внутри Пула МОЖЕТ находиться дополнительная информация по выполняемому Процессу. В случае, если такой информации в Пуле не содержится, то он МОЖЕТ представлять собой «черный ящик».	
Дорожка (Lane)	Дорожка используется для отображения распределения ролей и может быть как вертикальной, так и горизонтальной (также может использоваться для разделения внутреннего пространства Пула). Служит для упорядочивания и категоризации Действий.	

7.3. Типы Диаграмм Бизнес-процессов (BPMN Diagram Types)

BPMN 2.0 содержит описание трех основных моделей **Процессов**: *приватный* **Процесс** (как *выполняемый*, так и *невыполняемый*), *публичный* **Процесс** и **Хореография** (**Choreography**). С помощью вышеперечисленных основных моделей может быть создано множество вариантов диаграмм **Бизнес-процессов**. Ниже приведены подмодели **Бизнес-процессов**, спроектированные с помощью **BPMN 2.0**:

- Высокоуровневые *невыполняемые* **Действия** (нефункциональный анализ).
- Детализированный *выполняемый* **Бизнес-процесс**.
- **Бизнес-процесс** «As-is» (устаревший).
- **Бизнес-процесс** «To-be» (новый).
- **Хореография** (**Choreography**). Описание поведения, ожидаемого от двух или более у *Участников* **Процесса**.
- Детализированный *приватный* **Бизнес-процесс** (как *выполняемый*, так и *невыполняемый*), включающий взаимоотношения между одним или более внешними участниками (**Процесс** типа «черный ящик»).
- Два или более детализированных *выполняемых* взаимодействующих **Процесса**.
- Детализированный *выполняемый* **Бизнес-процесс**, взаимодействующий с **Хореографией**.
- Два или более *публичных* **Процесса**.
- *Публичный* **Процесс**, взаимодействующий с **Хореографией**.
- Два или более детализированных *выполняемых* **Бизнес-процесса**, взаимодействующих посредством **Хореографии**.

Данная нотация была создана для возможности описания вышеперечисленных примеров **Бизнес-процессов**. Однако следует отметить, что создание различных вариантов сочетания подмоделей предоставлено производителям инструментов моделирования **Бизнес-процессов**. При использовании **BPMN 2.0** разработчику модели **Бизнес-процесса** РЕКОМЕНДУЕТСЯ быть ориентированным на выбранный объект моделирования, например, *приватный* **Бизнес-процесс** или **Хореографию**, хотя сама нотация ничего не навязывает.

7.4. Использование текста, цвета и линий в моделировании диаграмм

Текстовые аннотации объектов используются разработчиком модели с целью отобразитьдополнительную информацию о **Процессе** или атрибутах объектов, расположенных на диаграмме.

- Элементы потока и другие элементы диаграммы МОГУТ носить текстовые метки (labels) (например, имя потока и/или названия других его атрибутов). Текстовые метки могут помещаться как внутри фигуры, так и над или под ней. Месторасположение текстовых меток, а также их направление может быть любым в зависимости от задумки разработчика модели или программы моделирования.
- Заливка графического элемента МОЖЕТ БЫТЬ как белого цвета, так и прозрачной.
    - Графическая нотация МОЖЕТ допускать использование какого-либо другого цвета заливки для удовлетворения требований разработчика модели или программы моделирования (например, выделение значения атрибута объекта). Однако следует помнить о следующих правилах:
        - События, определяющие дальнейший ход потока, ДОЛЖНЫ иметь темную заливку (см. заголовки Конечное события и Промежуточное событие).
        - Дорожки Участников в фигуре Хореографии или Подхореографии ДОЛЖНЫ иметь светлую заливку в том случае, если Хореография/Подхореография (**Choreography/Sub-choreography**) не запускают Действие (см. заголовки **Хореография** и **Подхореография**).
- Элементы потока и маркеры МОГУТ БЫТЬ того размера, который удовлетворяет требованиям разработчика модели или программы моделирования.
- Линии, используемые в моделировании диаграмм, МОГУТ БЫТЬ черными.
    - Графическая нотация допускает использование других цветов линий для удовлетворения требований разработчика модели или программы моделирования (например, выделение значения атрибута объекта).
    - Графическая нотация МОЖЕТ допускать использование разного дизайна линий для удовлетворения требований разработчика модели или программы моделирования (например, выделение значения атрибута объекта), однако, при условии, что выбранный дизайн линий НЕ ДОЛЖЕН противоречить ни одному из вариантов, предложенных языком BPMN. Таким образом, дизайн линий, используемых для изображения Потока операций, Потока сообщений, а также Ассоциаций, изменяться НЕ ДОЛЖЕН.

7.5. Правила соединения элементов потока

Входящий **Поток операций** может быть присоединен к любой точкой **Элемента потока** (слева, справа, сверху, снизу). Подобно ему, *Исходящий* **Поток операций** может брать начало из любой точки **Элемента потока** (слева, справа, сверху, снизу). **Поток сообщений** обладает теми же свойствами, что и **Поток операций**. Язык **BPMN** может подстраиваться под предъявляемые требования, однако, для соединения **Элементов потока** разработчикам моделей РЕКОМЕНДУЕТСЯ использовать имеющийся опыт, что облегчит понимание создаваемых диаграмм и сделает ход изображаемого **Бизнес-процесса** прозрачным и доступным для понимания. Это особенно важно в том случае, если в диаграмме присутствуют такие графические элементы, как **Поток операций** или **Поток сообщений**. В данном случае оптимальным вариантом является выбор направления **Потока операций**, располагающегося либо слева направо, либо сверху вниз, а затем и выбор направления **Потока сообщений**, который необходимо расположить под углом в 90° по отношению к уже выбранному **Потоку операций**. При выполнении всех вышеперечисленных требований создаются удобные для работы диаграммы.

7.5.1. Правила соединения потоков операций

Таблица 7.3 содержит изображения **Элементов потока**, используемые языком **BPMN**, а также показывает, каким образом данные графические элементысоединяются друг с другом посредством **Потока операций**. Эти

правила применимы как к диаграмме **Процесса**, так и к диаграмме **Хореографии**. Символ ↗ обозначает, что графический элемент, изображенный в одной из строк таблицы, может соединяться с графическим элементом, изображенным в соответствующей колонке. В таблице не указывается количество входящих и исходящих соединений графического элемента, зависящее от различных конфигураций. Следующая глава содержит детальную информацию о правилах соединения каждого отдельно взятого графического элемента. *Обратите внимание, что в случае, если* ***Подпроцесс*** *занимает всю протяженность диаграммы, то графические элементы, находящиеся внутри данного* ***Подпроцесса****, не могут быть соединены с графическими элементами, находящимися за его пределами. Подобно этому,* ***Поток операций*** *не может пересекать границ* ***Пула****.*

Таблица 7.3 – Правила Соединения Потока Операций

От/К						
		‰	↗	↗	↗	↗
		‰	↗	↗	↗	↗
		‰	↗	↗	↗	↗
		‰	↗	↗	↗	↗
		‰	↗	↗	↗	↗

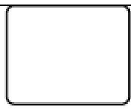
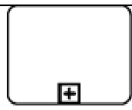
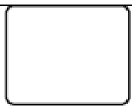
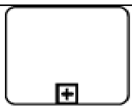
Примечание: В таблице отображены лишь графические элементы, имеющие *Входящие* или *Исходящие* **Потоки операций**. Такие объекты, как **Пул, Дорожка, Объект данных, Группа** и **Текстовая аннотация**, в таблице не содержатся. Действие здесь подразумевает **Действие** и **Подпроцесс** в контексте **Процесса**, а также **Действия Хореографии** и **Подхореографии** в контексте **Хореографии**.

7.5.2. Правила соединения потоков сообщений

Таблица 7.4 содержит изображения объектов моделирования **BPMN**, а также показывает, каким образом данные объекты соединяются друг с другом посредством **Потока сообщений**. Эти правила также применимы к элементам диаграммы **Взаимодействия**. Символ ⇗ обозначает, что графический элемент, изображенный в одной из строк таблицы, может соединяться с графическим элементом, изображенным в соответствующей колонке. В таблице не указывается количество входящих и исходящих соединений графического элемента,

зависящее от различных конфигураций. Следующая глава содержит детальную информацию о правилах соединения каждого отдельно взятого графического элемента. *Обратите внимание, что* ***Поток сообщений*** *не соединяется с объектами, расположенными в пределах одного* ***Пула***.

Таблица 7.4 – Правила Соединения Потока сообщений

От/К						
						
		⇗	⇗	⇗	⇗	
		⇗	⇗	⇗	⇗	
		⇗	⇗	⇗	⇗	
		⇗	⇗	⇗	⇗	
		⇗	⇗	⇗	⇗	

Примечание: В таблице отображены лишь графические элементы, имеющие *Входящие* или *Исходящие* **Потоки сообщений**. Такие объекты, как **Дорожка**, **Шлюз**, **Объект данных**, **Группа** и **Текстовая аннотация**, в таблице не содержатся.

7.6. Расширяемость BPMN

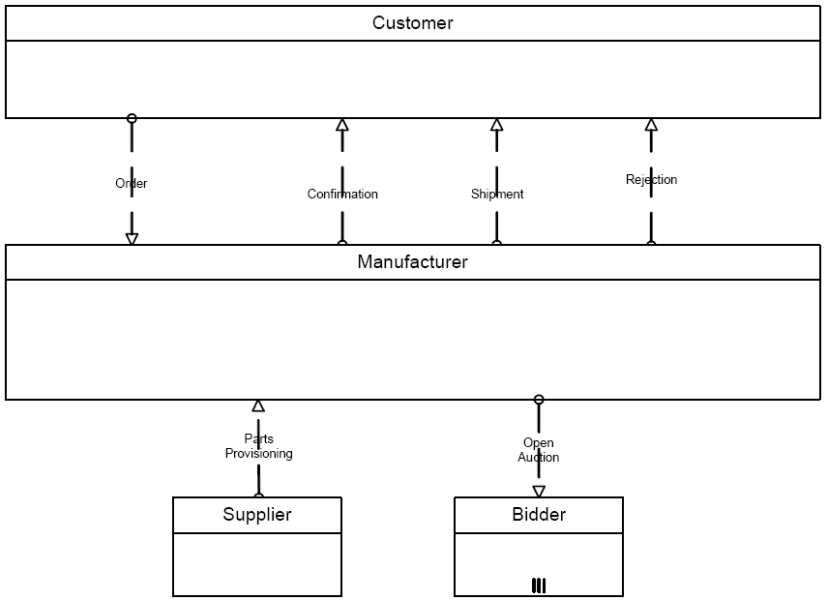
BPMN 2.0 описывает механизм, позволяющий расширять список атрибутов для стандартных графических элементов диаграммы. При необходимости разработчиком модели или программой моделирования могут быть задействованы нестандартные атрибуты графических элементов или Артефакты, такие, как уникальные требования длявертикальной области. Для того, чтобы не нарушить логику, описываемую в **BPMN**, такие атрибуты НЕ ДОЛЖНЫ противоречить семантике использования любого их графических элементов **BPMN**. Необходимо отметить, что, несмотря на возможность добавления новых атрибутов, должны быть сохранены все основные принципы построения и наглядность диаграммы для лучшего её восприятие пользователем любого уровня подготовки. Помните, что фигуры основных элементов потока (**События**, **Действия** и **Шлюзы**) НЕ ДОЛЖНЫ видоизменяться.

Данная спецификация делает различие между обязательными и дополнительными элементами (см. раздел 8.3.2, содержащий описание синтаксиса для создания расширений). Если используются обязательные элементы, то они

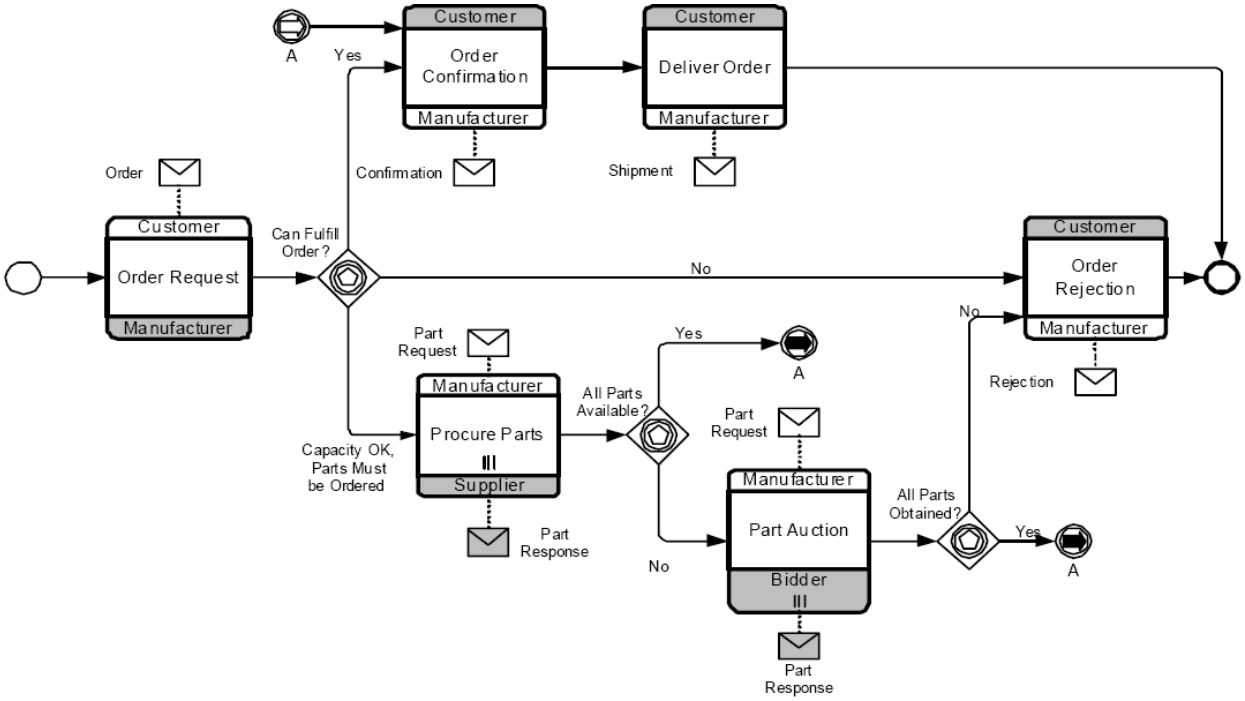
ДОЛЖНЫ учитываться при внедрении диаграмм. Если используются дополнительные элементы, то при внедрении диаграмм они МОГУТ БЫТЬ опущены.

7.7. Примеры Процессов BPMN

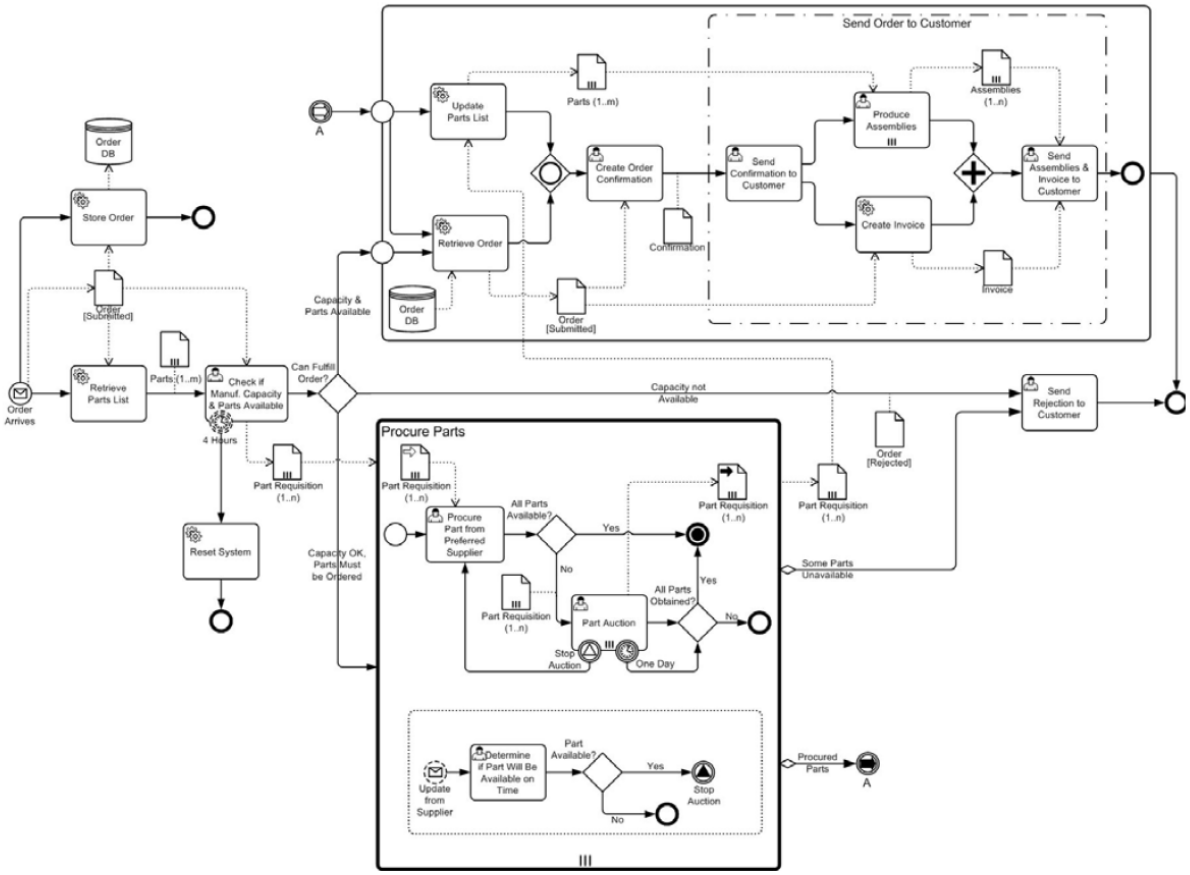
В данном подразделе представлен пример производственного процесса, рассмотренный с разных ракурсов.



Фигура 7.6 – Пример диаграммы Взаимодействия с Пулами в виде «черных ящиков».



Фигура 7.7 – Пример диаграммы автономной Хореографии.

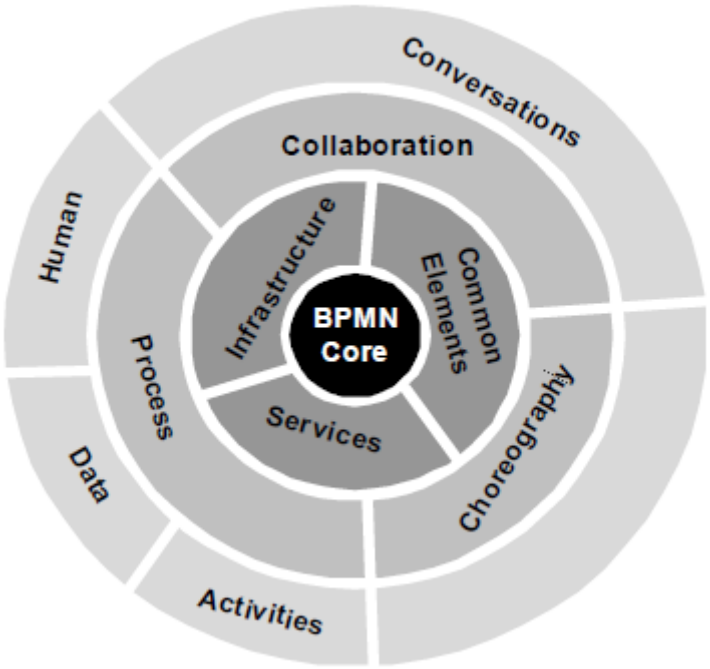


Фигура 7.8 – Пример диаграммы автономного Процесса (Оркестровки).

8. Структура BPMN

Примечание: Следование содержимому данной главы является НЕОБХОДИМЫМ условием соответствия всем требованиям **BPMN**. Для более подробной информации см. подраздел 2.1.

С технической точки зрения, построение структуры **BPMN** основывается на принципе расширяемости вышестоящих слоев основного ряда простых элементов, определенных в данной спецификации как *Элементы Ядра* (*Core Elements*). Отталкиваясь от набора основных конструкций, разделение на слои используется для описания дополнительных элементов спецификации, которые используются для расширения существующих и добавления новых конструкций. Разделение на слои по понятной причине зависит от путей разрешения ситуации. Модель типа XML Schema легко приспосабливается к структурной модели с импортом и механизмом принятия решений, который устраняет неточности в выборе определений для элементов внешних слоев.



Фигура 8.1 – Вид Ядра и структуры слоев BPMN

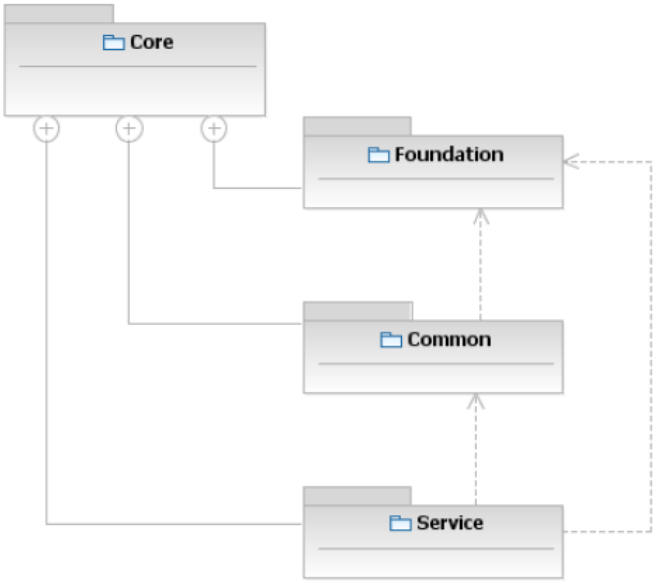
На фигуре 8.1 отображены основные принципы разделения на слои, тактика которого вполне прозрачна. При таком подходе для расширения используются формализованные конструкции.

Дополнительным преимуществом разделения является то, что может быть реализована совместимость слоев, благодаря чему производители инструментов моделирования могут применять различные уровни совместимости. Таким образом, для поддержки различных вертикальных областей и целевой аудитории они могут определять собственные слои. Также, разделение на слои предоставляет механизм для повторного определения уже существующих понятий без воздействия на обратную совместимость (совместимость данной спецификации с чем-либо). Обеспечивается определение двух или более не пригодных для компоновки слоев, а совместимость с данной спецификацией и обратная совместимость достигаются без каких-либо неудобств.

Структура **BPMN** разделена на слои, при этом каждый слой выстраивается на вершине иерархии и определяет расширение расположенных ниже слоев. У структуры имеется *Ядро* (*Core* или kernel), содержащее наиболее важные элементы **BPMN**, НЕОБХОДИМЫЕ для построения диаграмм, а именно: **Процесс** (**Process**), **Хореография** (**Choreography**) и **Взаимодействие** (**Collaboration**). *Ядро* должно быть простым, сжатым и расширяемым, а также реализовывать определенное поведение.

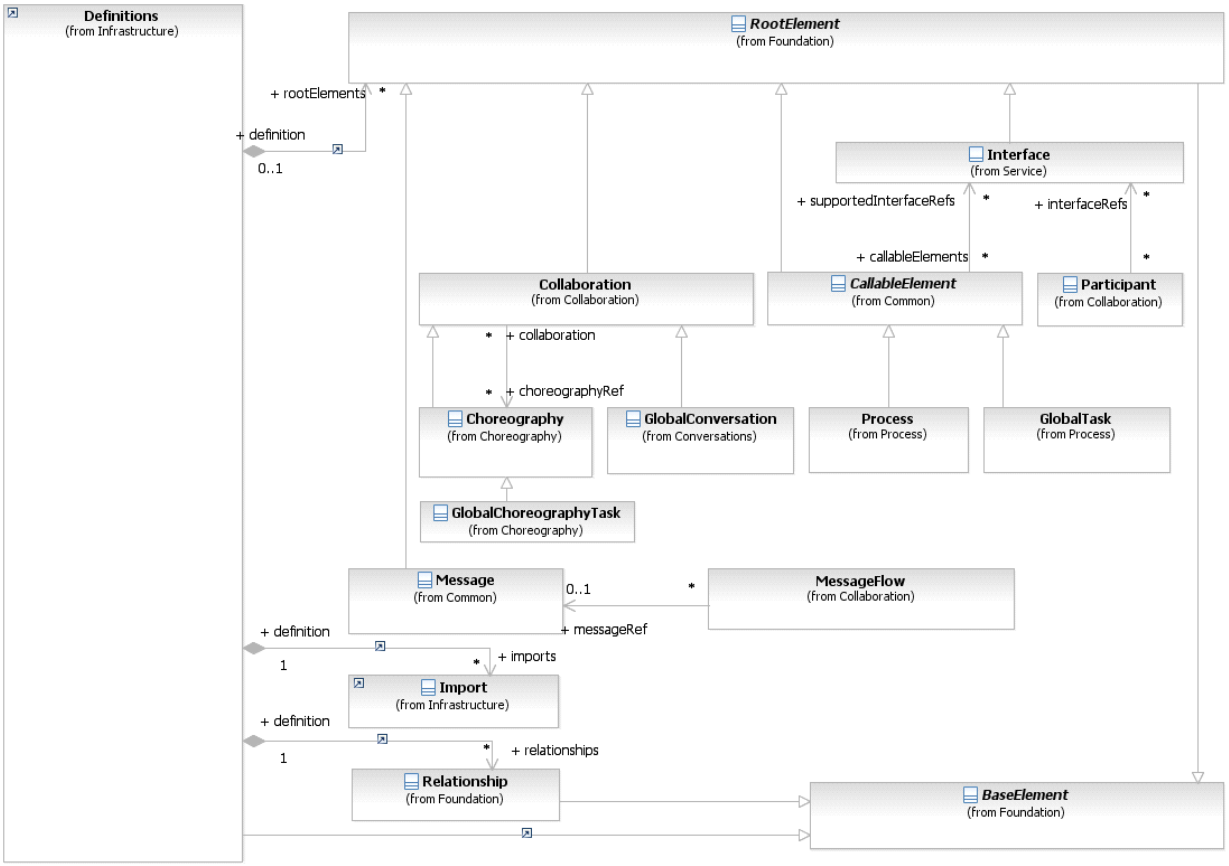
Ядро вмещает три пакета (см. фигуру 8.2):

1. `Foundation`. Содержит основные конструкции, необходимые для моделирования диаграмм **BPMN**.
2. `Service`. Содержит основные конструкции, необходимые для моделирования сервисов и интерфейсов.
3. `Common`. Содержит классы, являющиеся общими для слоев **Процесса** (**Process**), **Хореографии** (**Choreography**) и **Взаимодействия** (**Collaboration**).



Фигура 8.2 – Диаграмма классов, отображающая пакеты Ядра (core packages)

На фигуре 8.3 отображена организация основных элементов моделирования **BPMN**.



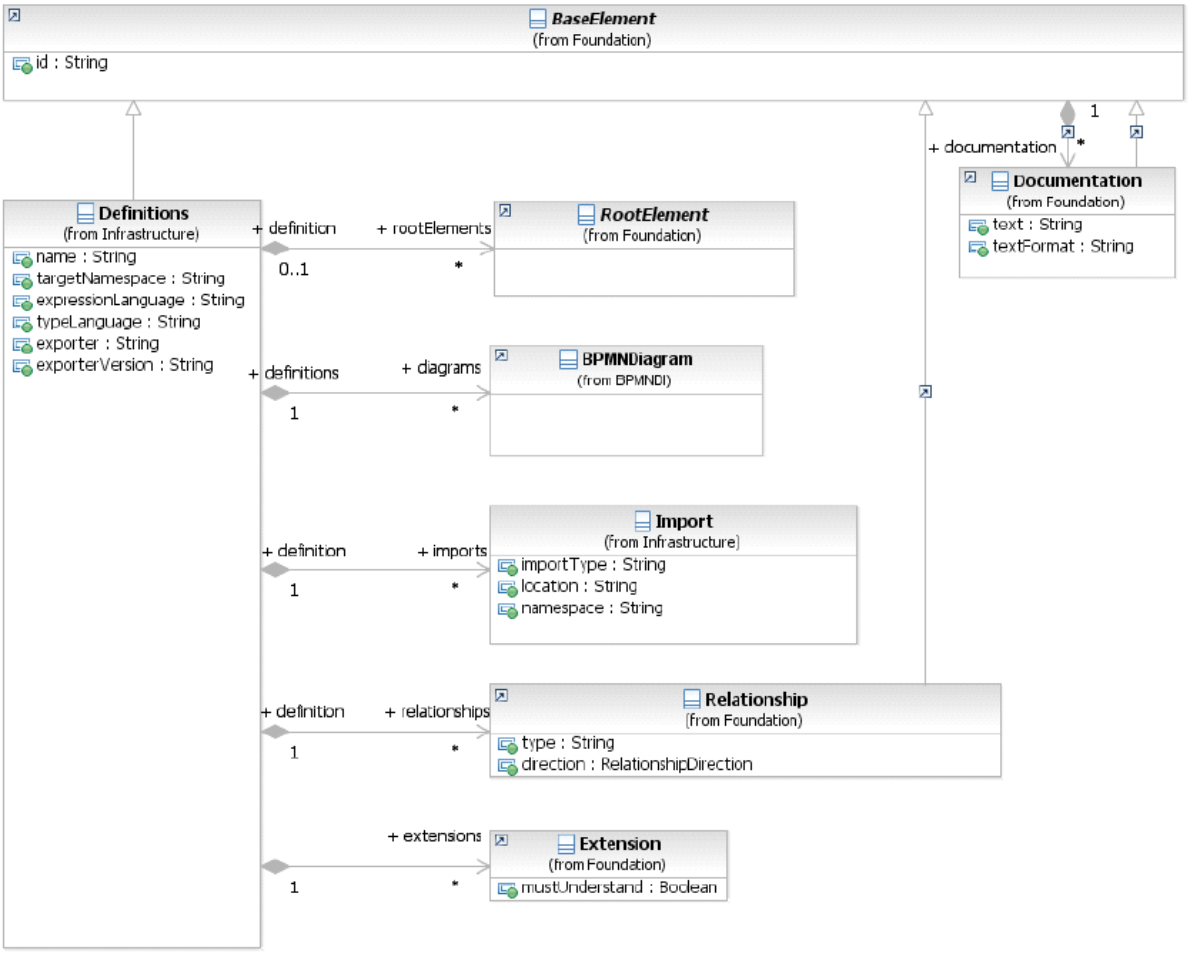
Фигура 8.3 – Диаграмма классов, отображающая организацию основных элементов BPMN

8.1. Пакет Infrastructure

В BPMN пакет Infrastructure состоит из двух элементов, используемых как для моделей с абстрактным синтаксисом, так и для диаграмм.

8.1.1. Класс Definitions

Класс Definitions – это крайний объект всех элементов **BPMN**. Посредством элементов данного класса определяются границы видимости и пространство имен элементов, содержащих элементы класса. Обмен файлами **BPMN** всегда осуществляется благодаря использованию одного или более элементов Definitions.



Фигура 8.4. – Диаграмма класса Definitions

Элемент Definitions наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 8.1 содержит информацию о дополнительных атрибутах и ассоциациях элемента Definitions.

Таблица 8.1 – Атрибуты и ассоциации элемента Definitions

Название атрибута	Описание/использование
name: string	Название элемента `Definition`.
targetNamespace: string	При помощи данного атрибута определяется пространство имен, ассоциированное с `Definition`. Использование данного атрибута

	также позволяет удовлетворять требованиями языка XML Schema.
expressionLanguage: string [0..1]	При помощи данного атрибута определяется язык формального `Выражения` (formal `Expression`), используемого в этом выражении (само оно находится в элементах `Definition`). Значением по умолчанию является «http://www.w3.org/1999/XPath», однако, оно МОЖЕТ БЫТЬ изменено для каждого отдельно взятого формального Выражения. Язык ДОЛЖЕН БЫТЬ указан в формате URL.
typeLanguage: string [0..1]	При помощи данного атрибута определяется тип системы, используемый элементами `Definitions`. Значением по умолчанию является «http://www.w3.org/2001/XMLSchema», однако, оно может быть изменено для каждого отдельно взятого `ItemDefinition`. Язык ДОЛЖЕН БЫТЬ указан в формате URL.
rootElements: RootElement [0..*]	При помощи данного атрибута определяется список корневых элементов `Definitions`. На данные элементы может быть указана ссылка. Они видны другим `Definitions`.
diagrams: BPMNDiagram [0..*]	При помощи данного атрибута определяется список `Диаграмм BPMN`, содержащихся в `Definitions` (для получения более подробной информации о `Диаграммах BPMN` см. Главу 12).
imports: Import [0..*]	При помощи данного атрибута осуществляется импорт определенных извне элементов и открывается доступ для использования их элементами `Definitions`.
extensions: Extension [0..*]	При помощи данного атрибута указываются расширения (`extensions`) помимо атрибутов и ассоциаций, указанных в спецификации **BPMN** (для получения более подробной информации о возможности расширения см. подраздел 8.2.3).
relationships: Relationship [0..*]	Данный атрибут позволяет выполнять расширение моделей **BPMN** и интегрировать их в более масштабные системные **Процессы** или **Процессы** разработки.
exporter: string [0..1]	При помощи данного атрибута указывается инструмент для экспорта файла с моделью **BPMN**.
exporterVersion: string [0..1]	При помощи данного атрибута указывается версия инструмента для экспорта файла с моделью **BPMN**.

8.1.2. Класс Import

Класс `Import` используется при наличии ссылки на внешний элемент (это может быть как элемент **BPMN**, так и не относящийся к данной спецификации элемент). Элементы `Imports` ДОЛЖНЫ БЫТЬ видны.

Таблица 8.2 содержит информацию об атрибутах элемента `Import`.

Таблица 8.2 – Атрибуты элемента Import

Название атрибута	Описание/использование
importType: string	Посредством данного атрибута указывается тип документа, импортируемого путем добавления абсолютного URL, который определяет используемую в документе кодировку. При импорте документов типа XML Schema 1.0 значением по умолчанию является «http://www.w3.org/2001/XMLSchema», при импорте документов WSDL 2.0 таким значением является «http://www.w3.org/TR/wsdl20/», а при импорте документов типа BPMN 2.0 значением по умолчанию является «http://www.omg.org/spec/BPMN/20100524/MODEL». Также МОГУТ поддерживаться и другие типы документов. Документы типов Xml Schema 1.0, WSDL 2.0 и BPMN 2.0 ДОЛЖНЫ поддерживаться в обязательном порядке.
location: string [0..1]	Посредством данного атрибута указывается расположение импортированного элемента.
namespace: string	Посредством данного атрибута указывается пространство имен импортированного элемента.

8.1.3. XML схемы пакета Infrastructure

Таблица 8.3 – XML схема элементов Definitions

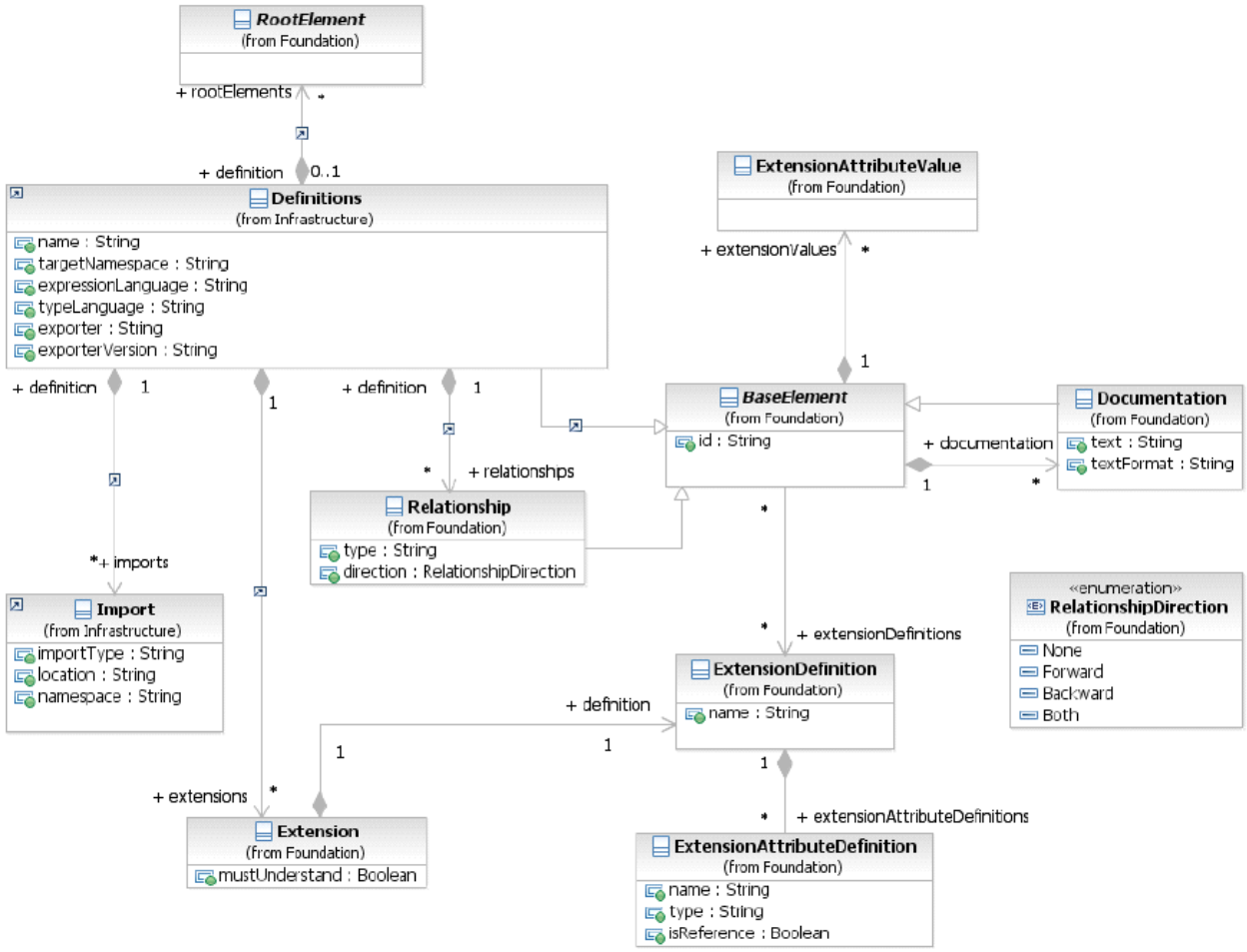
```
<xsd:element name="definitions" type="tDefinitions"/>
<xsd:complexType name="tDefinitions">
    <xsd:sequence>
        <xsd:element ref="import" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="extension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="rootElement" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="bpmndi:BPMNDiagram" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="relationship" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="optional"/>
    <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="expressionLanguage" type="xsd:anyURI" use="optional"
    default="http://www.w3.org/1999/XPath"/>
    <xsd:attribute name="typeLanguage" type="xsd:anyURI" use="optional"
default="http://www.w3.org/2001/XMLSchema"/>
    <xsd:anyAttribute name="exporter" type="xsd:ID"/>
    <xsd:anyAttribute name="exporterVersion" type="xsd:ID"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
```

Таблица 8.4 - XML схема элемента Import

```
<xsd:element name="import" type="tImport"/>
<xsd:complexType name="tImport">
    <xsd:attribute name="namespace" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="location" type="xsd:string" use="required"/>
    <xsd:attribute name="importType" type="xsd:anyURI" use="required"/>
</xsd:complexType>
```

8.2. Пакет Foundation

Пакет `Foundation` содержит два класса, распределенных между другими пакетами `Ядра` (см. фигуру 8.5) модели с абстрактным синтаксисом.



Фигура 8.5 – Классы пакета Foundation

8.2.1. Base Element

Элемент BaseElement представляет собой абстрактный суперкласс для большинства элементов **BPMN**. С его помощью становится возможным наследование другими элементами его атрибутов id и документации.

Таблица 8.5 содержит информацию об атрибутах и ассоциациях элемента `BaseElement`.

Таблица 8.5 – Атрибуты и ассоциации элемента BaseElement

Название атрибута	Описание/использование
id: string	Данный атрибут используется для указания уникальных значений элементов **BPMN**. Атрибут id НЕОБХОДИМ в случае, если на данный элемент есть (или предполагается) ссылка. Если же ничто не ссылается на данный элемент (и это даже не предполагается), то атрибут id МОЖНО опустить.
documentation: Documentation [0..*]	Данный атрибут используется для размещения

	аннотации к элементу **BPMN** (к примеру, описание или какая-то другая документация).
extensionDefinitions: ExtensionDefinition [0..*]	Данный атрибут используется для привязки дополнительных атрибутов и ассоциаций к какому-либо элементу BaseElement. Такая связь не может быть применена в случае, если произведена замена XML-схемы, т.к. механизмы XSD, осуществляющие поддержку `любых атрибутов` и элементов, уже удовлетворили этому требованию. Для получения более подробной информации о возможности расширения см. подраздел 8.2.3.
extensionValues: Exten-sionAttributeValue [0..*]	Данный атрибут используется для установки значений расширяемых атрибутов и ассоциаций. Он не может быть применен в случае, если произведена замена XML-схемы, т.к. механизмы XSD, осуществляющие поддержку `любых атрибутов` и элементов, уже удовлетворили этому требованию. Для получения более подробной информации о возможности расширения см. подраздел 8.2.3.

8.2.2. Documentation

Благодаря использованию элемента Documentation, все элементы **BPMN**, наследующие атрибуты или ассоциации элемента `BaseElement`, могут содержать одно или два текстовых описания.

Элемент `Documentation` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.6 содержит информацию о дополнительных атрибутах элемента `Documentation`.

Таблица 8.5 – Атрибуты элемента Documentation

Название атрибута	Описание/использование
text: string	Данный атрибут используется для ввода текстового описания элемента **BPMN**.
textFormat: string	Посредством данного атрибут указывается формат текста. Это ДОЛЖЕН БЫТЬ документ MIME-типа. Значением по умолчанию является «text/plain».

На схеме **BPMN** complexType tDocumentation не содержит текстовых атриабутов или элементов. Вместо этого, текст документа появляется в теле данного элемента. Рассмотрите пример:

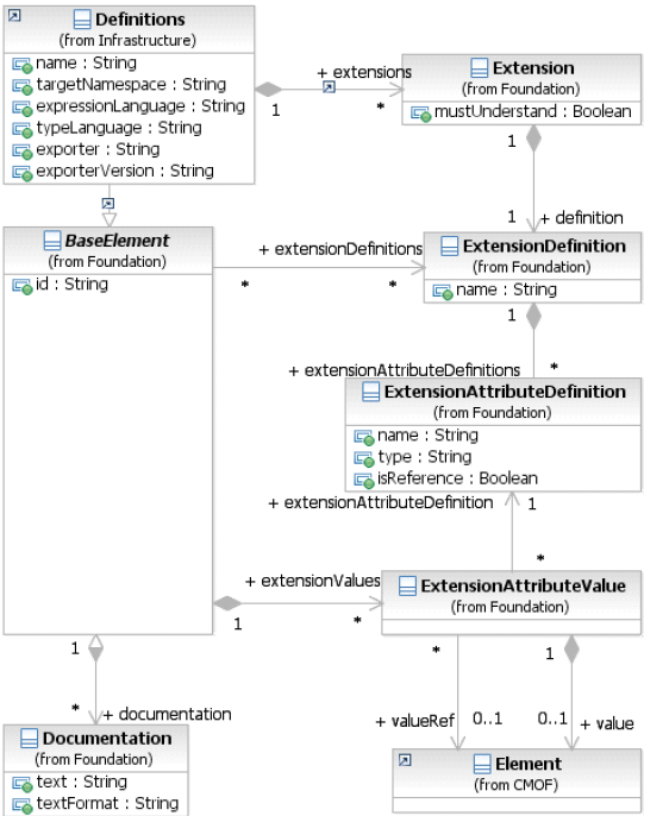
```
<documentation>An example of how the documentation text is
entered.</documentation>
```

8.2.3. Extensibility

Предполагается, что метамодель **BPMN** способна к расширению, что позволяет людям, использующим этот язык, расширять конкретные метамодели с тем условием, чтобы они оставались совместимыми с **BPMN**.

Существует ряд элементов, используемых для расширения. Применение таких элементов позволяет пользователям добавлять атрибуты и элементы помимо стандартных для **BPMN**.

Благодаря способности к расширению, в результате чего стандартные элементы остались неизмененными и легкоузнаваемыми для пользователей **BPMN**, модели стали взаимозаменяемы. В процессе замены МОГУТ БЫТЬ утеряны лишь дополнительные атрибуты и элементы.



Фигура 8.6 – Диаграмма класса Extension

Класс Extension, используемый в **BPMN**, состоит в основном из четырех различных элементов:

1. Extension
2. ExtensionDefinition
3. ExtensionAttributeDefinition
4. ExtensionAttributeValue

`ExtensionDefinition` и `ExtensionAttributeDefinition` являются основными элементами класса `Extension`. Позднее был составлен список атрибутов для добавления к стандартным элементам **BPMN**. В списке атрибутов содержится информация о названии и типе каждого добавляемого атрибут. Используя эти данные, пользователь **BPMN** может осуществлять интеграцию любой метамодели в метамодель **BPMN**, а также повторно использовать уже существующие элементы модели.

Элемент `ExtensionDefinition` может быть создан отдельно от какого-либо элемента или определения **BPMN**.

Для того чтобы использовать его в рамках элемента модели **BPMN** (элемента `Definitions`), данный элемент `ExtensionDefinition` ДОЛЖЕН БЫТЬ ассоциирован с элементом Extension, используемым для привязки `ExtensionDefinition` к определениям (элементам) модели **BPMN**.

Любой элемент **BPMN**, входящий в подкласс `BaseElement` этого **BPMN**, может быть расширен путем присоединения к нему дополнительных атрибутов. При этом необходимо, чтобы расширяемый элемент **BPMN**

был ассоциирован с элементом `ExtensionDefinition`, указанным на уровне определений (элементов `Definitions`) модели **BPMN**.

Также следует помнить, что любой «расширенный» элемент **BPMN** содержит текущее значение атрибута расширения. Это значение, определенное элементом `ExtensionAttributeValue`, в свою очередь, хранит в себе значение типа `Element`. Оно также ассоциировано с соответствующим определением атрибута.

Элемент Extension

Элемент `Extension` связывает элемент `ExtensionDefinition` и его атрибуты с определением модели **BPMN** (импортирует элемент `ExtensionDefinition` и его атрибуты в необходимое определение).

Таблица 8.7 содержит информацию об атрибутах и ассоциациях элемента `Extension`.

Таблица 8.7 – Атрибуты и ассоциации элемента Extension

Название атрибута	Описание/использование
mustUnderstand: boolean [0..1] = False	Используется для указания того, ДОЛЖНА ли БЫТЬ семантика, указанная посредством элемента расширения и его атрибутов, понятна пользователям для корректного управления моделью **BPMN**. Значением по умолчанию является «*False*».
definition: ExtensionDefinition	Определяет содержимое расширения. Обратите внимание, что на XML- схеме это определение поставляется внешним файлом XML schema и на него ссылается QName.

ExtensionDefinition

Класс `ExtensionDefinition` служит для определения и группировки дополнительных атрибутов. Данный тип не может быть использован в случае, если произведена замена XML-схемы, т.к. Сложные Типы (`Complex Types`) XSD уже удовлетворили этому требованию.

Таблица 8.8 содержит информацию об атрибутах и ассоциациях элемента `ExtensionDefinition`.

Таблица 8.8 – Атрибуты и ассоциации элемента ExtensionDefinition

Название атрибута	Описание/использование
name: string	Название расширения. Используется в качестве пространства имен для присвоения уникальной характеристики содержимому расширения.
extensionAttributeDefinitions: ExtensionAttributeDefinition [0..*]	Особые атрибуты, составляющие расширение.

ExtensionAttributeDefinition

Посредством элемента `ExtensionAttributeDefinition` указываются новые атрибуты. Данный тип не может быть использован в случае, если произведена замена XML-схемы, т.к. механизмы XSD, поддерживающие «`AnyAttribute`» или «`Any`», уже удовлетворили этому требованию.

Таблица 8.9 содержит информацию об атрибутах элемента `ExtensionAttributeDefinition`.

Таблица 8.9 – Атрибуты элемента ExtensionAttributeDefinition

Название атрибута	Описание/использование
name: string	Название атрибута расширения.
type: string	Тип, ассоциированный с данным атрибутом.
isReference: boolean [0..1] = False	Указывает на то, будет ли значение атрибута содержаться в нем, либо на него будет дана ссылка.

ExtensionAttributeValue

Элемент `ExtensionAttributeValue` содержит значение атрибута. Данный тип не может быть использован в случае, если произведена замена XML-схемы, т.к. механизмы XSD, поддерживающие «`AnyAttribute`» или «`Any`», уже удовлетворили этому требованию.

Таблица 8.10 содержит информацию об ассоциациях элемента `ExtensionAttributeValue`.

Таблица 8.10 – Ассоциации элемента ExtensionAttributeValue

Название атрибута	Описание/использование
value: [Element [0..1]	Значение содержащегося атрибута, используемое в случае, если ассоциированное значение `ExtensionAttributeDefinition.isReference` равно «*false*». Тип данного Элемента ДОЛЖЕН соответствовать типу, указанному в ассоциированном с ним элементе `ExtensionAttributeDefinition`.
valueRef: [Element [0..1]	Значение содержащегося атрибута, используемое в случае, если ассоциированное значение `ExtensionAttributeDefinition.isReference` равно «true». Тип данного Элемента ДОЛЖЕН соответствовать типу, указанному в ассоциированном с ним элементе `ExtensionAttributeDefinition`.
extensionAttributeDefinition: ExtensionAttributeDefinition	Указывает атрибут расширения, для которого определяется данное значение.

Расширение XML-схем

Таблица 8.11 – Расширение XML-схемы

```
<xsd:element name="extension" type="tExtension"/>
<xsd:complexType name="tExtension">
      <xsd:sequence>
            <xsd:element ref="documentation" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="definition" type="xsd:QName"/>
      <xsd:attribute name="mustUnderstand" type="xsd:boolean" use="optional"/>
</xsd:complexType>
```

Пример XML

Приведенный ниже пример содержит описание **Задачи**, определенной в Ядре **BPMN**, которая была расширена с помощью Входных и Выходных данных, определенных уже за пределами Ядра.

Таблица 8.12 – Пример XML-схемы

```
<xsd:schema …>
        …
    <xsd:element name="task" type="tTask"/>
    <xsd:complexType name="tTask">
        <xsd:complexContent>
            <xsd:extension base="tActivity"/>
        </xsd:complexContent>
    </xsd:complexType>
        …
</xsd:schema>
```

Таблица 8.13 – Пример расширения XML-схемы

```
<xsd:schema …>
        …
    <xsd:group name="dataRequirements">
        <xsd:sequence>
            <xsd:element ref="dataInput" minOccurs="0" maxOccurs="unbounded" />
            <xsd:element ref="dataOutput" minOccurs="0" maxOccurs="unbounded" />
            <xsd:element ref="inputSet" minOccurs="0" maxOccurs="unbounded" />
            <xsd:element ref="outputSet" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:group>
        …
</xsd:schema>
```

Таблица 8.14 – Экземпляр XML-примера (XML instance)

```
<bpmn:definitions id="ID_1" …>
        …
    <bpmn:extension mustUnderstand="true" definition="bpmn:dataRequirements"/>
        …
    <bpmn:task name="Retrieve Customer Record" id="ID_2">
        <bpmn:dataInput name="Order Input" id="ID_3">
            <bpmn:typeDefinition typeRef="bo:Order" id="ID_4"/>
        </bpmn:dataInput>
        <bpmn:dataOutput name="Customer Record Output" id="ID_5">
            <bpmn:typeDefinition typeRef="bo:CustomerRecord" id="ID_6"/>
        </bpmn:dataOutput>
        <bpmn:inputSet name="Inputs" id="ID_7" dataInputRefs="ID_3"/>
        <bpmn:outputSet name="Outputs" id="ID_8" dataOutputRefs="ID_5"/>
    </bpmn:task>
        …
</bpmn:definitions>
```

8.2.4. Ссылки на внешние объекты

Одной из целей данного документа является обзор основных элементов, необходимых для построения синтаксически верных моделей **Процессов** с богатой семантикой. Такие модели могут использоваться для описания **Процессов** (**Processes**), **Хореографий** (**Choreographies**) и исполнительских действий различных уровней абстракции. Как подчеркивалось в данной спецификации, возможность расширения позволяет обогащать содержащиеся в **BPMN** данные, а также дополнять модели для выполнения их деталей.

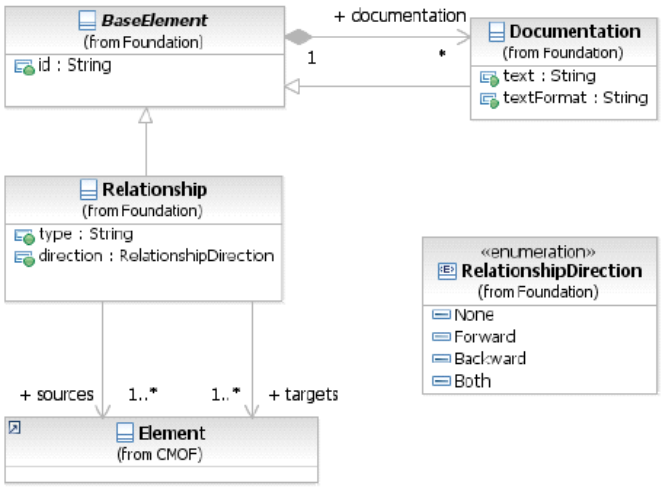
Модели **Процессов** не являются изолированными и, как правило, задействованы в более сложных **Бизнес-процессах** и **Процессах** развития систем. Нижеследующее описание элементов позволяет интегрировать `Артефакты (Artifacts)` **BPMN** в такие **Процессы** развития путем детализации ненавязчивой модели тождественности/взаимоотношений между `Артефактами (Artifacts)` **BPMN** и элементами другой адресуемой модели предметной области.

Модель «Идентификатор/Ссылка» упрощена для возможности создания типизированных отношений, относящихся к различным категориям. Благодаря этому устанавливаются свободные взаимоотношения между Артефактами (как относящимися к **BPMN**, так и не относящимися к нему). Для расширения моделей **BPMN** и их интеграции в более масштабные системные **Процессы** и **Процессы** развития необходимо лишь определить «тип взаимоотношения» («relationship type»), который можно ассоциировать с элементами `Артефактов` (как **BPMN**, так и `Артефактов`, применяемых в других типах моделирования).

Такие расширения способны, например, связывать различными способами взаимоотношения «происхождения» («derivation») и «определения» («definition») между артефактами UML и Артефактами **BPMN**. Согласно спецификации **BPMN**, элемент диаграммы UML может быть соотнесен с элементом **Процесса** без воздействия на сами `Артефакты`, что, однако, позволяет выполнять интеграцию моделей, относящихся к данным взаимоотношениям.

При моделировании применяется внешняя спецификация взаимоотношений расширения между `Артефактами` **BPMN** и другими моделями классификации взаимоотношений (внешними моделями). Такие внешние модели используются для установления связи между элементами **BPMN** и другими структурными и неструктурными определениями метаданных посредством пересечения отношений, указанных во внешнем определении.

Модель UML для данной спецификации аналогична простой гибкой модели (см. ниже), в которой указанные взаимоотношения могут быть установлены путем определения ссылки на соответствующие объекты, находящиеся в указанном пространстве имен.
The UML model for this specification follow a simple extensible pattern as shown below; where named relationships can be established by referencing objects that exist in their given namespaces.



Фигура 8.7 – Метамодель Внешних Взаимоотношений (External Relationship)

Элемент `Relationship` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.15 содержит информацию о дополнительных атрибутах элемента `Relationship`.

Таблица 8.15 – Атрибуты элемента Relationship

Название атрибута	Описание/использование
type: string	Описательное имя элемента.
direction: RelationshipDirection {None | Forward | Backward | Both}	Данный атрибут определяет направление взаимоотношений.
sources: [Element [1..*]	Данная ассоциация определяет артефакты, расширенные посредством взаимоотношений.
targets: [Element[1..*]	Данная ассоциация используется для расширения

	семантики исходного элемента(ов). This association defines artifacts used to extend the semantics of the source element(s).

Таким образом, вы можете, к примеру, создавать взаимоотношения между различными артефактами, что позволяет добавлять внешние комментарии, используемые для отслеживания, деривации, внешних классификаций и т.д. Примером таких взаимоотношений «реконструкции» могут служить отношения между артефактом Visio ™ и `Артефактом` **BPMN**.

Таблица 8.16 - XML-схема реконструкции

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace=""
      typeLanguage="" id="a123" expressionLanguage=""
      xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL Core-Common.xsd"
      xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:src="http://www.example.org/Processes/Old"
      xmlns:tgt="http://www.example.org/Processes/New">

      <import importType="http://office.microsoft.com/visio" location="OrderConfirmationProcess.vsd"
            namespace="http://www.example.org/Processes/Old"/>
      <import importType="http://www.omg.org/spec/BPMN/20100524/MODEL"
            location="OrderConfirmationProcess.xml"
            namespace="http://www.example.org/Processes/New"/>

      <relationship type="reengineered" id="a234" direction="both">
            <documentation>An as-is and to-be relationship. The as-is model is expressed as a Visio dia-
                  gram. The re-engineered process has been split in two and is captured in BPMN 2.0 for-
                  mat.</documentation>
            <source ref="src:OrderConfirmation"/>
            <target ref="tgt:OrderConfirmation_PartI"/> <target
            ref="tgt:OrderConfirmation_PartII"/>
      </relationship>
</definitions>
```

8.2.5. Корневой элемент

`RootElement` является абстрактным суперклассом для всех элементов **BPMN**, содержащихся в элементах `Definitions`. Каждому из таких элементов соответствует определенный жизненный цикл. Эти элементы не могут быть удалены, даже если другие элементы подвергаются удалению. Примерами корневых элементов (`RootElements`) являются **Взаимодействие** (**Collaboration**), **Процесс** (**Process**) и **Хореография** (**Choreography**). В зависимости от вариантов использования, на корневые элементы (`RootElements`) могут ссылаться множество других элементов (к примеру, они могут использоваться несколько раз). Некоторые элементы `RootElements` МОГУТ содержаться в других элементах, а не только в элементах `Definitions`; это было сделано во избежание сохранения побочных явлений независимых жизненных циклов. Элемент `EventDefinition`, к примеру, хранится в **Процессе** (**Process**), поскольку используется только в нем. В этом случае элемент `EventDefinition` будет зависеть от жизненного цикла **Процесса** (**Process**).

Элемент RootElement наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5), однако, для него не может быть определено никаких других атрибутов и ассоциаций.

8.2.6. Представление XML-схем для пакета Foundation (Foundation Package)

Таблица 8.17 – XML-схема для элемента BaseElement

```
<xsd:element name="baseElement" type="tBaseElement"/>
<xsd:complex Type name="tBaseElement" abstract="true">
      <xsd:sequence>
            <xsd:element ref="documentation" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="extensionElements" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
```

```
        <xsd:attribute name="id" type="xsd:ID" use="optional"/>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>

<xsd:element name="baseElementWithMixedContent" type="tBaseElementWithMixedContent"/>
<xsd:complexType name="tBaseElementWithMixedContent" abstract="true" mixed="true">
        <xsd:sequence>
                <xsd:element ref="documentation" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="extensionElements" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:ID" use="optional"/>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>

<xsd:element name="extensionElements" type="tExtensionElements"/>
<xsd:complexType name="tExtensionElements">
        <xsd:sequence>
                <xsd:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
</xsd:complexType>


<xsd:element name="documentation" type="tDocumentation"/>
<xsd:complexType name="tDocumentation" mixed="true">
        <xsd:sequence>
                <xsd:any namespace="##any" processContents="lax" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:ID" use="optional"/>
        <xsd:attribute name="textFormat" type="xsd:string" default="textplain"/>
</xsd:complexType>
```

Таблица 8.18 – XML-схема для элемента RootElement

```
<xsd:element name="rootElement" type="tRootElement"/>
<xsd:complexType name="tRootElement" abstract="true">
        <xsd:complexContent>
                <xsd:extension base="tBaseElement"/>
        </xsd:complexContent>
</xsd:complexType >
```

Таблица 8.19 – XML-схема для элемента Relationship

```
<xsd:element name="relationship" type="tRelationship"/>
<xsd:complexType name="tRelationship">
        <xsd:complexContent>
                <xsd:extension base="tBaseElement">
                        <xsd:sequence>
                                <xsd:element name="source" type="xsd:QName" minOccurs="1" maxOccurs="unbounded"/> <xsd:element
                                name="target" type="xsd:QName" minOccurs="1" maxOccurs="unbounded"/>
                        </xsd:sequence>
                        <xsd:attribute name="type" type="xsd:string" use="required"/>
                        <xsd:attribute name="direction" type="tRelationshipDirection"/>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tRelationshipDirection">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="None"/>
                <xsd:enumeration value="Forward"/>
                <xsd:enumeration value="Backward"/>
                <xsd:enumeration value="Both"/>
        </xsd:restriction>
</xsd:simpleType>
```

8.3. Общие элементы (Common Elements)

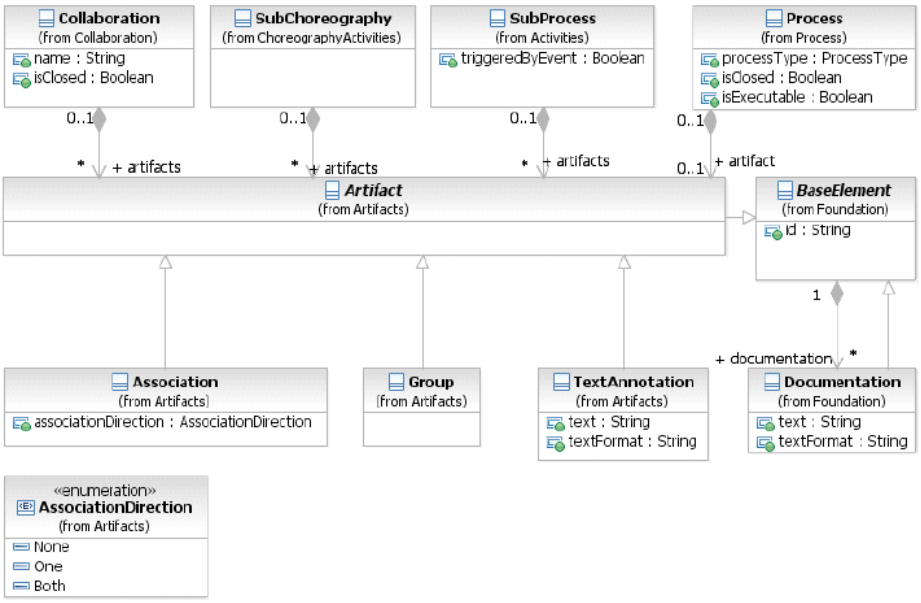
В следующих подразделах содержится описание элементов **BPMN**, которые МОГУТ добавляться на диаграммы разных типов (**Процесс** (**Process**), **Взаимодействие** (**Collaboration**), и **Хореография** (**Choreography**)).

8.3.1. Артефакты (Artifacts)

Для разработчиков моделей в **BPMN** предусмотрена возможность внесения дополнительной информации о **Процессе (Process)**. Такая информация не связана непосредственно с **Потоками Операций (Sequence Flow)** или **Потоками Сообщений (Message Flow)** этого **Процесса**.

В данном документе представлено три типа стандартных `Артефактов`: **Ассоциация (Association), Группа** (**Group**) и **Текстовая Аннотация (Text Annotation)**, что, однако, не исключает возможности добавления других `Артефактов` в последующих версиях спецификации (т.е. в будущем МОГУТ БЫТЬ добавлены новые `Артефакты`). Разработчики моделей или инструменты моделирования МОГУТ расширять диаграммы **BPMN**, добавляя при этом на Диаграммы собственные типы `Артефактов`. Любой искусственно добавленный `Артефакт` ДОЛЖЕН соответствовать правилам соединения с **Потоком Операций** и **Потоком Сообщений** (см. ниже). Для соединения `Артефактов` с *Элементами Потока (Flow Objects)* используется **Ассоциация (Association)** (для получения более подробной информации см. таблицу 7.2).

На фигуре 8.8 отображена диаграмма класса `Artifacts. Артефакт` хранится либо во **Взаимодействии** (**Collaboration**), либо в элементе `FlowElementsContainer` (**Процесса (Process)** или **Хореографии (Choreography)**).



Фигура 8.8 – Метамодель Артефакта

Общие определения для Артефактов

В следующих подразделах представлена информация об определениях, общих для всех `Артефактов`.

Соединение Артефакта с Потоком Операций

Для получения более подробной информации об элементах **BPMN**, которые МОГУТ являться как источниками, так и целями **Потоков Операций** (**Sequence Flow**), см. подраздел 7.5.1.

- `Артефакты` НЕ ДОЛЖНЫ являться целями **Потоков Операций**.
- `Артефакты` НЕ ДОЛЖНЫ служить источниками **Потоков Операций**.

Соединение Артефакта с Потоком Сообщений

Для получения более подробной информации об элементах **BPMN**, которые МОГУТ являться как источниками, так и целями **Потоков Сообщений** (**Message Flow**), см. подраздел 7.5.2.

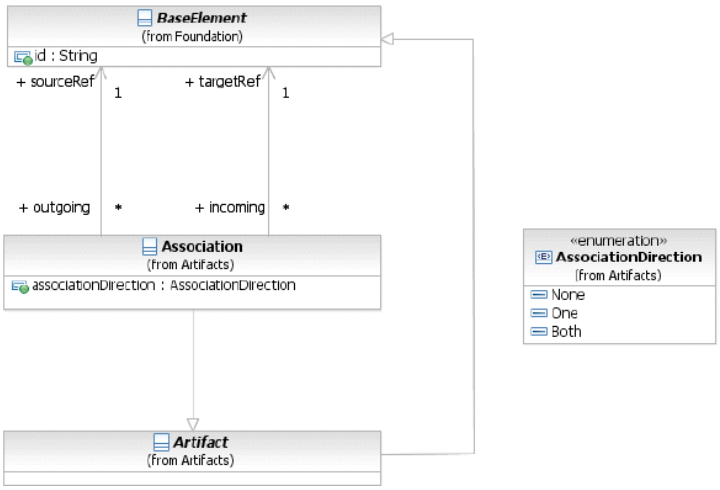
- Артефакты НЕ ДОЛЖНЫ являться целями **Потоков Сообщений**.
- Артефакты НЕ ДОЛЖНЫ служить источниками **Потоков Сообщений**.

Ассоциация (Association)

Ассоциация (**Association**) используется для установки соответствия между какой-либо информацией и Артефактом и *Элементами Потока* (*Flow Objects*). Текстовые и графические элементы, не являющиеся *Элементами Потока*, также могут быть ассоциированы с *Элементами Потока* и каким-либо Потоком операций. **Ассоциация** используется и для отображения **Действия** (**Activity**) *компенсации*. Для получения более подробной информации о *компенсации* см. раздел 10.6.

- **Ассоциация** должна быть выполнена пунктирной линией (см. фигуру 8.9).
- Текст, цвет, размер, а также линии, используемые для изображения **Ассоциации**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».

Фигура 8.9 – Ассоциация (Association)



Фигура 8.10 – Диаграмма класса Association

Для того чтобы отобразить направление **Ассоциации**, необходимо следовать правилам:

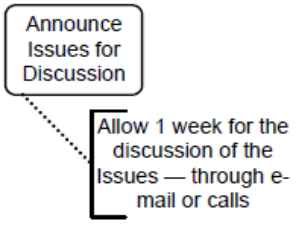
- Графический элемент **Ассоциация** МОЖЕТ отображаться со стрелкой (см. фигуру 8.11).
  - **Ассоциация** может иметь одно (1) или два направления.

Фигура 8.11 – Направленная Ассоциация

Обратите внимание, что описанная в **BPMN 1.2** направленная **Ассоциация** использовалась для отображения того, каким образом **Объекты Данных** (**Data Object**) входили в **Действия** (**Activity**) или выходили из них. В **BPMN 2.0** для этого используется **Ассоциация Данных** (**Data Association**) (для получения более подробной

информации см. фигуру 10.64). При добавлении на диаграмму **Ассоциации Данных** используется та же нотация, что и для добавления направленной **Ассоциации** (см. фигуру 8.11).

Ассоциация используется для соединения указанного пользователем текста (**Аннотации** (**Annotation**)) с *Элементами Потока* (*Flow Objects*) (см. фигуру 8.12).



Фигура 8.12 – Ассоциация, применяемая к Текстовой Аннотации

Элемент **Association** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.20 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Association**.

Таблица 8.20 – Атрибуты и ассоциации элемента Association

Название атрибута	Описание/использование
associationDirection: AssociationDirection = None {None | One | Both}	Данный атрибут указывает на то, будет ли Ассоциация иметь направление (т.е. будет ли отображаться со стрелкой). Значением по умолчанию является «`None`» (подразумевает отсутствие стрелки). Значение «`One`» используется для указания того, что **Ассоциация** ДОЛЖНА БЫТЬ направлена к Элементу, являющемуся Источником. Значение «`Both`» указывает на то, что стрелка ДОЛЖНА отображаться на обоих концах графического элемента **Ассоциация**.
sourceRef: BaseElement	Определяет элемент `BaseElement`, от которого направлена **Ассоциация**.
targetRef: BaseElement	Определяет элемент `BaseElement`, к которому направлена **Ассоциация**.

Группа (Group)

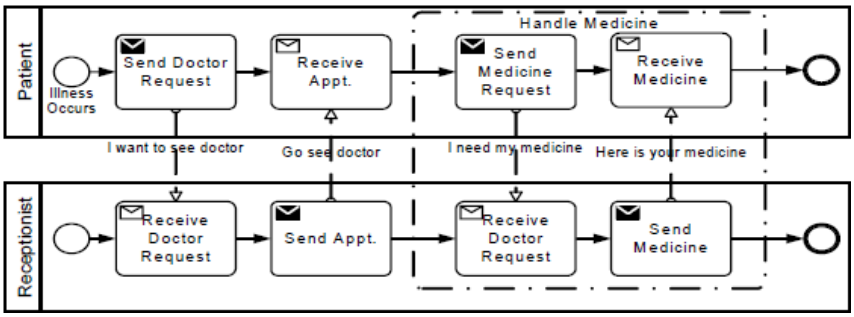
Элемент **Группа** (**Group**) является `Артефактом`, служащим в качестве неформального механизма отображения группы элементов на диаграмме. Такая группировка связана со значением `CategoryValue` соответствующего элемента. Это означает, что **Группа** представляет собой визуальное отображение конкретного значения `CategoryValue`.

Примечание: Значение `CategoryValue` также может быть выделено другими способами, например, цветом, выбранным разработчиком модели или инструментом моделирования.

- **Группа** представляет собой прямоугольник с закругленными краями, который ДОЛЖЕН БЫТЬ выполнен жирным пунктиром (см. фигуру 8.13).
    - Текст, цвет, размер, а также линии, используемые для изображения **Группы**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».

Фигура 8.13 – Артефакт типа Группа

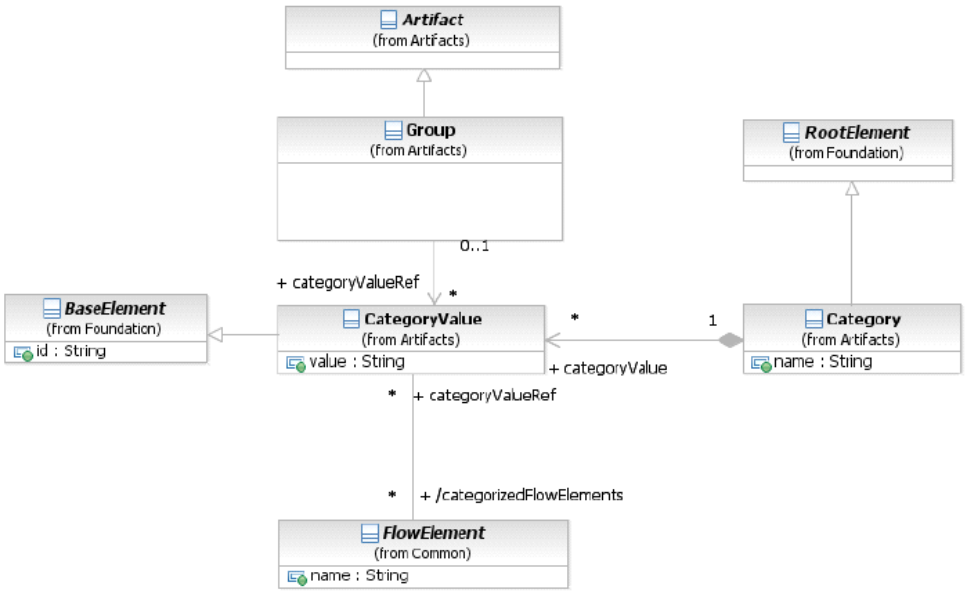
Как и **Артефакт**, **Группа** не является **Действием** (**Activity**) или одним из `Элементов Потока (Flow Object)`, поэтому данный графический элемент не может быть соединен с **Потоком Операций** (**Sequence Flow**) или с **Потоком Сообщений** (**Message Flow**). Ограничения использования **Пулов** (**Pool**) и **Дорожек** (**Lane**) не распространяются на использование **Групп**. Это означает, что для объединения элементов **Диаграммы Группа** может простираться за границы **Пула** (см. фигуру 8.14). В таком качестве **Группа** используется для отображения **Действий** (**Activity**), являющихся частью масштабных взаимоотношений типа B2B (business-to-business).



Фигура 8.14 – Группа, объединяющая Действия из разных Пулов

Группы обычно используются для выделения областей Диаграммы, при этом на их использование не налагается никаких ограничений (применяемых по отношению к **Подпроцессу** (**Sub-Process**)). Выделенная область (сгруппированные элементы) Диаграммы может быть разделена в целях создания отчетности и произведения анализа. **Группы** не оказывают влияния на ход `Процесса (Process)`.

На фигуре 8.15 отображена диаграмма класса **Group**.



Фигура 8.15 – Диаграмма классов элемента Group

Элемент **Group** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством связи с элементом **Atrifact**. Таблица 8.21 содержит информацию о дополнительных ассоциациях элемента **Group**.

Таблица 8.21 – Ассоциации элемента Group

Название атрибута	Описание/использование
categoryValueRef: Category- Value [0..1]	Данный атрибут указывает значение `categoryValueRef`, которое представлено **Группой**. Для получения более подробной информации об элементах `Category` `CategoryValue` см. следующий подраздел. Название **Группы** состоит из названия элемента `Category` и значения `CategoryValue`, разделенных знаком «.» (точкой). Значения `CategoryValue` назначаются на графические элементы, расположенные внутри **Группы**.

Категория (Category)

Категории, семантика для которых определяется пользователем, могут использоваться для документирования или анализа. К примеру, элементы `FlowElements` могут быть разбиты на категории, включающие ориентированные на заказчика элементы и элементы, ориентированные на поддержку. Кроме того, для каждой из **Категорий** могут быть подсчитаны стоимость и время выполнения **Действий** (**Activity**).

Группы являются одним из способов, при помощи которых `Категории` объектов отображаются на диаграмме. Это означает, что **Группа** является визуальным отображением конкретного значения `CategoryValue`. Графические элементы, заключенные в **Группы**, назначаются на конкретное значение `CategoryValue` данной Группы. Значение `CategoryValue`, перед которым указано название `Категории` и стоит разделитель «:», отображается на диаграмме в качестве названия **Группы**. Обратите внимание, что Категории могут быть выделены другими способами, например, цветом, выбранным разработчиком модели или инструментом моделирования. На диаграмме одна Категория может использоваться в нескольких **Группах**.

Элемент `Category` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством связи с элементом `RootElement`. Таблица 8.22 содержит информацию о дополнительных ассоциациях элемента `Category`.

Таблица 8.22 – Ассоциации элемента Category

Название атрибута	Описание/использование
name: string	Описательное имя элемента.
categoryValue: CategoryValue [0..*]	Посредством атрибута `categoryValue` указывается одно или более значений элемента `Category`. К примеру, если `Category` означает «Регион», то для неё могут быть установлены следующие значения: «север», «юг», «запад», «восток».

Элемент `CategoryValue` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.23 содержит информацию о дополнительных атрибутах и ассоциациях элемента `CategoryValue`.

Таблица 8.23 – Атрибуты и ассоциации элемента CategoryValue

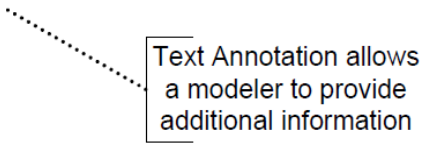
Название атрибута	Описание/использование
value: string	Посредством данного атрибута устанавливается значение элемента `CategoryValue`.
category: Category [0..1]	Данный атрибут используется для определения `Категории` как таковой и содержит значение `CategoryValue` (для получения более подробной информации о `Категории` см. Описание выше).
categorizedFlowElements: FlowElement [0..*]	Посредством данного атрибута указываются все элементы (например, `События (Events)`, `Действия (Activities)`, `Шлюзы (Gateways)` и `Артефакты (Artifacts)`), заключенные в `Группу`.

Текстовая Аннотация (Text Annotation)

Текстовая Аннотация представляет собой механизм, при помощи которого разработчик модели может добавлять на Диаграмму **BPMN** дополнительную информацию, являющуюся важной для конечного пользователя Диаграммы.

- **Текстовая Аннотация** представляет собой негерметичный прямоугольник, который ДОЛЖЕН БЫТЬ выполнен одинарной жирной линией (см. фигуру 8.16).
    - Текст, цвет, размер, а также линии, используемые для изображения **Текстовой Аннотации**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».

Графический элемент **Текстовая Аннотация** может быть присоединен к определенному элементу на Диаграмме при помощи **Ассоциации** (**Association**), однако, он не оказывает влияния на ход **Процесса**. Текст, ассоциированный с **Аннотацией**, может располагаться в пределах данного графического элемента.



Фигура 8.16 – Текстовая Аннотация

Элемент **Text Annotation** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.24 содержит информацию о дополнительных атрибутах элемента **Text Annotation**.

Таблица 8.24 – Атрибуты элемента Text Annotation

Название атрибута	Описание/использование
text: string	Данный атрибут представляет собой текст, который разработчик модели желает сообщить конечному пользователю Диаграммы.
textFormat: string	Посредством данного атрибута указывается формат текста. Его значение ДОЛЖНО БЫТЬ указано в формате MIME-тип. Значением по умолчанию является «text/plain».

Представление XML-схемы для Артефактов

Таблица 8.25 - XML-схема для элемента Artifact

```
<xsd:element name="artifact" type="tArtifact"/>
<xsd:complexType name="tArtifact" abstract="true">
	<xsd:complexContent>
		<xsd:extension base="tBaseElement"/>
	</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.26 - XML-схема для элемента Association

```
<xsd:element name="association" type="tAssociation" substitutionGroup="artifact"/>
<xsd:complexType name="tAssociation">
	<xsd:complexContent>
		<xsd:extension base="tArtifact">
			<xsd:attribute name="sourceRef" type="xsd:QName" use="required"/>
			<xsd:attribute name="targetRef" type="xsd:QName" use="required"/>
			<xsd:attribute name="associationDirection" type="tAssociationDirection" default="None"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tAssociationDirection">
	<xsd:restriction base="xsd:string">
		<xsd:enumeration value="None"/>
		<xsd:enumeration value="One"/>
		<xsd:enumeration value="Both"/>
		</xsd:restriction>
</xsd:simpleType>
```

Таблица 8.27 - XML-схема для элемента Category

```
<xsd:element name="category" type="tCategory" substitutionGroup="rootElement"/>
<xsd:complexType name="tCategory">

	<xsd:complexContent>
		<xsd:extension base="tRootElement">
			<xsd:sequence>
				<xsd:element ref="categoryValue" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="name" type="xsd:string"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.28 - XML-схема для элемента CategoryValue

```
<xsd:element name="categoryValue" type="tCategoryValue"/>
<xsd:complexType name="tCategoryValue">
	<xsd:complexContent>
		<xsd:extension base="tBaseElement">
			<xsd:attribute name="value" type="xsd:string" use="optional"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.29 - XML-схема для элемента Group

```
<xsd:element name="group" type="tGroup" substitutionGroup="artifact"/>
<xsd:complexType name="tGroup">
	<xsd:complexContent>
		<xsd:extension base="tArtifact">
			<xsd:attribute name="categoryValueRef" type="xsd:QName" use="optional"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.30 - XML-схема для элемента Text Annotation

```
<xsd:element name="textAnnotation" type="tTextAnnotation" substitutionGroup="artifact"/>
<xsd:complexType name="tTextAnnotation">
    <xsd:complexContent>
        <xsd:extension base="tArtifact">
            <xsd:sequence>
                <xsd:element ref="text" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="textFormat" type="xsd:string" default="textplain"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="text" type="tText"/>
<xsd:complexType name="tText" mixed="true">
    <xsd:sequence>
        <xsd:any namespace="##any" processContents="lax" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

8.3.2. Корреляция (Correlation)

Выполнение **Бизнес-Процесса** может осуществляться в течение нескольких дней или даже месяцев, сопровождаясь асинхронным общением через **Сообщения** (**Message**). В то же время параллельно могут выполняться множество *экземпляров* (*instance*) конкретного **Процесса**, к примеру, множество экземпляров процесса заказа, каждый из которых представляет собой отдельно взятый заказ. *Корреляция* используется для установки соответствия между каким-либо **Сообщением** и *исходящим* **Обменом сообщениями** (**Conversation**) двух различных *экземпляров* **Процесса**. **BPMN** позволяет использовать данные из существующих **Сообщений** для установки такой *корреляции*. К примеру, в процессе заказа каждый отдельно взятый экземпляр этого процесса может быть идентифицирован посредством относящегося к нему атрибута ordered или customerID, а не путем запроса технических данных о *корреляции*.

Суть *Корреляции* заключается в ассоциировании **Сообщения** с **Задачей** типа **Отправка** (**Send Task**) или **Задачей** типа **Получение** (**Receive Task**), что также называется *маршрутизацией экземпляра* (*instance routing*). *Корреляция* особенно важна, когда для осуществления маршрутизации *экземпляра* не может быть использована поддержки инфраструктура. Обратите внимание, что такая ассоциация может просматриваться на различных уровнях моделирования, т.е. во **Взаимодействии** (**Collaboration** (**Conversation**)), **Хореографии** (**Choreography**) и **Процессе** (**Process**). Однако собственно *корреляция* выполняется в ходе выполнения процесса (к примеру, на одном из уровней **Процесса**). Посредством *корреляций* описывается набор утверждений, касаемых **Сообщения** (как правило, передаваемых приложением данных). Эти утверждения должны быть удовлетворены, иначе **Сообщение** не будет ассоциировано с обособленной **Задачей Отправка** или **Задачей Получение**. Кроме того, любая **Задача Отправка** и любая **Задача Получение** может участвовать в одном или более **Обменах сообщениями** (**Conversation**). К тому же **Задача** такого типа идентифицирует **Сообщение**, которое она отсылает, и, таким образом, устанавливает связь с одним или несколькими `Ключами Корреляции - CorrelationKeys.`

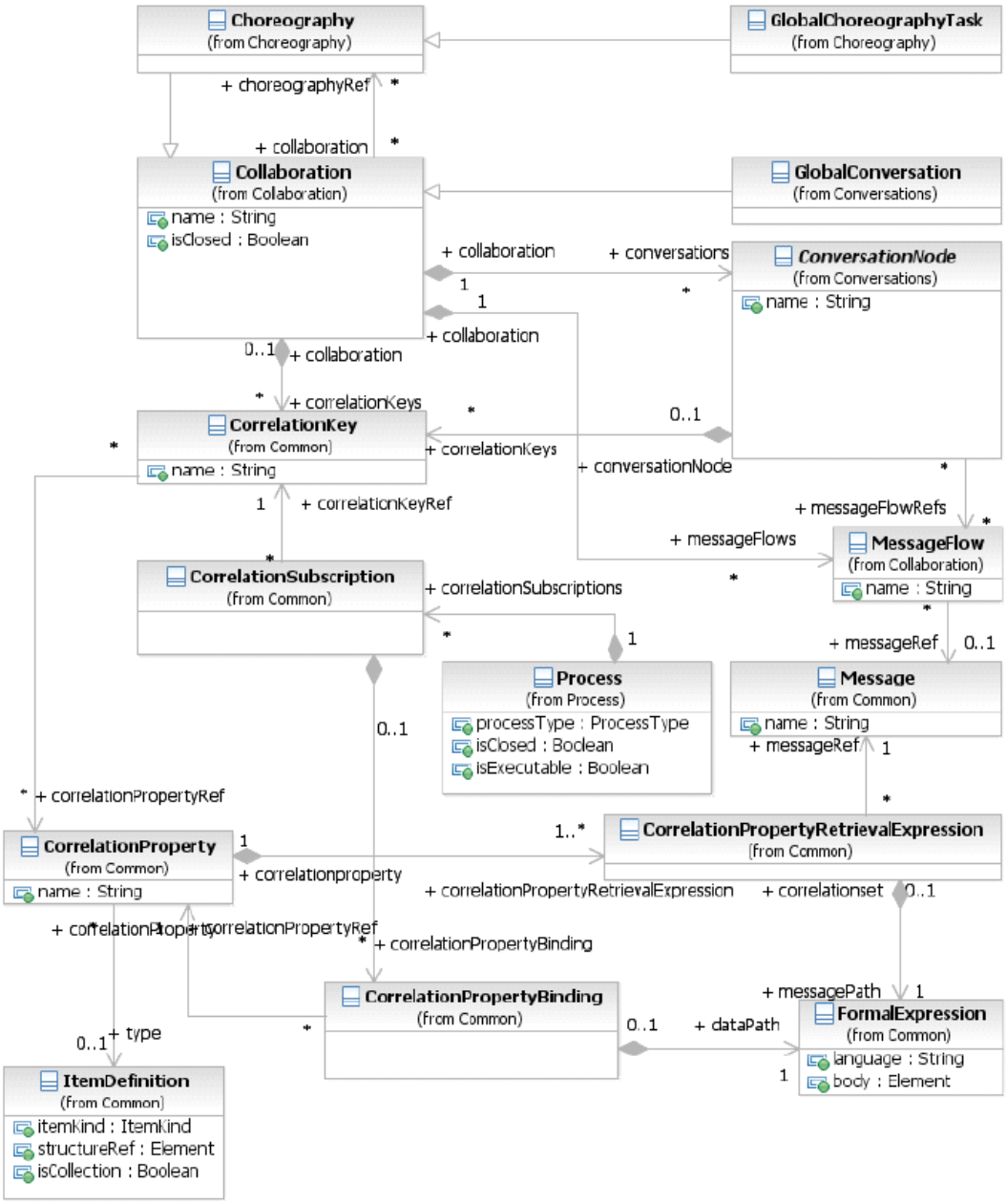
Существуют два механизма неэксклюзивной корреляции:

1. Корреляция, основанная на ключах. В данном случае между **Сообщениями**, которые были отправлены и получены в ходе **Обмена сообщениями** (**Conversation**), устанавливается логическая корреляция посредством одного или более общих ключей `CorrelationKeys.` Это означает, что в любом **Сообщении**, которое отсылается или принимается в ходе **Обмена сообщениями** (**Conversation**), должно храниться значение по-меньшей мере одного экземпляра `CorrelationKey`, входящее в передаваемые им данные. Посредством `CorrelationKey` указывается ключ (составной) (key/composite

key). **Сообщение**, которое отсылается или принимается первым, инициализирует один или более экземпляров `CorrelationKey`, ассоциированных с **Обменом сообщениями** (**Conversation**). Другими словами, оно указывает значения для экземпляров `CorrelationProperty`, являющихся полями/частичными ключами (fields/partial keys) элементов `CorrelationKey`. `CorrelationKey` может быть использован лишь в случае, если **Сообщение** оказало влияние на все поля `CorrelationProperty` ключа, содержащего значение. Если следующее **Сообщение** получает экземпляр `CorrelationKey` (с учетом, что данный `CorrelationKey` был уже инициализирован в **Обмене сообщениями** (**Conversation**)), то значение `CorrelationKey`, содержащееся в **Сообщении**, ДОЛЖНО соответствовать значению `CorrelationKey`, содержащемуся в **Обмене сообщениями**. Если же следующее **Сообщение** получает экземпляр CorrelationKey, ассоциированный с **Обменом сообщениями** (**Conversation**), однако, предварительно не инициализированный, то значение `CorrelationKey` становится ассоциированным с **Обменом сообщениями**. Поскольку в **Обмен сообщениями** (**Conversation**) могут входить различные **Сообщения**, его структура также может быть разной, а каждое значение `Свойства Корреляции - CorrelationProperty -` сопровождается количеством правил извлечения (`CorrelationPropertyRetrievalExpression`) для соответствующих частичных ключей, соответствующим количеству **Сообщений.**

2. Корреляция, основанная на контексте. В данном случае контекст **Процесса** (например, **Объекты данных** (**Data Object**) или `Свойства (Properties)`) могут динамически воздействовать на критерии соответствия. Это означает, что `CorrelationKey` может быть дополнен `CorrelationSubscription`, относящимся к **Процессу**. `CorrelationSubscription` объединяет столько элементов `CorrelationPropertyBindings`, сколько `CorrelationProperties` содержится в данном `CorrelationKey`. `CorrelationPropertyBinding` связан с конкретным `CorrelationProperty`, а также ссылается на `Формальное Выражение - FormalExpression`, которое указывает на правило динамического извлечения, действующее поверх контекста **Процесса**. В ходе выполнения процесса экземпляр `CorrelationKey` для отдельно взятого **Обмена сообщениями** (**Conversation**) распространяется (и динамически обновляется), исходя из контекста **Процесса** и используя `FormalExpressions` (`формальные выражения`). В этом смысле изменения в контексте **Процесса** могут менять условия корреляции.

Корреляция может быть использована в **Потоках Сообщений** (**Message Flows**), входящих в состав **Взаимодействия** (**Collaboration**) и **Хореографии** (**Choreography**) (для получения более подробной информации см. Главы 9 и 11 соответственно). Ключи, используемые в **Потоках Сообщений**, представляют собой ключи контейнеров или группировок **Потоков Сообщений**, представленными в виде **Взаимодействия**, **Хореографии**, **Точек Обмена Сообщениями** (**Conversation Nodes**) и **Действий Хореографии** (**Choreography Activities**). Результатом могут стать множественные `CorrelationKeys`, используемые в том же самом **Потоке Сообщений** (возможно, из-за многочисленных слоев ограничения). В частности, вызовы **Взаимодействия** и **Хореографии** представляют собой особые **Точки Обмена Сообщениями** (**Conversation Nodes**) и **Действия Хореографии** соответственно. Они рассматриваются в качестве ограничений, необходимых для *корреляции*. Значения `CorrelationKeys`, заданные в вызывающем операторе, используются в **Потоках Сообщений** вызываемого **Взаимодействия** или вызываемой **Хореографии**.



Фигура 8.17 – Диаграмма классов элемента Correlation

Ключ Корреляции (CorrelationKey)

`Ключ Корреляции (CorrelationKey)` представляет собой составной ключ из одного или более `Свойств Корреляции (CorrelationProperties)`, который главным образом определяет извлечение `Выражений (Expressions)` поверх **Сообщений**. В результате каждое `Свойство Корреляции` выступает в роли частичного ключа *корреляции*. В любом **Сообщении**, которое отсылается или принимается как часть какого-то **Обмена сообщениями** (**Conversation**), `Свойства Корреляции` должны указывать выражение `CorrelationPropertyRetrievalExpression`, которое устанавливает связь (ссылку) между `Формальным Выражением (FormalExpression)` и передаваемыми **Сообщением** данными. Это означает, что для каждого **Сообщения**, используемого в **Обмене Сообщениями**, существует `Выражение (Expression)`, которое извлекает соответствующие данные, передаваемые в **Сообщении**.

Элемент CorrelationKey наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 8.31 содержит информацию о дополнительных ассоциациях элемента CorrelationKey.

Таблица 8.31 – Ассоциации элемента CorrelationKey

Название атрибута	Описание/использование
name: string [0..1]	Указывает название элемента CorrelationKey.
correlationPropertyRef: CorrelationProperty [0..*]	Свойства корреляции (CorrelationProperties), представляющие собой частичные ключи данного Ключа Корреляции (CorrelationKey).

Основанная на ключах Корреляция (Key-based Correlation)

Основанная на ключах *корреляция* представляет собой простую и эффективную форму *корреляции*, при которой для идентификации **Обмена Сообщениями** (**Conversation**) используются один или несколько ключей. Любое входящее **Сообщение** (**Message**) должно быть противопоставлено Ключу Корреляции (CorrelationKey) путем извлечения Свойств Корреляции (CorrelationProperties) из этого **Сообщения** в соответствии с выражением CorrelationPropertyRetrievalExpression. При этом для данного **Обмена Сообщениями** выполняется сравнение составного ключа с экземпляром Ключа Корреляции (CorrelationKey). Смысл заключается в использовании объединенного «токена» (token), как отправляемого, так и получаемого, а также *исходящих* (*outgoing*) и *входящих* (*incoming*) **Сообщений**. **Сообщения** ассоциируются с каким-либо **Обменом Сообщениями** (**Conversation**) в том случае, если составной ключ, извлеченный из передаваемых данных, соответствует Ключу Корреляции (CorrelationKey), инициализированному для данного **Обмена Сообщениями**.

Задача Отправка (**Send Task**) или **Задача получение** (**Receive Task**), стоящая первой в составе **Обмена Сообщениями** (**Conversation**), ДОЛЖНА распространить по-меньшей мере один из экземпляров Ключа Корреляции (CorrelationKey) за счет извлечения значений из Свойств Корреляции (CorrelationProperties) в соответствии с выражением CorrelationPropertyRetrievalExpression, содержащимся в предварительно переданном или полученном **Сообщении**. Затем распространенные в **Обмене Сообщениями** экземпляры Ключа Корреляции (CorrelationKey) используются для описываемой процедуры согласования, когда составной ключ извлекается из входящих **Сообщений** и используется для идентификации соответствующего **Обмена Сообщениями** (**Conversation**). Когда такие, не способные к инициации Сообщения являются источникам значений для Ключей Корреляции (CorrelationKey), ассоциированных с **Обменом Сообщениями**, однако, ещё не распространенных, тогда извлеченные значения будут ассоциированы с экземпляром данного **Обмена Сообщениями**.

Элемент CorrelationProperty наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5) посредством связи с элементом RootElement. Таблица 8.32 содержит информацию о дополнительных ассоциациях элемента CorrelationProperty.

Таблица 8.32 – Ассоциации элемента CorrelationProperty

Название атрибута	Описание/использование
name: string [0..1]	Указывает название элемента CorrelationProperty.
type: string [0..1]	Указывает тип элемента CorrelationProperty.
correlationPropertyRetrieval- Expression: CorrelationPropertyRetrieval- Expression [1..*]	Представляет собой выражение CorrelationPropertyRetrievalExpressions для данного Ключа Корреляции (CorrelationKey), представляющее, в свою

	очередь, ассоциации Формальных Выражений (FormalExpressions), или путей извлечения, с Сообщениями, входящими в состав данного **Обмена Сообщениями** (**Conversation**).

Элемент CorrelationPropertyRetrievalExpression наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 8.33 содержит информацию о дополнительных ассоциациях элемента CorrelationPropertyRetrievalExpressions.

Таблица 8.33 – Ассоциации элемента CorrelationPropertyRetrievalExpression

Название атрибута	Описание/использование
messagePath: FormalExpression	Формальное Выражение (FormalExpression), определяющее то, каким образом Свойство Корреляции (CorrelationProperty) извлекается из передаваемых в Сообщении данных.
messageRef: Message	Конкретное Сообщение, из которого Формальное Выражение (FormalExpression) извлекает Свойство Корреляции (CorrelationProperty).

Основанная на контексте Корреляция (Context-based Correlation)

По сравнению с *корреляцией*, основанной на ключах, *корреляция*, основанная на контексте, является более точной. В дополнении к скрытому распространению *экземпляра* Ключа Корреляции (CorrelationKey), начиная с первого отправленного или полученного **Сообщения** (**Message**), добавляется ещё один механизм установления связи между Ключом Корреляции (CorrelationKey) и контекстом **Процесса**. Это означает, что **Процесс** МОЖЕТ предоставить Подписку на Корреляцию (CorrelationSubscription), которая по отношению к Ключу Корреляции (CorrelationKey) выполняет функцию специального дубликата. В данном случае **Обмен сообщениями** (**Conversation**) МОЖЕТ дополнительно ссылаться на данные контекста **Процесса**, которые скрыто обновляются. Это позволяет определить, является ли получение **Сообщения** необходимым или нет. В ходе выполнения экземпляр Ключа Корреляции (CorrelationKey) хранит составной ключ, значение которого динамически высчитывается, исходя из контекста Процесса, а также автоматически обновляется, независимо от того, когда изменяются расположенные ниже **Объекты Данных** (**Data Object**) или Свойства (Properties).

Элементы CorrelationPropertyBindings представляют собой частичные ключи Подписки на корреляцию (CorrelationSubscription), каждый из которых относится к какому-то Свойству корреляции (CorrelationProperty) соответствующего Ключа Корреляции (CorrelationKey). Посредством Формального выражения (FormalExpression) определяется то, каким образом, исходя из контекста **Процесса**, распространяется и обновляется *экземпляр* Свойства Корреляции (CorrelationProperty) в ходе выполнения.

Элемент CorrelationSubscription наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 8.34 содержит информацию о дополнительных ассоциациях элемента CorrelationSubscription.

Таблица 8.34 – Ассоциации элемента CorrelationSubscription

Название атрибута	Описание/использование
correlationKeyRef: CorrelationKey	Ключ Корреляции (CorrelationKey), на который ссылается элемент

	CorrelationSubscription.
correlationPropertyBinding: CorrelationPropertyBinding [0..*]	Связь со Свойствами Корреляции (CorrelationProperties) и Формальными Выражениями (FormalExpressions) (правила извлечения поверх контекста Процесса).

Элемент CorrelationPropertyBinding наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 8.35 содержит информацию о дополнительных ассоциациях элемента CorrelationPropertyBinding.

Таблица 8.35 – Ассоциации элемента CorrelationPropertyBinding

Название атрибута	Описание/использование
dataPath: FormalExpression	Формальное Выражение (FormalExpression), определяющее правило извлечения поверх контекста Процесса.
correlationPropertyRef: CorrelationProperty	Свойства Корреляции (CorrelationProperties), на которые ссылается данный элемент CorrelationPropertyBinding.

В ходе выполнения механизм корреляции работает следующим образом: когда создается экземпляр **Процесса**, экземпляры Ключа Корреляции (CorrelationKey) во всех **Обменах Сообщениями** (**Conversations**) инициализируются вместе с некоторыми начальными значениями, посредством которых указывается корреляция с *любым* входящим **Сообщением** (**Message**) данных **Обменов Сообщениями**. Свойство Подписки (SubscriptionProperty) обновляется независимо от того, когда меняется любой **Объект Данных** (**Data Objects**) или Свойство (Properties), на которые в соответствующем Формальном Выражении (FormalExpression) есть ссылка. В результате, входящее **Сообщение** противопоставлено экземпляру Ключа Корреляции (CorrelationKey). Дальше, по ходу следования маршрута **Процесса**, Свойства Подписки (SubscriptionProperties) опять-таки могут изменить (в том числе, и скрыто) критерии корреляции. В качестве альтернативы может быть использован установленный механизм получения первой **Задачи Отправка** (**Send Task**) или **Задачи Получение** (**Receive Task**), используемых для распространения *экземпляра* Ключа Корреляции (CorrelationKey).

Представление XML-схемы для Корреляции (**Correlation**)

Таблица 8.36 – XML-схема для элемента Correlation Key

```
<xsd:element name="correlationKey" type="tCorrelationKey"/>
<xsd:complexType name="tCorrelationKey">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="correlationPropertyRef" type="xsd:QName" minOccurs="0" maxOc-
                        curs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:String" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.37 – XML-схема для элемента Correlation Property

```
<xsd:element name="correlationProperty" type="tCorrelationProperty" substitutionGroup="rootElement"/>
<xsd:complexType name="tCorrelationProperty">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
```

```
                <xsd:sequence>
                    <xsd:element ref="correlationPropertyRetrievalExpression" minOccurs="1" maxOc-
                            curs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:String" use="optional"/>
              <xsd:attribute name="type" type="xsd:QName"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.38 – XML-схема для элемента Correlation Property Binding

```
<xsd:element name="correlationPropertyBinding" type="tCorrelationPropertyBinding"/>
<xsd:complexType name="tCorrelationPropertyBinding">
      <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:sequence>
                    <xsd:element name="dataPath" type="tFormalExpression" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute name="correlationPropertyRef" type="xsd:QName" use="required"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.39 – XML-схема для элемента Correlation Property Retrieval Expression

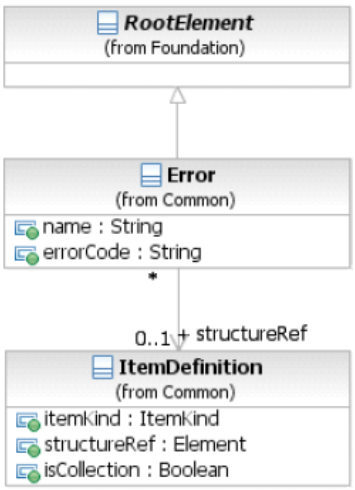
```
<xsd:element name="correlationPropertyRetrievalExpression" type="tCorrelationPropertyRetrievalExpression"/>
<xsd:complexType name="tCorrelationPropertyRetrievalExpression">
            <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:sequence>
                    <xsd:element name="messagePath" type="tFormalExpression" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute name="messageRef" type="xsd:QName" use="required"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.40 – XML-схема для элемента Correlation Subscription

```
<xsd:element name="correlationSubscription" type="tCorrelationSubscription"/>
<xsd:complexType name=" tCorrelationSubscription ">
      <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:sequence>
                    <xsd:element name="process" type="xsd:QName" use="required"/>
                    <xsd:element ref="correlationKeyRef" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="correlationPropertyBinding" type="xsd:QName" minOccurs="0" maxOc-
                            curs="unbounded"/>
                </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

8.3.3. Ошибка (Error)

`Ошибка (Error)` представляет собой содержимое **События** типа **Ошибка** (**Error Event**) либо `Ошибку (Fault)` при сбое `Операции (Operation)`. Посредством класса `ItemDefinition` указывается структура `Ошибки (Error)`. `Ошибка` генерируется тогда, когда в ходе обработки **Действия** (**Activity**) возникает критическая проблема, либо когда выполнение `Операции` завершается ошибкой.



Фигура 8.18 – Диаграмма классов элемента Error

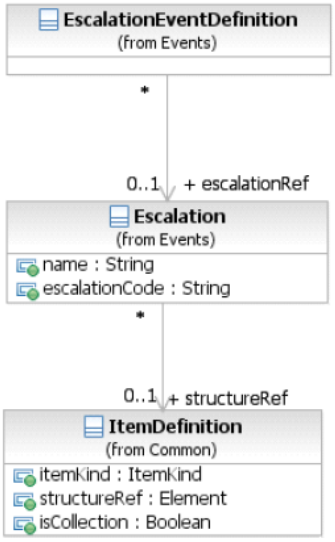
Элемент `Error` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством его связи с элементом `RootElement`. Таблица 8.41 содержит информацию о дополнительных атрибутах и ассоциациях элемента `Error`.

Таблица 8.41 – Атрибуты и ассоциации элемента Error

Название атрибута	Описание/использование
structureRef : ItemDefinition [0..1]	Используется для определения полезной нагрузки `Ошибки`.
name : string	Описательное имя `Ошибки`.
errorCode: string	Для **Конечных Событий** (**End Event**): Если *результатом* является `Ошибка`, то ДОЛЖЕН предоставляться `errorCode` (в данном случае значение атрибута `processType Процесса` равно «`executable`»). Инициирует `Ошибку`. Для **Промежуточных Событий** (**Intermediate Event**) в составе *стандартного потока операций*: Если *триггер* является `Ошибкой`, то ДОЛЖЕН быть введен `errorCode` (в данном случае значение атрибута `processType` **Процесса** равно «`executable`»). Инициирует `Ошибку`. Для **Промежуточных Событий** (**Intermediate Event**), присоединенных к границе **Действия** (**Activity**): Если триггер является `Ошибкой`, то МОЖЕТ БЫТЬ введен `errorCode`. **Событие** данного типа обрабатывает (catch) `Ошибку`. В случае если значение `errorCode` не введено, то **Событие** ДОЛЖНО инициироваться любой ошибкой. В случае если значение `errorCode` все же введено, то **Событие** ДОЛЖНО инициироваться лишь соответствующей `Ошибкой`, указанной в `errorCode`.

8.3.4. Эскалация (Escalation)

Эскалация (Escalation) возникает в момент, когда **Процесс** каким-либо образом должен отреагировать на происходящее. Класс ItemDefinition используется для определения структуры Эскалации.



Фигура 8.19 – Диаграмма классов элемента Escalation

Элемент Escalation наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5) посредством его связи с элементом RootElement. Таблица 8.42 содержит информацию о дополнительных ассоциациях элемента Escalation.

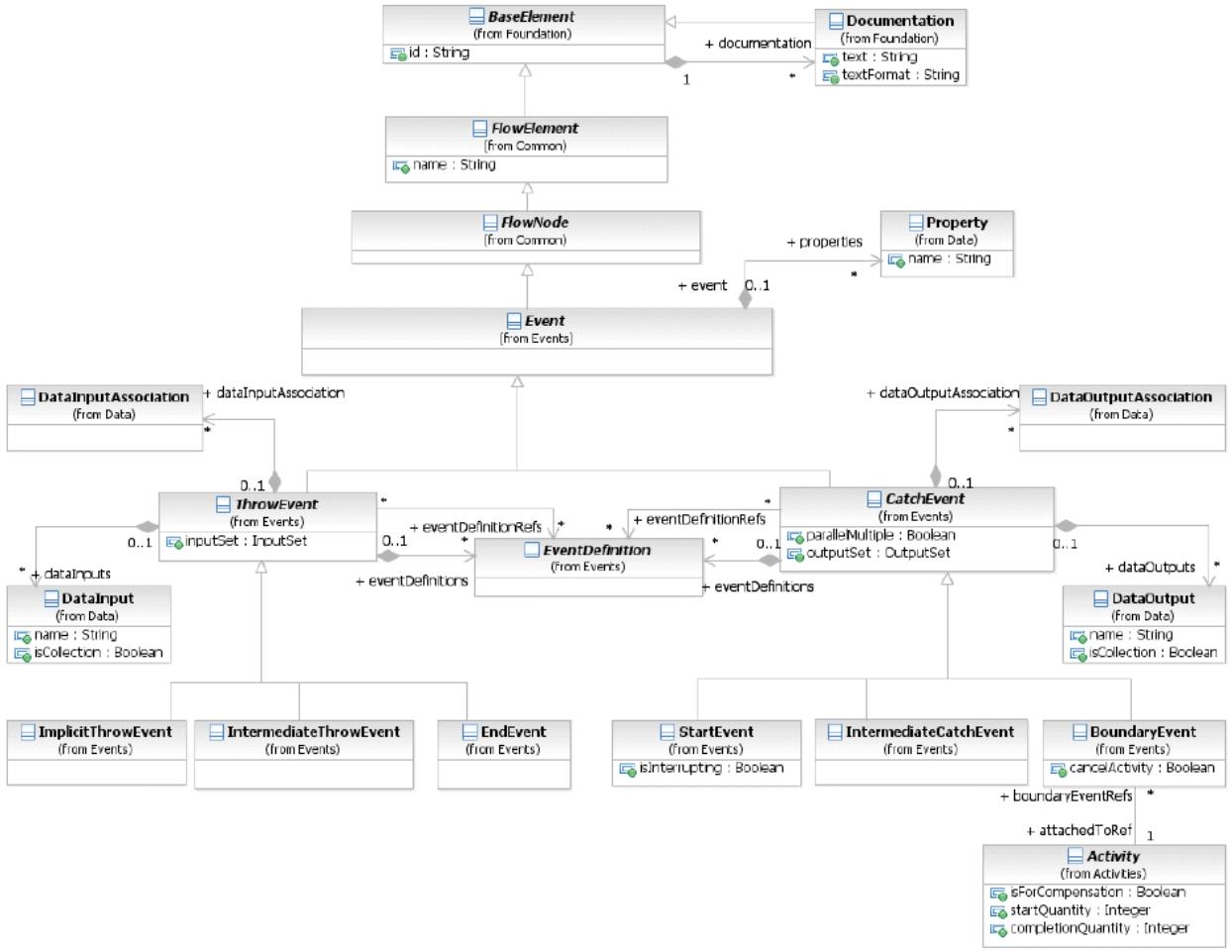
Таблица 8.42 – Атрибуты и ассоциации элемента Escalation

Название атрибута	Описание/использование
structureRef : ItemDefinition [0..1]	Используется для определения полезной нагрузки Эскалации.
name : string	Описательное имя Эскалации.
escalationCode: string	Для **Конечных Событий** (**End Event**): Если *результатом* является Эскалация, то ДОЛЖЕН предоставляться escalationCode (в данном случае значение атрибута processType **Процесса** равно «executable»). Инициирует Эскалацию. Для **Промежуточных Событий** (**Intermediate Event**) в составе *стандартного потока операций*: Если *триггер* является Эскалацией, то ДОЛЖЕН быть введен escalationCode (в данном случае значение атрибута processType **Процесса** равно «executable»). Инициирует Эскалацию. Для **Промежуточных Событий** (**Intermediate Event**), присоединенных к границе **Действия** (**Activity**): Если *триггер* является Эскалацией, то МОЖЕТ БЫТЬ введен errorCode. **Событие** данного типа обрабатывает (catch) Эскалацию. В случае, если значение escalationCode не введено, то **Событие** ДОЛЖНО инициироваться любой Эскалацией. В случае, если значение

	`escalationCode` все же введено, то **Событие** ДОЛЖНО инициироваться лишь соответствующей `Эскалацией`, указанной в `escalationCode`.

8.3.5. События (Events)

Событие (**Event**) – это то, что происходит в течение **Процесса**. **Событие** оказывает влияние на ход **Процесса** и чаще всего имеет причину (cause) или оказывает воздействие (impact). Сам термин «событие» является достаточно объемным и подразумевает многие явления, происходящие в **Процессе**. Начало выполнения **Действия**, его завершение, смена статуса документа, прибывшее **Сообщение** и т.д., - все это может подходить под категорию «события». Однако в **BPMN** существуют ограничения использования **Событий**. Таким образом, **Событиями BPMN** могут называться лишь те события, которые оказывают влияние на последовательность **Действий Процесса** или время их выполнения.



Фигура 8.20 – Диаграмма классов элемента Event

Элемент **Event** наследует атрибуты и ассоциации элемента `FlowElement` (см. таблицу 8.44), однако, не может иметь каких-либо дополнительных атрибутов и ассоциаций.

Информацию о типах **Событий** (**Стартовое**, **Промежуточное**, **Конечное**) см. в подразделе 10.4.5.

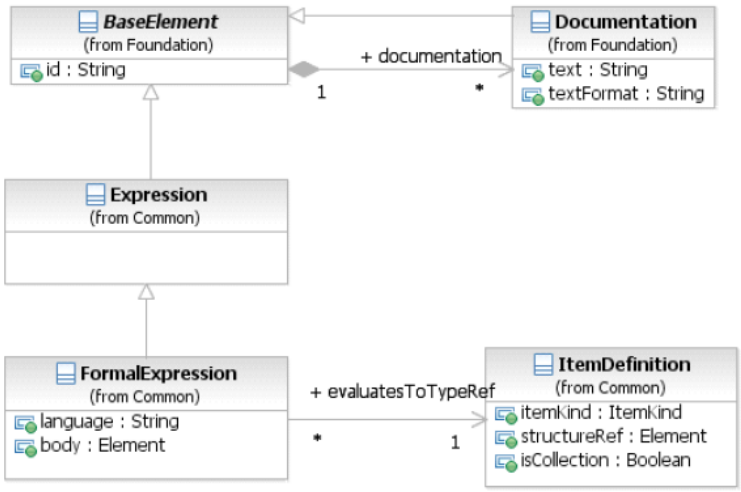
8.3.6. Выражения (Expressions)

Класс Expression используется для определения Выражения (Expression) в виде текста на естественном языке. Такие Выражения являются неисполняемыми. Текст на естественном языке вводится с использованием атрибута documentation, заимствованного из элемента BaseElement.

Элемент Expression наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5), однако, не может иметь каких-либо дополнительных атрибутов и ассоциаций.

Выражения используются довольно во многих местах модели **BPMN**. С их помощью добывается информация из различных элементов (как правило, из элементов, содержащих данные). Наиболее распространенный пример использования Выражений – это моделирование решений, когда условные Выражения (conditional Expressions) используются для определения направления маршрутов в соответствии с определенными критериями.

BPMN поддерживает использование недостаточно определенных Выражений, в которых логика представлена в виде описательного текста на естественном языке разработчика модели. Язык также поддерживает формальные Выражения (formal Expressions), в которых логика является исполняемой и описана с использованием конкретного языка Выражений.



Фигура 8.21 – Диаграмма классов элемента Expression

Класс Expression

Класс Expression (выражений) используется для определения Выражения (Expression) в виде текста на естественном языке. Такие Выражения являются неисполняемыми и считаются недостаточно определенными.

Выражение (Expression) может использоваться двумя способами: оно может содержаться там же, где и используется, либо оно может быть указано на одном из уровней **Процесса**, куда к нему ведет ссылка из места, где оно должно быть использовано.

Элемент Expression наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5), однако, не может иметь каких-либо дополнительных атрибутов и ассоциаций.

Класс Formal Expression

Класс FormalExpression (формальных выражений) используется для определения исполняемых Выражений (Expression) с использованием конкретного языка Выражений. В качестве дополнения к

основному требованию определения таких Выражений также может быть использовано описание на естественном языке.

Язык Выражений, использующийся по умолчанию для определения всех Выражений, указывается в элементе Definitions посредством атрибута expressionLanguage, хотя для определения какого-либо Формального Выражения, использующего этот атрибут, указанный по умолчанию язык может быть изменен.

Элемент FormalExpression наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5) посредством его связи с элементом Expression. Таблица 8.43 содержит информацию о дополнительных атрибутах и ассоциациях элемента FormalExpression.

Таблица 8.43 – Атрибуты и ассоциации элемента FormalExpression

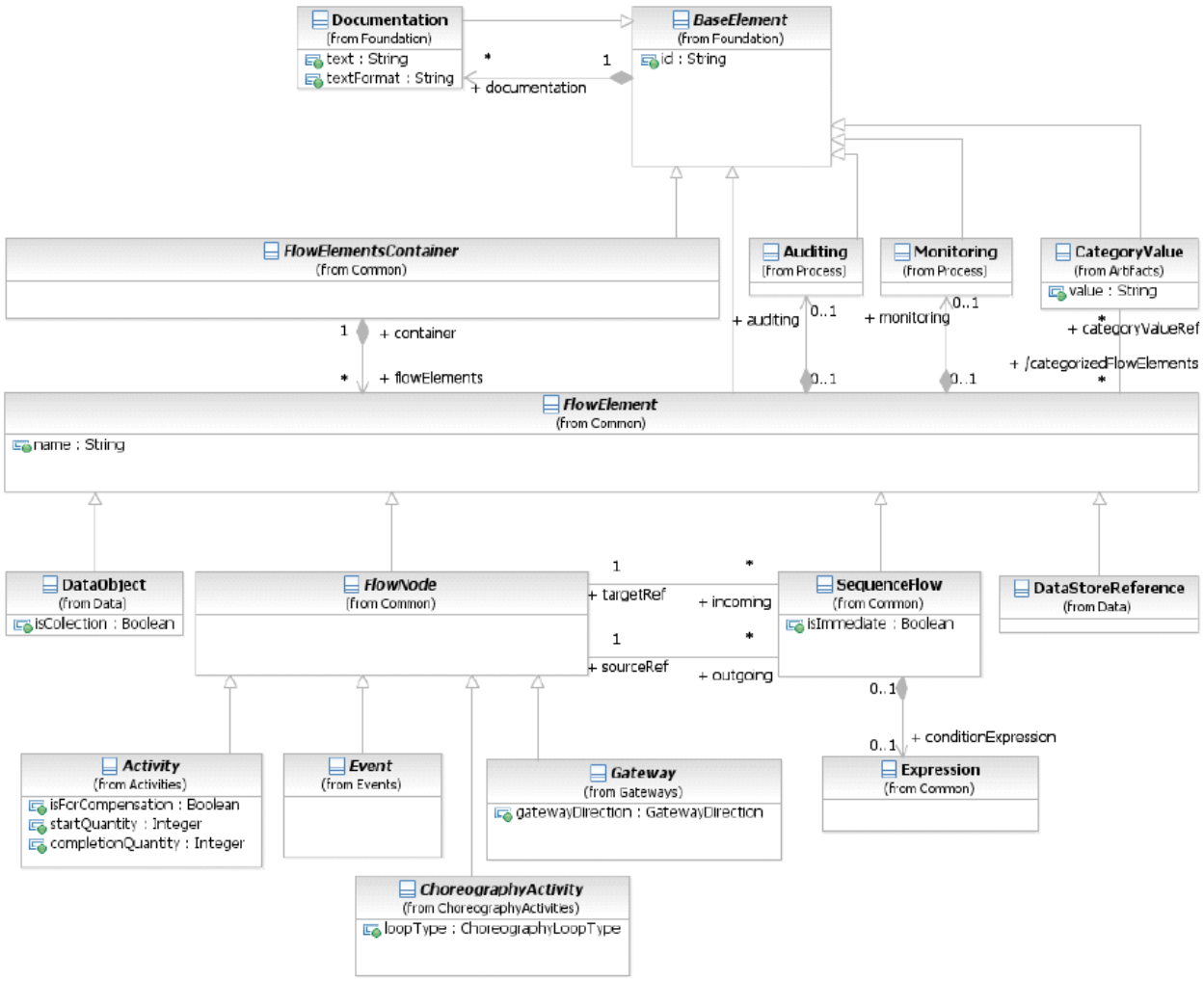
Название атрибута	Описание/использование
language: string [0..1]	Отменяет язык Выражений (Expression language), указанный в Definitions. **Язык ДОЛЖЕН иметь формат UML.**

body: Element	Тело Выражения. Обратите внимание, что данный атрибут не является значимым в случае, если для замены использован язык XML Schema. Напротив, сложным типом FormalExpression поддерживается смешанное наполнение. Тело Выражения определяется как наполнение элемента. Пример:
	`<formalExpression id="ID_2">`
	`  count(../dataObject[id="CustomerRecord_1"]/emailAddress) > 0`
	`  <evaluatesToType id="ID_3" typeRef="xsd:boolean"/>`
	`</formalExpression>`

evaluatesToTypeRef: ItemDefinition	Тип объекта, возвращаемый Выражением при вычислении. К примеру, *условные* Выражения (*conditional* Expressions) вычисляются как *булевские* (*boolean*).

8.3.7. Элемент Потока (Flow Element)

FlowElement представляет собой абстрактный суперкласс для всех элементов, входящих в состав потоков **Процесса** - так называемых Узлов Потока (FlowNodes, см. соотв. пункт раздела 8.3.13). Узлы Потока состоят из **Действий** (**Activities**, см. раздел 10.2), **Действий Хореографии** (**Choreography Activities**, см. раздел 11.4), **Шлюзов** (**Gateways**, см. раздел 10.5), **Событий** (**Events**, см. раздел 10.4), **Объектов Данных** (**Data Objets**, см. раздел 10.3.1), **Ассоциаций Данных** (**Data Association**, см. соотв. пункт раздела 10.3.2), **Потоков Операций** (**Sequence Flow**, см. раздел 8.3.13).



Фигура 8.22 – Диаграмма класса элемента FlowElement

Элемент `FlowElement` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.44 содержит информацию о дополнительных атрибутах и ассоциациях элемента `FlowElement`.

Таблица 8.44 – Атрибуты и ассоциации элемента FlowElement

Название атрибута	Описание/использование
name: string [0..1]	Описательное имя элемента.
categoryValueRef: Category- Value [0..*]	Ссылка на `Значения Категории (Category Values)`, ассоциированные с данным `Элементом Потока (Flow Element)`.
auditing: Auditing [0..1]	Метод указания свойств `проверки`. Данный атрибут может использоваться только для **Процесса**.
monitoring: Monitoring [0..1]	Метод указания свойств `мониторинга`. Данный атрибут может использоваться только для **Процесса**.

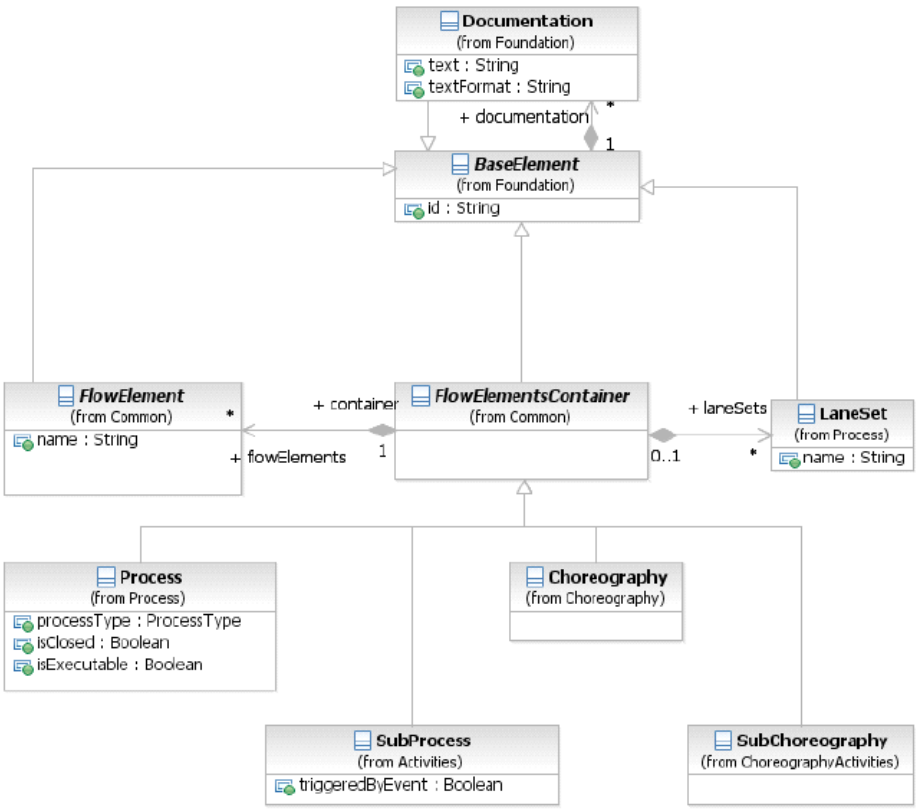
8.3.8. Контейнер Элементов Потока (Flow Elements Container)

Элемент `FlowElementsContainer` представляет собой абстрактный суперкласс для диаграмм **BPMN** (видов) и служит для определения расширенного набора элементов, отображаемых на диаграммах. В основном, `Контейнер Элементов Потока` содержит такие `Элементы Потока`, как **События** (**Events**, см. раздел

10.4), **Шлюзы** (**Gateways**, см. раздел 10.5), **Потоки Операций** (**Sequence Flow**, см. раздел 8.3.13), **Действия** (**Activities**, см. раздел 10.2) и **Действия Хореографии** (**Choreography Activities**, см. раздел 11.4).

Действий (**Activities**, см. раздел 10.2), **Действий Хореографии** (**Choreography Activities**, см. раздел 11.4), **Шлюзов** (**Gateways**, см. раздел 10.5), **Событий** (**Events**, см. раздел 10.4), **Объектов Данных** (**Data Objets**, см. раздел 10.3.1), **Ассоциаций Данных** (**Data Association**, см. соотв. пункт раздела 10.3.2), **Потоков Операций** (**Sequence Flow**, см. раздел 8.3.13).

BPMN выделяет четыре (4) типа `Контейнеров Элементов Потока` (см. фигуру 8.23): **Процесс** (**Process**), **Подпроцесс** (**Sub-Process**), **Хореография** (**Choreography**) и **Подхореография** (**Sub-Choreography**).



Фигура 8.23 – Диаграмма классов элемента FlowElementContainers

Элемент `FlowElementsContainer` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.45 содержит информацию о дополнительных ассоциациях элемента `FlowElementsContainer`.

Таблица 8.45 – Ассоциации элемента FlowElementsContainer

Название атрибута	Описание/использование
flowElements: Flow Element [0..*]	Данная ассоциация используется для указания конкретных ***элементов потока*** (***flow elements***), содержащихся в `Контейнере Элементов Потока`. *Элементами Потока* являются: **События, Шлюзы**, **Потоки Операций**, **Действия**, **Объекты Данных**, **Ассоциации Данных** и **Действия Хореографии**. Обратите внимание: • **Действия Хореографии** НЕ ДОЛЖНЫ являться `Элементами Потока` **Процесса**.

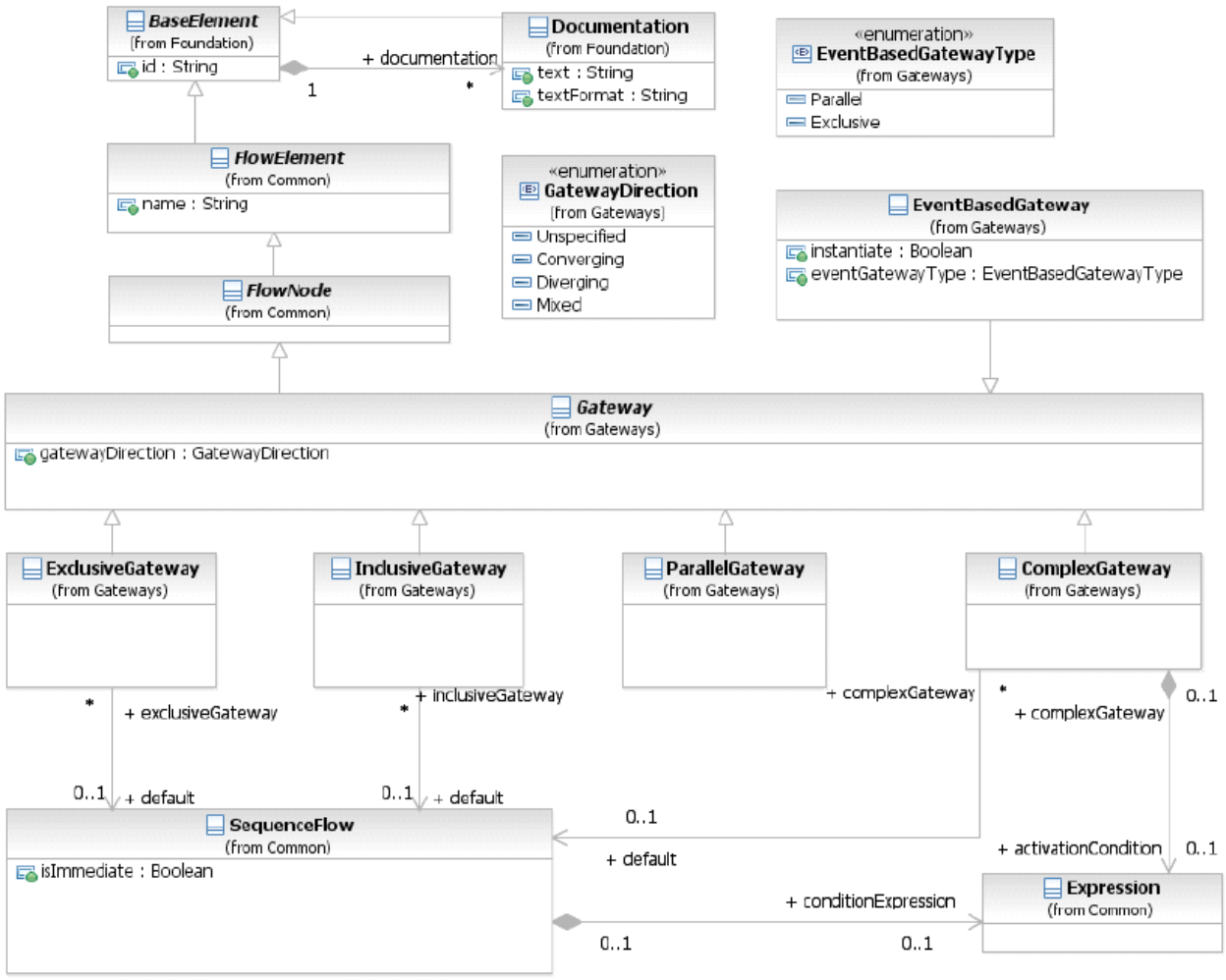
	• **Действия**, **Ассоциации Данных**, а также **Объекты Данных** НЕ ДОЛЖНЫ являться `Элементами Потока` **Хореографии**.
laneSets: LaneSet [0..*]	Данный атрибут служит для определения списка элементов `LaneSets`, используемых в `Контейнере Элементов Потока`. `LaneSets` не используются в **Хореографиях** и **Подхореографиях**.

8.3.9. Шлюзы (Gateways)

Шлюзы используются для контроля расхождений и схождений **Потоков операций** (маршрутов *Токенов*) в **Процессе**. В случае, если необходимости в контроле потока нет, то отпадает и необходимость использования **Шлюза**. Под термином «шлюз» подразумевается пропускное устройство, которое либо пропускает поток через себя, либо не пропускает. Это означает, что при запуске **Шлюза** *токены*, прибывающие к нему либо сливаются на входе, либо разделяются на выходе.

Как и **Действия** (**Activities**), **Шлюзы** могут получать и генерировать дополнительные *токены* контроля, эффективно осуществляя контроль исполнительной семантики текущего **Процесса**. Основное отличие состоит в том, что **Шлюзы** не служат для отображения осуществляемой «деятельности», а также не оказывают влияния на критерии выполнения **Процесса** (стоимость, время и т.д.).

Шлюзы используются для контроля как сходящихся, так и расходящихся **Потоков Операций** (**Sequence Flows**). Это означает, что один **Шлюз** может иметь множество потоков, как на входе, так и на выходе. Разработчики моделей и инструменты моделирования могут захотеть применить лучшие наработки по использованию **Шлюзов**, используя лишь одну из предложенных функций. Именно поэтому для объединения и последующего разделения **Потоков Операций** необходимо использовать два последовательных **Шлюза**.



Фигура 8.24 – Диаграмма классов элемента Gateway

Для получения более подробной информации о типах **Шлюзов** (**Эксклюзивном**, **Неэксклюзивном, Параллельном**, **Основанном на Событиях**, **Комплексном**), используемых в **Процессах**, см. раздел 10.5; информацию о типах **Шлюзов**, используемых в **Хореографиях** (**Choreographies**), см. в разделе 11.6.

Класс **Gateway** представляет собой абстрактный тип. Каждый из его конкретных подклассов определяет свою семантику для каждого типа **Шлюзов**, в соответствии с которой **Шлюз** ведет себя в той или иной ситуации.

Элемент **Gateway** наследует атрибуты и ассоциации элемента `FlowElement` (см. таблицу 8.44). Таблица 8.46 содержит информацию о дополнительных атрибутах элемента **Gateway**.

Таблица 8.46 – Атрибуты элемента Gateway

Название атрибута	Описание/использование
gatewayDirection: GatewayDirection = Unspecified { Unspecified | Converging | Diverging | Mixed }	Посредством значений данного атрибута указываются ограничения на то, каким образом МОГУТ использоваться **Шлюзы**. • **Unspecified**: В данном случае ограничения отсутствуют. **Шлюз** МОЖЕТ иметь любое количество *входящих* и *исходящих* **Потоков Операций**. • **Converging**: **Шлюз** МОЖЕТ иметь любое

	количество *входящих* **Потоков Операций**, однако, количество *исходящих* от данного **Шлюза Потоков Операций** ДОЛЖНО БЫТЬ не более одного (1). • **Diverging**: **Шлюз** МОЖЕТ иметь любое количество *исходящих* **Потоков Операций**, однако, количество *входящих* **Потоков Операций** ДОЛЖНО БЫТЬ не более одного (1). • **Mixed**: **Шлюз** соединен с множеством *исходящих* и *входящих* **Потоков Операций**.

8.3.10. Определение компонента (Item Definition)

Такие элементы **BPMN**, как `Объекты Данных (DataObjects)` и `Сообщения (Messages)`, представляют собой компоненты, на которые в ходе выполнения **Процесса** можно оказывать воздействие и которые можно передавать, изменять и хранить. Такие компоненты могут быть физическими (к примеру, деталями транспортного средства) и информационными (например, каталогом деталей транспортного средства).

Важнейшей характеристикой любого компонента **Процесса** является его структура. **BPMN** не требует использования какого-то определенного формата структуры данных, однако, нотация выделяет язык XML Schema, так как он определен по умолчанию. Атрибут `structure` ссылается на текущую структуру данных.

Для всех элементов формат структуры данных по умолчанию определяется в элементе `Definitions` с указанием значения его атрибута `typeLanguage`. К примеру, значение атрибута `typeLanguage`, равное «*http://www.w3.org/2001/XMLSchema*», указывает на то, что структура данных, используемая для элементов `Definitions`, должна иметь тип XML Schema. Если же значение не указано, то по умолчанию определяется тип XML Schema. Элемент `Импорт (Import)` используется для дальнейшего указания расположения структуры данных (если она пригодна к использованию). Например, если форматом структуры данных является XML schema, то для указания расположения файла этой схемы используется `Import`.

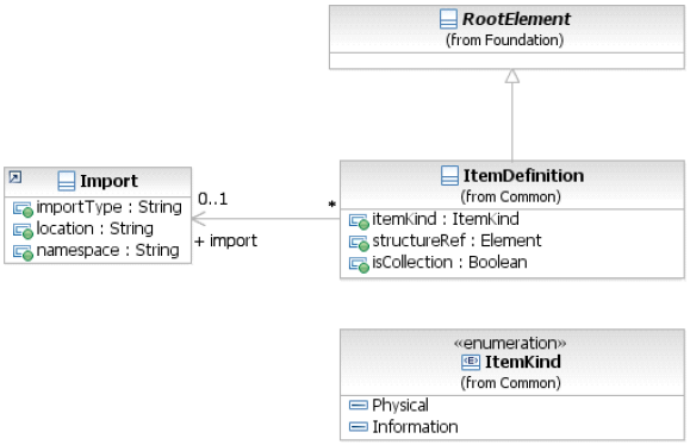
Определения структуры всегда указываются в качестве отдельных сущностей, поэтому они не могут быть встроены при их использовании. Далее будет понятно, почему любое упоминание об определении структуры является ссылкой на элемент. Вот почему этот класс наследует атрибуты элемента `RootElement`.

Посредством элемента `ItemDefinition` указывается ссылка для импорта, ведущая к месту расположения необходимого определения структуры.

В случаях, когда структура данных представляет собой коллекцию, это разнообразие может быть указано посредством атрибута `isCollection`. Если значение этого атрибута равно «*true*», но текущий тип структуры данных коллекцией не является, такая модель считается неверной. Благодаря гибкости инструментов **BPMN**, имеется возможность выполнения элементарной проверки на наличие таких противоречий и создания отчетов об ошибках. Значением по умолчанию, определенным для данного элемента, является «*false*».

Атрибут `itemKind` определяет, является ли компонент физическим или информационным.

На фигуре 8.25 отображена диаграмма классов элемента `ItemDefinition`. Если значение элемента `ItemDefinition` все же указано, оно содержится в `Definitions`.



Фигура 8.25 – Диаграмма классов элемента ItemDefinition

Элемент `ItemDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством его связи с элементом `RootElement`. Таблица 8.47 содержит информацию о дополнительных атрибутах и ассоциациях элемента `ItemDefinition`.

Таблица 8.47 – Атрибуты и ассоциации элемента ItemDefinition

Название атрибута	Описание/использование
itemKind: ItemKind = Information { Information | Physical }	Определяет, является ли компонент физическим или информационным. Возможны значения «`physical`» и «`information`». Значением по умолчанию является «`information`».
structureRef: [Element [0..1]	Конкретная структура данных, которая будет использована.
import: Import [0..1]	Определяет расположение структуры данных и её формат. В случае если значение атрибута `importType` не указано, берется значение `typeLanguage`, определенное для элемента `Definitions`, содержащего данный элемент `ItemDefinition`.
isCollection: boolean = False	Значение «*true*» данного атрибута указывает на то, что текущий тип структуры данных – коллекция.

8.3.11. Сообщение (Message)

Сообщение (**Message**) представляет собой содержимое диалога между двумя *Участниками* (*Participants*). Согласно **BPMN 2.0**, **Сообщение** выступает в роли графического декоратора (в **BPMN 1.2** данный элемент относился к элементам поддержки). Для определения структуры **Сообщения** используется элемент `ItemDefinition`.

Графически **Сообщение** отображается следующим образом:

- Графический элемент **Сообщение** представляет собой прямоугольник, линии в котором расположены таким образом, чтобы получилось изображение конверта (см. фигуру 8.26). Данный графический элемент ДОЛЖЕН БЫТЬ выполнен одинарной тонкой линией.
    - Текст, цвет, размер, а также линии, используемые для изображения **Сообщения**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».

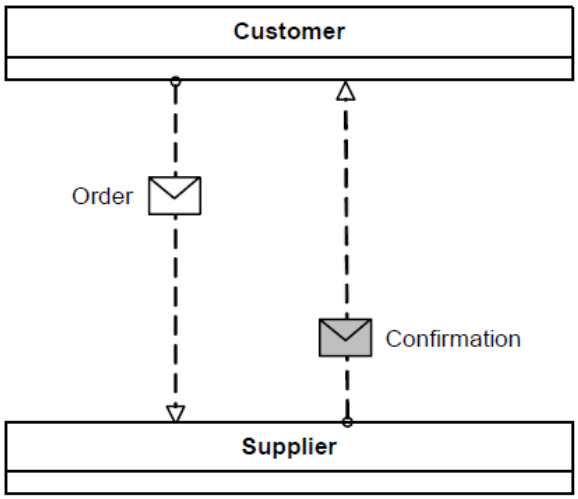
Фигура 8.26 - Сообщение

Для одной **Задачи Хореографии** (**Choreography Task**) на Диаграмме **Хореографии** (**Choreography**) МОЖЕТ использоваться более одного **Сообщения**. В данном случае необходимо знать, какое из **Сообщений** является первым, т.е. инициирует взаимодействие. Для удобства графическое изображение **Сообщения**, не являющегося инициирующим, имеет светлую заливку (см. фигуру 8.27).

Фигура 8.27 – Неинициирующее Сообщение

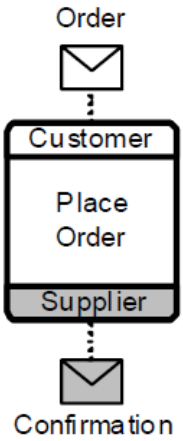
- Любое **Сообщение**, отправляемое неинициирующим *Участником* (*Participant*) или **Подхореографией** (**Sub-Choreography**), ДОЛЖНО иметь светлую заливку.

Во **Взаимодействии** (**Collaboration**) сам диалог представлен **Потоком Сообщений** (**Message Flow**, для получения более подробной информации см. раздел **Поток Сообщений**, расположенный ниже). По желанию, **Сообщение** может изображаться в качестве графического декоратора **Потока Сообщений Взаимодействия** (см. фигуры 8.28 и 8.29).



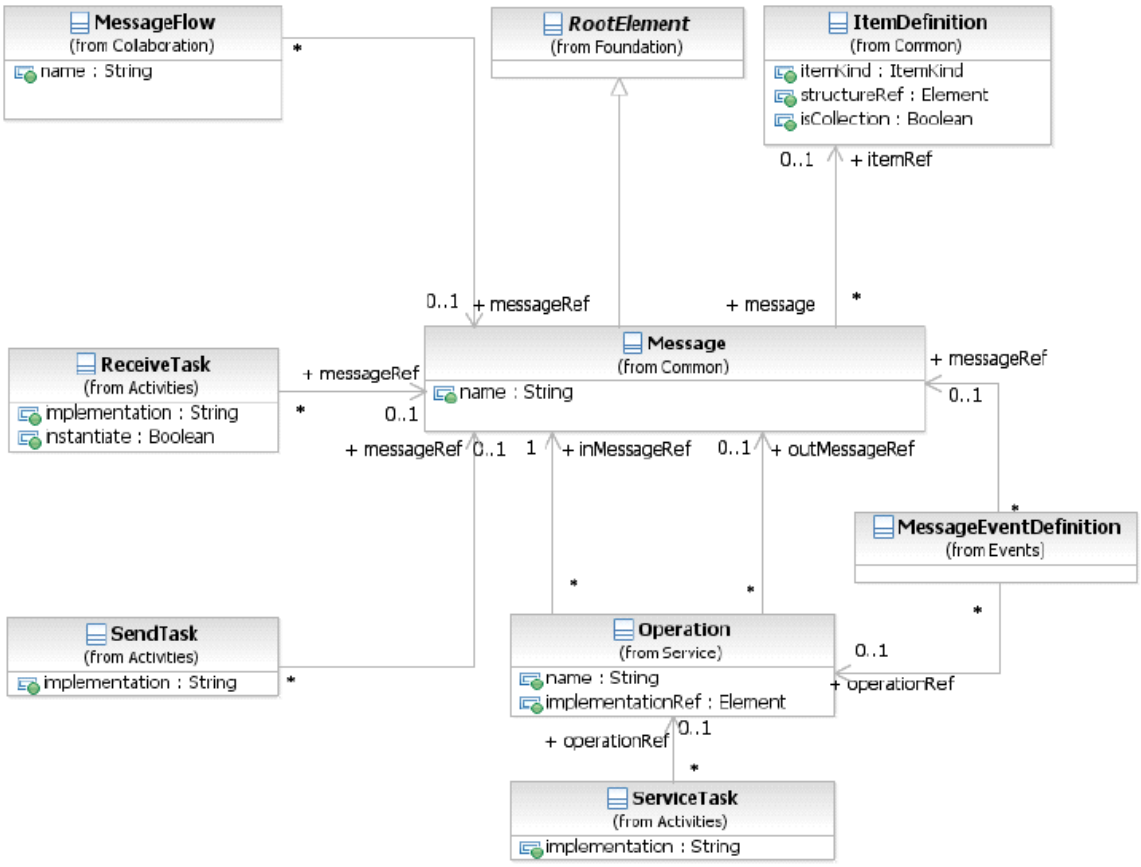
Фигура 8.28 – Ассоциация Сообщений, перекрывающая Потоки Сообщений

В **Хореографии** (**Choreography**) диалог представлен **Задачей Хореографии** (**Choreography Task**, для получения более подробной информации см. раздел 11.4.1). **Сообщение** в ней может изображаться в качестве декоратора, ассоциированного с **Задачей Хореографии** (см. фигуру 8.29).



Фигура 8.29 – Сообщения, графически ассоциированные с Задачей Хореографии

На фигуре 8.30 представлена диаграмма классов, содержащая атрибуты и ассоциации элемента **Message**.



Фигура 8.30 – Диаграмма классов элемента Message

Элемент **Message** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством его связи с элементом `RootElement`. Таблица 8.48 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Message**.

Таблица 8.48 – Атрибуты и ассоциации элемента Message

Название атрибута	Описание/использование

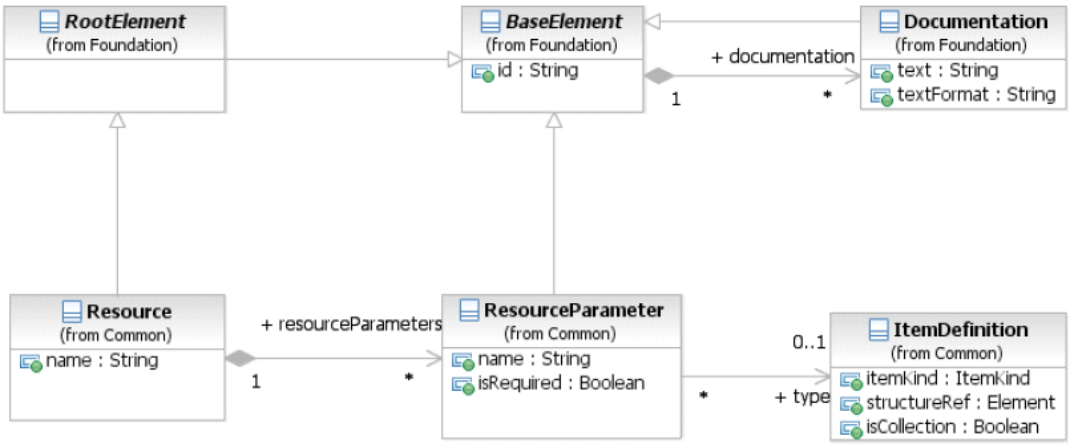
name: string	Название представляет собой текстовое описание **Сообщения**.
itemRef : ItemDefinition [0..1]	Данный атрибут используется для определения полезной нагрузки **Сообщения**.

8.3.12. Ресурсы (Resources)

Класс `Resource` используется для определения ресурсов, на которые ссылаются **Действия** (**Activities**). `Ресурсы` могут представлять собой как людские (`Human Resources`), так и любые другие типы `Ресурсов`, назначаемых на **Действия** в ходе выполнения **Процесса**.

Понятие `Ресурса` – абстрактное, т.к. оно лишь определяет `Ресурс`, однако, не объясняет, к примеру, того, какие ассоциации для текущих ID пользователей создаются в ходе выполнения **Процесса**. Для множественных **Действий** (**Activities**) могут назначаться одни и те же `Ресурсы`.

Для каждого `Ресурса` может быть определен набор `Параметров Ресурса (ResourceParameters)`. Эти параметры используются для указания запроса, к примеру, в Структуру Организации. Каждое **Действие**, ссылающееся на параметризированный `Ресурс`, может связывать значения, доступные в рамках этого **Действия**, с данными параметрами.



Фигура 8.31 – Диаграмма классов элемента Resource

Элемент `Resource` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством его связи с элементом `RootElement`. Таблица 8.49 содержит информацию о дополнительных ассоциациях элемента `Resource`.

Таблица 8.49 – Атрибуты и ассоциации элемента Resource

Название атрибута	Описание/использование
name: string	Посредством данного атрибута указывается название `Ресурса`.
resourceParameters: ResourceParameter [0..*]	Данная ассоциация определяет параметры, необходимые в ходе выполнения **Процесса** для выбора `Ресурса`.

Как было описано выше, элемент `Resource` может определять `параметры`, необходимые для указания запроса о выборе текущих `ресурсов` (например, пользовательские ID).

Элемент `ResourceParameter` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством его связи с элементом `RootElement`. Таблица 8.50 содержит информацию о дополнительных ассоциациях элемента `ResourceParameter`.

Таблица 8.50 – Атрибуты и ассоциации элемента ResourceParameter

Название атрибута	Описание/использование
name: string	Определяет название параметра для запроса.
type: ItemDefinition	Определяет тип параметра для запроса.
isRequired: boolean	Указывает на то, является ли данный параметр опциональным или обязательным.

8.3.13. Поток Операций (Sequence Flow)

Поток Операций (**Sequence Flow**) используется для отображения последовательности `Элементов Потока` (`Flow Elements`), входящих в состав **Процесса** или **Хореографии** (**Choreography**). Каждый **Поток Операций** имеет лишь один *источник* (*source*) и одну *цель* (*target*). И цель, и источник **Потока Операций** ДОЛЖНЫ относиться к следующим `Элементам Потока`: **События** (**Стартовое**, **Промежуточное**, **Конечное**), **Действия** (**Задачи** и **Подпроцессы**, - для **Процессов**), **Действия Хореографии** (**Задачи Хореографии** и **Подхореографии**, - для **Хореографий**), **Шлюзы**.

- **Поток Операций** отображается в виде линии со стрелкой. Линия ДОЛЖНА БЫТЬ одинарной и жирной (фигура 8.32).
    - Текст, цвет, размер, а также линии, используемые для изображения **Потока Операций**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».

Фигура 8.32 – Поток Операций

По желанию, для **Потока операций** может быть определено условное Выражение (condition Expression), указывающее на то, что *токен* пересечет **Поток Операций** только в случае, если значение данного Выражения будет равно «*true*». Условное Выражение используется, как правило, тогда, когда источником **Потока Операций** является **Шлюз** (**Gateway**) или **Действие** (**Activity**).

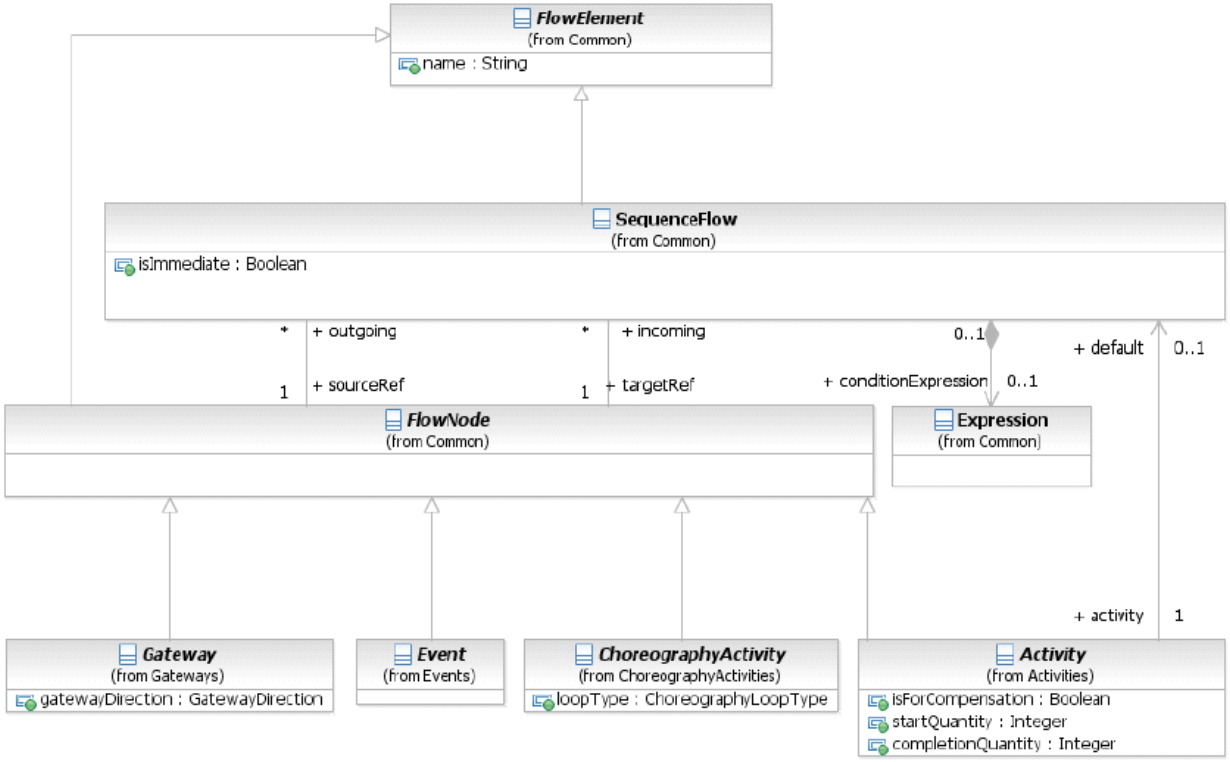
- *Условный исходящий* (*conditional outgoing*) **Поток Операций**, направленный от **Действия**, ДОЛЖЕН отображаться с маркером, выполненным в виде небольшого ромба и находящимся в начале графического элемента **Потока Операций** (см. фигуру 8.33).
    - В случае если *условный* (*conditional*) **Поток Операций** направлен от **Действия**, являющегося источником, то от этого **Действия** ДОЛЖЕН БЫТЬ направлен по-меньшей мере ещё один *исходящий* **Поток Операций**.
- *Условный исходящий* (*conditional outgoing*) **Поток Операций**, направленный от **Шлюза**, НЕ ДОЛЖЕН отображаться с маркером в виде ромба, находящимся в начале графического элемента **Потока Операций**.
    - **Шлюз**, являющийся источником, НЕ ДОЛЖЕН быть **Параллельным** (**Parallel**) или основанным на **Событиях** (**Event**).

Фигура 8.33 – Условный Поток Операций

Поток Операций, источником которого является **Эксклюзивный** (**Exclusive**), **Неэксклюзивный** (**Inclusive**) или **Комплексный** (**Complex**) **Шлюз** или **Действие**, может являться **Потоком Операций** *по умолчанию* (*default*). Такой **Поток Операций** изображается с маркером, указывающим на его статус. **Поток Операций** *по умолчанию* выбирается (т.е. через него проходит токен) только в случае, если все остальные исходящие **Потоки Операций**, направленные от **Действия** или **Шлюза**, не являются верными (к примеру, условные `Выражения`, содержащиеся в них, имеют значения «*false*»).

- *Исходящий* **Поток Операций** *по умолчанию* (*default outgoing* **Sequence Flow**) ДОЛЖЕН изображаться с маркером, выполненным в виде символа «/» (наклонной черты) и находящимся в начале графического элемента **Потока Операций** (см. фигуру 8.34).

Фигура 8.34 – Поток Операций по умолчанию



Фигура 8.35 – Диаграмма классов элемента SequenceFlow

Элемент **SequenceFlow** наследует атрибуты и ассоциации элемента `FlowElement` (см. таблицу 8.44). Таблица 8.51 содержит информацию о дополнительных атрибутах и ассоциациях элемента **SequenceFlow**.

Таблица 8.51 – Атрибуты и ассоциации элемента SequenceFlow

Название атрибута	Описание/использование
sourceRef: FlowNode	Посредством данного атрибута указывается `Узел Потока (FlowNode)`, от которого направлен **Поток Операций**. Для **Процессов**: *Источниками* могут являться лишь **Действия**, **Шлюзы** и **События**. Однако **Действия**, представляющие собой **Событийные Подпроцессы**, не могут быть *источниками*.

	Для **Хореографии** (**Choreography**): *Источниками* могут являться лишь **Действия**, **Шлюзы** и **События** Хореографии.
targetRef: FlowNode	Посредством данного атрибута указывается `Узел Потока (FlowNode)`, к которому направлен **Поток Операций**. Для **Процессов**: Целями могут являться лишь **Действия**, **Шлюзы** и **События**. Однако **Действия**, представляющие собой **Событийные Подпроцессы**, не могут быть целями. Для **Хореографии** (**Choreography**): *Источниками* могут являться лишь **Действия**, **Шлюзы** и **События** Хореографии.
conditionExpression: Expression [0..1]	Опциональное булевское `Выражение`, выступающее в роли пропускного условия. *Токен* будет помещен в данный **Поток Операций** лишь в случае, если значение для `условного Выражения` (`conditionExpression`) будет равно «*true*».
isImmediate: boolean [0..1]	Опциональное булевское значение, указывающее на то, может ли **Действие** или **Действие Хореографии**, не включенное в модель с **Потоком Операций**, появляться между элементами, которые данный **Поток Операций** соединяет. Значение «true» означает, что НЕ МОГУТ, а значение «false» означает, что МОГУТ. Необходимо также обратить внимание на атрибут isClosed Процесса, Хореографии и Взаимодействия (Collaboration). Если значение данного атрибута не определено, то выбор семантики по умолчанию зависит от типа модели, содержащей **Потоки Операций**: • При отсутствии значения для невыполняемых **Процессов** (публичных **Процессов** и невыполняемых приватных **Процессов**) и Хореографий будет выбрана та же семантика, как если бы значение было равно «*false*». • При отсутствии значения для выполняемых **Процессов** будет выбрана та же семантика, как если бы значение было равно «*true*». • Значение данного атрибута для выполняемых **Процессов** НЕ ДОЛЖНО быть равно «*false*».

Узел Потока (Flow Node)

Элемент `FlowNode` используется для указания ассоциаций с каким-то элементом как с источником и целью **Потока Операций** (см. фигуру 8.35). Это необходимо для того, чтобы не указывать отдельные ассоциации элементов, способных соединяться с **Потоком Операций** (см. описание выше). С **Потоком Операций** могут соединяться лишь **Шлюзы**, **Действия**, **Действия Хореографии** и **События**, поэтому только эти элементы могут являться подклассами элемента `FlowNode`.

Известно, что **Шлюзы**, **Действия**, **Действия Хореографии** и **События** обладают собственными атрибутами, ассоциациями и правами наследования. Что касается элемента `FlowNode`, то он не может ничего

наследовать от других элементов **BPMN**. Таблица 8.52 содержит информацию о дополнительных ассоциациях элемента `FlowNode`.

Таблица 8.52 – Ассоциации элемента FlowNode

Название атрибута	Описание/использование
incoming: Sequence Flow [0..*]	Посредством данного атрибута указывается *входящий* **Поток Операций** `Узла Потока`.
outgoing: Sequence Flow [0..*]	Посредством данного атрибута указывается *исходящий* **Поток Операций** `Узла Потока`. Является упорядоченным множеством.

8.3.14. Представление XML-схем для Пакета Общий (Common Package)

Таблица 8.53 – XML-схема для элемента Error

```
<xsd:element name="error" type="tError" substitutionGroup="rootElement"/>
<xsd:complexType name="tError">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:attribute name="name" type="xsd:string"/>
            <xsd:attribute name="errorCode" type="xsd:string"/>
            <xsd:attribute name="structureRef" type="xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.54 – XML-схема для элемента Escalation

```
<xsd:element name="escalation" type="tEscalation" substitutionGroup="rootElement"/>
<xsd:complexType name="tEscalation">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:attribute name="name" type="xsd:string"/>
            <xsd:attribute name="escalationCode" type="xsd:string"/>
            <xsd:attribute name="structureRef" type="xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.55 – XML-схема для элемента Expression

```
<xsd:element name="expression" type="tExpression"/>
<xsd:complexType name="tExpression">
    <xsd:complexContent>
        <xsd:extension base="tBaseElementWithMixedContent"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.56 – XML-схема для элемента FlowElement

```
<xsd:element name="flowElement" type="tFlowElement"/>
<xsd:complexType name="tFlowElement" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="auditing" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="monitoring" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="categoryValueRef" type="xsd:QName" minOccurs="0" maxOc-
                        curs="unbounded"/>
            </xsd:sequence>
```

```
            <xsd:attribute name="name" type="xsd:string"/>
        </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.57 – XML-схема для элемента FlowNode

```
<xsd:element name="flowNode" type="tFlowNode"/>
<xsd:complexType name="tFlowNode" abstract="true">
        <xsd:complexContent>
            <xsd:extension base="tFlowElement">
                <xsd:sequence>
                    <xsd:element name="incoming" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element name="outgoing" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.58 – XML-схема для элемента FormalExpression

```
<xsd:element name="formalExpression" type="tFormalExpression" substitutionGroup="expression"/>
<xsd:complexType name="tFormalExpression">
        <xsd:complexContent>
            <xsd:extension base="tExpression">
                <xsd:attribute name="language" type="xsd:anyURI" use="optional"/>
                <xsd:attribute name="evaluatesToTypeRef" type="xsd:QName"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.59 – XML-схема для элемента InputOutputBinding

```
<xsd:element name="ioBinding" type="tinputOutputBinding"/>
<xsd:complexType name="tinputOutputBinding">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:attribute name="inputDataRef" type="xsd:IDREF"/>
                <xsd:attribute name="outputDataRef" type="xsd:IDREF"/>
                <xsd:attribute name="operationRef" type="xsd:QName"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.60 – XML-схема для элемента ItemDefinition

```
<xsd:element name="itemDefinition" type="tItemDefinition" substitutionGroup="rootElement"/>
<xsd:complexType name="tItemDefinition">
        <xsd:complexContent>
            <xsd:extension base="tRootElement">
                <xsd:attribute name="structureRef" type="xsd:QName"/>
                <xsd:attribute name="isCollection" type="xsd:boolean" default="false"/>
                <xsd:attribute name="itemKind" type="tItemKind" default="Information"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>


<xsd:simpleType name="tItemKind">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Information"/>
            <xsd:enumeration value="Physical"/>
        </xsd:restriction>
</xsd:simpleType>
```

Таблица 8.61 – XML-схема для элемента Message

```
<xsd:element name="message" type="tMessage" substitutionGroup="rootElement"/>
<xsd:complexType name="tMessage">
	<xsd:complexContent>
		<xsd:extension base="tRootElement">
			<xsd:attribute name="name" type="xsd:string"/>
			<xsd:attribute name="itemRef" type="xsd:QName"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.62 – XML-схема для элемента Resources

```
<xsd:element name="resource" type="tResource" substitutionGroup="rootElement"/>
<xsd:complexType name="tResource">
	<xsd:complexContent>
		<xsd:extension base="tRootElement">
			<xsd:sequence>
				<xsd:element ref="resourceParameter" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="name" type="xsd:string" use="required"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.63 – XML-схема для элемента ResourceParameter

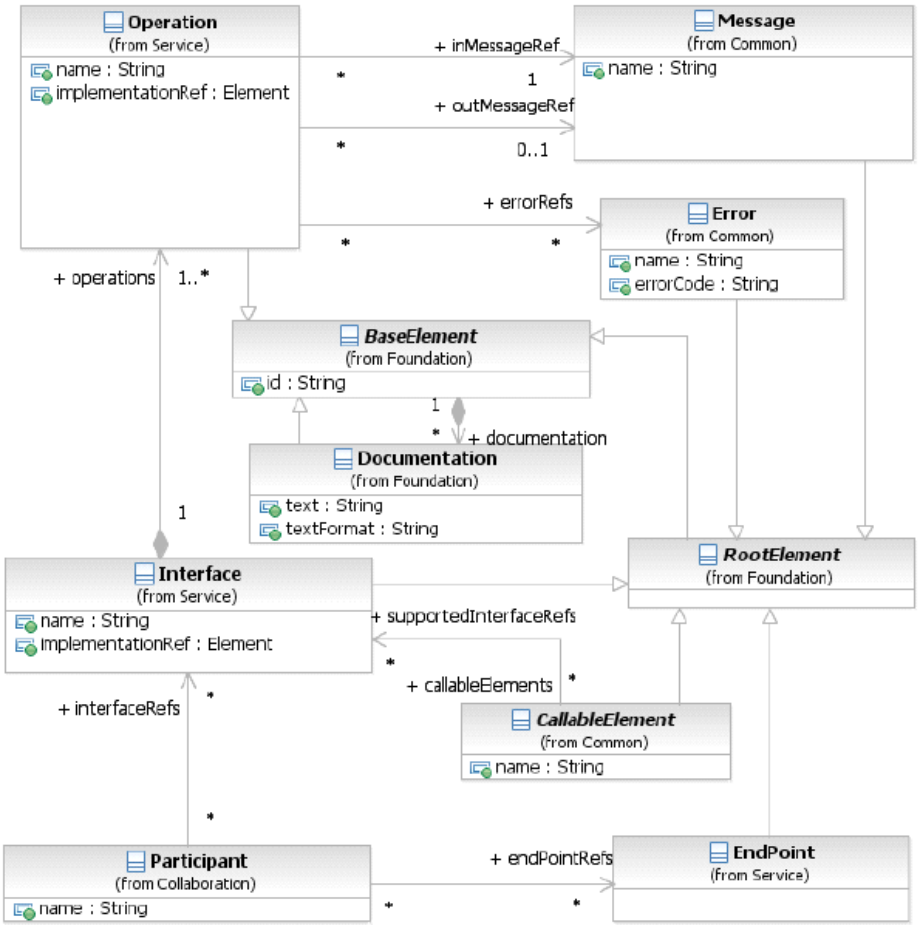
```
<xsd:element name="resourceParameter" type="tResourceParameter" />
<xsd:complexType name="tResourceParameter">
	<xsd:complexContent>
		<xsd:extension base="tBaseElement">
			<xsd:attribute name="name" type="xsd:string"/>
			<xsd:attribute name="type" type="xsd:QName"/>
			<xsd:attribute name="isRequired" type="xsd:Boolean" />
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

Таблица 8.64 – XML-схема для элемента SequenceFlow

```
<xsd:element name="sequenceFlow" type="tSequenceFlow" substitutionGroup="flowElement"/>
<xsd:complexType name="tSequenceFlow">
	<xsd:complexContent>
		<xsd:extension base="tFlowElement">
			<xsd:sequence>
				<xsd:element name="conditionExpression" type="tExpression" minOccurs="0" maxOccurs="1"/>
			</xsd:sequence>
			<xsd:attribute name="sourceRef" type="xsd:IDREF" use="required"/>
			<xsd:attribute name="targetRef" type="xsd:IDREF" use="required"/>
			<xsd:attribute name="isImmediate" type="xsd:boolean" use="optional"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

8.4. Пакет Сервис (Services)

Пакет Сервис (Service package) содержит конструкции, необходимые для моделирования сервисов, интерфейсов и видов деятельности.



Фигура 8.36 – Диаграмма классов пакета Service

8.4.1. Интерфейс (Interface)

Посредством элемента `Interface` указывается набор операций, выполняемых Сервисами (Services).

Элемент `Interface` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством его связи с элементом `RootElement`. Таблица 8.65 содержит информацию о дополнительных атрибутах и ассоциациях элемента `Interface`.

Таблица 8.65 – Атрибуты и ассоциации элемента Interface

Название атрибута	Описание/использование
name: string	Описательное имя элемента.
operations: Operation [1..*]	Посредством данного атрибута указываются операции, являющиеся частью `Интерфейса` (должна быть указана по-меньшей мере одна `Операция`).
callableElements: CallableElement [0..*]	Посредством данного атрибута указываются `Вызываемые Элементы` (`CallableElements`), использующие данный `Интерфейс`.
implementationRef: Element [0..1]	Данный атрибут позволяет указывать ссылку на конкретный артефакт базовой технологии реализации, представляющий интерфейс

	(например, WSDL porttype).

8.4.2. Конечная Точка (EndPoint)

В данной спецификации не рассматривается текущее определение адреса сервиса. Элемент `EndPoint` является точкой расширения от элемента `RootElement`. Элемент `EndPoint` МОЖЕТ БЫТЬ расширен посредством определений ссылок на конечные точки, описанных в других спецификациях (например, WS-Addressing). Элементы `EndPoint` могут быть определены для *Участников* (*Participants*).

8.4.3. Операция (Operation)

`Операция (Operation)` определяет **Сообщения** (**Message**), которые удаляются и, по желанию, могут создаваться тогда, когда вызывается `Операция`. `Операция` также может определять ошибки (от нуля и более), возвращаемые при сбое операции. Элемент `Operation` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 8.66 содержит информацию о дополнительных атрибутах и ассоциациях элемента `Operation`.

Таблица 8.66 – Атрибуты и ассоциации элемента Operation

Название атрибута	Описание/использование
name: string	Описательное имя элемента.
inMessageRef: Message	Посредством данного атрибута указывается входное **Сообщение** `Операции. Операция` имеет лишь одно такое **Сообщение**.
outMessageRef: Message [0..1]	Посредством данного атрибута указывается выходное **Сообщение** `Операции. Операция` может иметь максимум одно такое **Сообщение**.
errorRef: Error [0..*]	Посредством данного атрибута указываются ошибки, возвращаемые `Операцией`. Для `Операции` МОЖЕТ БЫТЬ указано любое количество элементов `Ошибка (Error)` – от нуля и более.
implementationRef: Ele-ment [0..1]	Данный атрибут позволяет указывать ссылку на конкретный артефакт базовой технологии реализации, представляющий операцию (например, операция WSDL).

8.4.4. Представление XML-схемы для Пакета Сервис (Service Package)

Таблица 8.67 – XML-схема для элемента Interface

```
<xsd:element name="interface" type="tInterface" substitutionGroup="rootElement"/>
<xsd:complexType name="tInterface">
    <xsd:complexContent>
        <xsd:extension base="tRootElement">
            <xsd:sequence>
                <xsd:element ref="operation" minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
            <xsd:attribute name="implementationRef" type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 8.68 – XML-схема для элемента Operation

```
<xsd:element name="operation" type="tOperation"/>
```

```
<xsd:complexType name="tOperation">
	<xsd:complexContent>
		<xsd:extension base="tBaseElement">
			<xsd:sequence>
		<xsd:element name="inMessageRef" type="xsd:QName" minOccurs="1" maxOccurs="1"/>
		<xsd:element name="outMessageRef" type="xsd:QName" minOccurs="0" maxOccurs="1"/>
		<xsd:element name="errorRef" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="name" type="xsd:string" use="required"/>
			<xsd:attribute name="implementationRef" type="xsd:QName" use="optional"/>
		</xsd:extension>
	</xsd:complexContent>
</xsd:complexType>
```

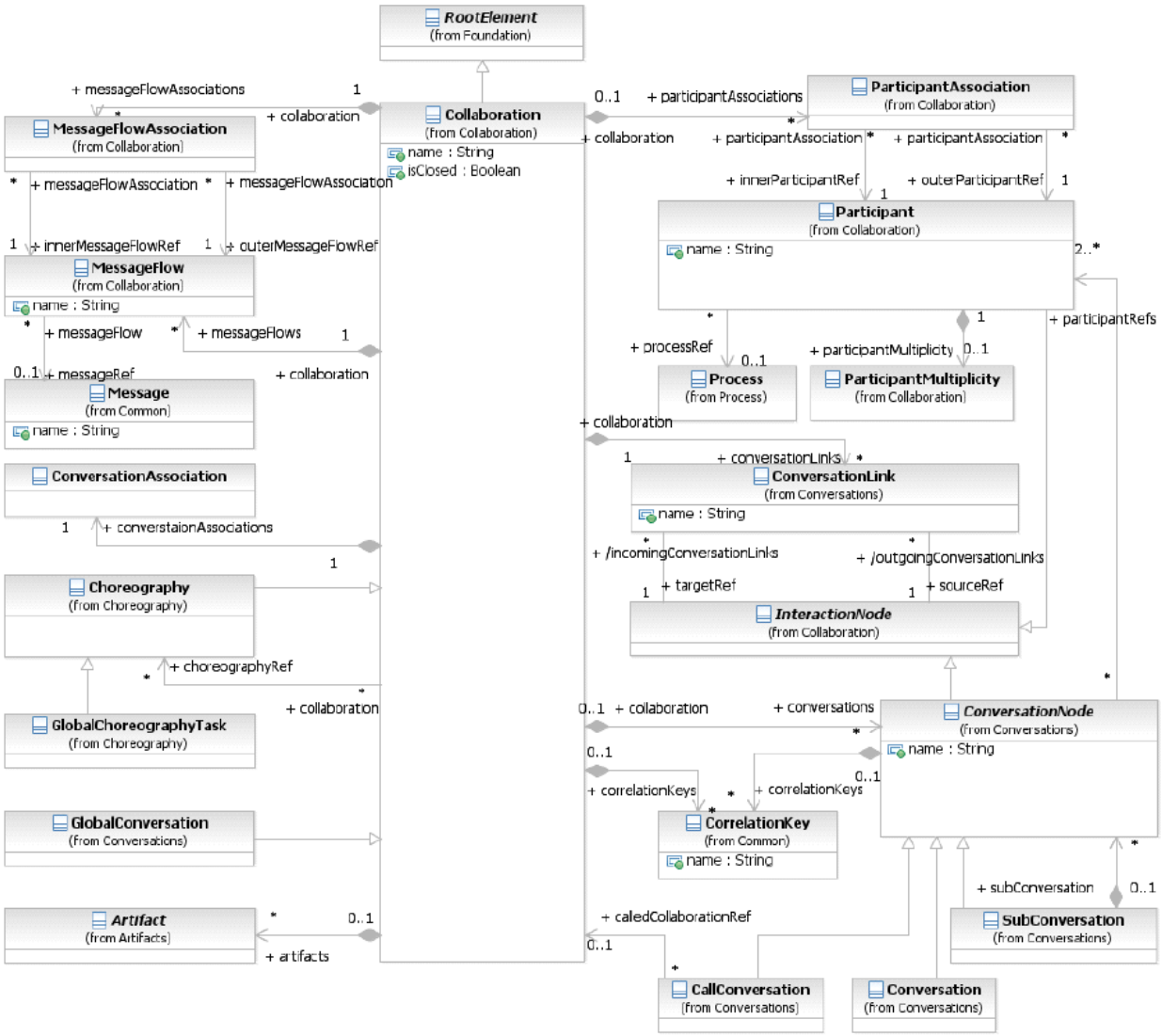
Таблица 8.69 – XML-схема для элемента EndPoint

```
<xsd:element name="endPoint" type="tEndPoint"/>
<xsd:complexType name="tEndPoint">
	<xsd:complexContent>
		<xsd:extension base="tRootElement"/>
	</xsd:complexContent>
</xsd:complexType>
```

9. Взаимодействие (Collaboration)

Примечание: Выполнение пунктов данной главы является НЕОБХОДИМЫМ условием для **Соответствия Требованиям Моделирования Хореографии BPMN**, **Соответствия Требованиям Моделирования Процесса BPMN** и **Соответствия Общим Требованиям BPMN**. Однако следование изложенной в данной главе информации НЕ ТРЕБУЕТСЯ для **Соответствия Требованиям Исполнения Процесса BPMN** или **Соответствия Требованиям Исполнения Процесса BPEL BPMN**. Для получения более подробной информации см. раздел 2.1 документа.

Пакет `Взаимодействие (Collaboration)` содержит классы, используемые для моделирования **Взаимодействия**, представляющего собой совокупность *Участников* (*Participants*), представленных в виде **Пулов** (Pools), и взаимоотношений, представленных в виде **Потоков Сообщений** (**Message Flows**). Внутри **Пулов Взаимодействия** или в **Хореографиях** между этими **Пулами** МОГУТ находиться **Процессы** (см. фигуру 9.1). **Хореография** (**Choreography**) представляет собой расширенный тип **Взаимодействия**. Если **Взаимодействие** определено, это значение содержится в `Definitions`.



Фигура 9.1 – Классы пакета Collaboration

Элемент **Collaboration** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством его связи с элементом `RootElement`. Таблица 9.1 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Collaboration**.

Таблица 9.1 – Атрибуты и ассоциации элемента Collaboration

Название атрибута	Описание/использование
name: string	Текстовое описание **Взаимодействия**.
choreographyRef: Choreography [0..*]	Ассоциация `choreographyRef` указывает **Хореографии**, которые могут отображаться между Пулами **Взаимодействия**. **Хореография** служит для определения делового соглашения (или заказа, при котором происходит обмен сообщениями) между взаимодействующими *Участниками*. Для получения более подробной информации о **Хореографии** см. Главу 11. Ассоциации `participantAssociations` (см.

	описание ниже) используются для установления соответствия между *Участниками* **Хореографии** и *Участниками* **Взаимодействия**. Ассоциации `MessageFlowAssociations` (см. описание ниже) используются для установления соответствия между **Потоками Сообщений Хореографии** и **Потоками Сообщений Взаимодействия**. Ассоциации `conversationAssociations` (см. описание ниже) используются для установления соответствия между **Обменами Сообщений Хореографии** и **Обменами Сообщений Взаимодействия**. Обратите внимание, что данный атрибут не применяется в **Хореографиях** или Глобальных Обменах Сообщениями (`GlobalConversation`), являющихся подтипами **Взаимодействия**. Таким образом, одна **Хореография** не может ссылаться на другую.
correlationKeys: CorrelationKey [0..*]	Данная ассоциация служит для указания Ключей Корреляции (CorrelationKeys), используемых для ассоциирования **Сообщений** с каким-либо **Взаимодействием**.
conversationAssociations: ConversationAssociation [0..*]	Посредством данного атрибута определяется список соответствий между **Обменами Сообщениями** ссылающегося **Взаимодействия** и **Обменами Сообщениями** другого **Взаимодействия**. Используется, если: • **Взаимодействие** ссылается на **Хореографию**.
conversations: ConversationNode [0..*]	Объединенное взаимоотношение conversations допускает наличие **Обменов Сообщениями** во **Взаимодействии** с целью группировки **Потоков Сообщений** данного **Взаимодействия** и установления ассоциаций с информацией *корреляции*, как того ТРЕБУЕТ модель определения **Взаимодействия** или **Процесса**. **Обмены Сообщениями** отображаются в том случае, если **Взаимодействие** является непосредственно **Взаимодействием**, а не **Хореографией**.
conversationLinks: ConversationLink [0..*]	Ссылки на **Обмен Сообщениями** (**Conversation Links**), используемые во **Взаимодействии**.
artifacts: Artifact [0..*]	Посредством данного атрибута определяется список Артефактов (Artifacts), содержащихся во **Взаимодействии**.
participants: Participant [0..*]	Посредством данного атрибута определяется список *Участников* (*Participants*), принимающих участие во **Взаимодействии**. Во **Взаимодействии** *Участники* отображаются в виде **Пулов**, а в **Хореографии** – в виде **Секций Участников** (**Participant Bands**) графических элементов **Задач Хореографии** (**Choreography Activities**).
participantAssociations: ParticipantAssociations	Посредством данного атрибута определяется

[0..*]	список соответствий между *Участниками* ссылающегося **Взаимодействия** и *Участниками* другого **Взаимодействия**. Используется, если: • **Взаимодействие** ссылается на **Хореографию**. • Ссылка на *определение Взаимодействия* для **Процесса** задана с помощью **Действия Вызов** (и соответствует *определению* **Взаимодействия** вызывающего **Процесса**).
messageFlow: Message Flow [0..*]	Посредством данного атрибута определяется список **Потоков Сообщений**, используемых во **Взаимодействии**. **Потоки Сообщений** отображаются во **Взаимодействии** (в виде пунктирных линий), но скрыты в **Хореографии**.
messageFlowAssociations: Message Flow Association [0..*]	Посредством данного атрибута определяется список соответствий между **Потоками Сообщений Взаимодействия** и **Потоками Сообщений** соответствующей модели. Используется, если: • **Взаимодействие** ссылается на **Хореографию**. Это означает своего рода «подключение» **Потоков Сообщений Взаимодействия** к соответствующим **Действиям Хореографии** (**Choreography Activities**).
IsClosed: boolean = false	Булевское значение, указывающее на то, могут ли **Потоки Сообщений**, смоделированные вне **Взаимодействия**, появиться при выполнении этого **Взаимодействия**. • Значение «*true*» означает, что они НЕ МОГУТ появляться. • Значение «*false*» означает, что они МОГУТ появляться.

Набор **Потоков Сообщений** (**Message Flows**) какого-либо **Взаимодействия** МОЖЕТ принадлежать **Обмену Сообщениями** (**Conversation**). **Обмен Сообщениями** (**Conversation**) представляет собой набор **Потоков Сообщений**, имеющих общее предназначение, другими словами, все они связаны с обработкой одного заказа (для получения более подробной информации об **Обмене Сообщениями** см. раздел 9.4).

9.1. Основные понятия Взаимодействия

В состав **Взаимодействия**, как правило, входят два или более **Пулов** (**Pools**), представляющих собой его *Участников* (*Participants*). Обмен **Сообщениями** (**Messages**) между *Участниками* отображается посредством **Потока Сообщений**, соединяющих два **Пула** (или объекты внутри **Пулов**). **Сообщения**, ассоциированные с **Потоками Сообщений**, также МОГУТ отображаться графически. Примеры **Взаимодействия** приведены на фигурах 9.3, 9.4 и 9.5.

Пул МОЖЕТ БЫТЬ пустым (т.е. представлять собой «черный ящик»), а МОЖЕТ содержать в себе **Процесс**. Поскольку **Хореографии** разделяют **Потоки Сообщений** внутри **Пулов**, они МОГУТ отображаться в пространстве между **Пулами**. **Взаимодействие** допускает использование любых сочетаний **Пулов**, **Процессов** и **Хореографий**.

9.1.1. Использование общих элементов BPMN во Взаимодействии

Некоторые из элементов **BPMN** являются общими как для **Процессов** и **Хореографий** (**Choreography**), так и для **Взаимодействий**. Такие элементы используются во всех перечисленных диаграммах. Следующие подразделы содержат информацию об использовании во **Взаимодействии Сообщений** (**Messages**), **Потоков Сообщений** (**Message Flows**), **Участников** (**Participants**), **Потоков Операций** (**Sequence Flows**), Артефактов (Artifacts), Корреляций (Correlations), Выражений (Expressions) и Сервисов (Services).

9.2. Пулы и Участники

Пул (**Pool**) представляет собой графическое изображение Участника (Participant) **Взаимодействия**. *Участник* (см. подраздел 9.2.1) может являться более узкой Партнерской Сущностью (PartnerEntity, к примеру, компания) либо представлять достаточно общую Партнерскую Роль (PartnerRole, к примеру, покупатель, продавец, производитель). **Пул** МОЖЕТ ссылаться, а МОЖЕТ НЕ ссылаться на **Процесс**. Пул НЕ ОБЯЗАТЕЛЬНО содержит **Процесс**, т.е. может быть «черным ящиком»,

- **Пул** представляет собой прямоугольник с острыми углами, который ДОЛЖЕН БЫТЬ выполнен жирной одинарной линией (см. фигуру 9.2).
    - Название **Пула** МОЖЕТ располагаться в любом месте данного графического элемента и иметь любое направление, однако, оно ДОЛЖНО быть отделено от содержимого **Пула** одинарной линией.
        - В случае, если **Пул** представляет собой «черный ящик», то его название МОЖЕТ располагаться в любом месте данного графического элемента без необходимости разделения.
    - На диаграмме лишь один **Пул** МОЖЕТ отображаться без границ. В случае, если диаграмма содержит несколько **Пулов**, то остальные **Пулы** (кроме одного) ДОЛЖНЫ иметь границы.

Текст, цвет, размер, а также линии, используемые для изображения **Пула**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».

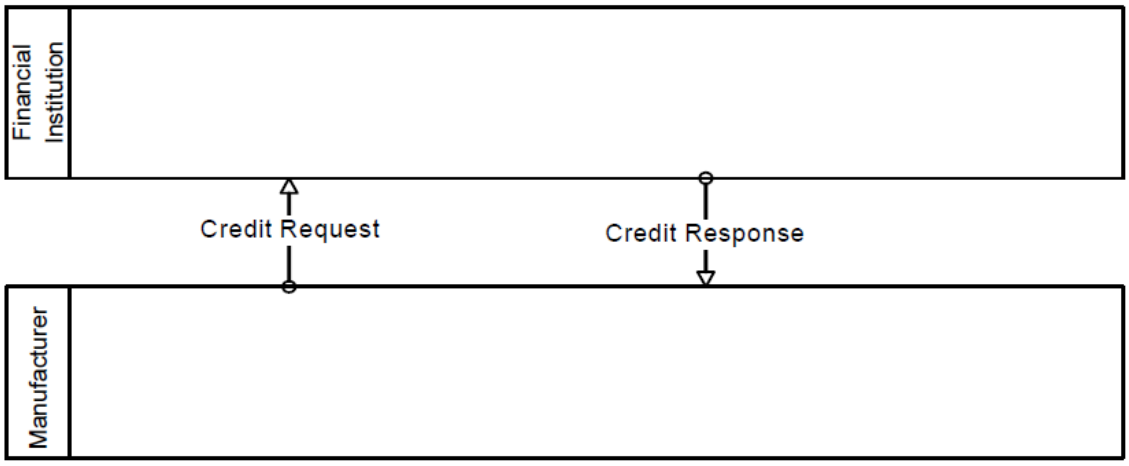
Фигура 9.2 - Пул

Для удобства работы с Диаграммой, **Пул** растянут на всю её длину (как по вертикали, так и по горизонтали). Однако на выбор размера и расположения **Пула** ограничения не налагаются. Разработчики моделей и инструменты моделирования могут использовать **Пулы** как угодно в целях сохранения на экране или напечатанной странице текущего размера Диаграммы.

Пул выступает в роли контейнера **Потоков Операций** (**Sequence Flows**) между **Действиями** (**Activities**), содержащимися в **Процессе**. **Потоки Операций** могут пересекать границы **Дорожек** (**Lanes**) внутри **Пула** (для получения более подробной информации о **Дорожках** см. подраздел 10.7), однако, не могу пересекать границ самого **Пула**. Это означает, что **Процесс** полностью находится в **Пуле**. Связь между **Пулами** отображается посредством **Потоков Сообщений**.

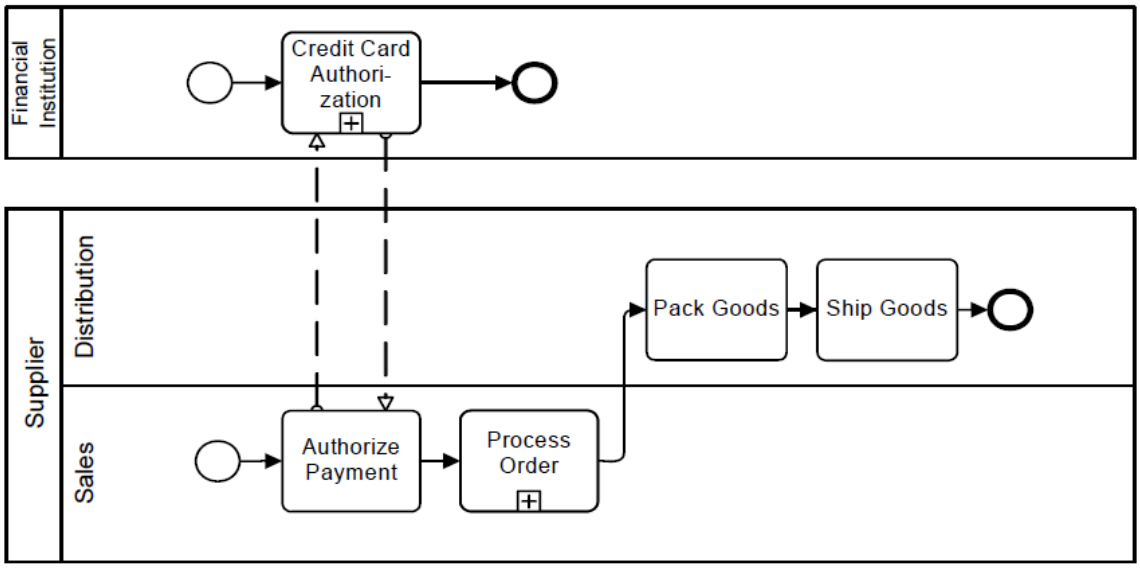
Другая особенность использования **Пула** заключается в возможности добавления в него деталей **Действия** (Activity). Соответственно, **Пул** МОЖЕТ отображаться как «Белый Ящик» (все детали показаны) или как «Черный Ящик» (все детали спрятаны). С **Пулом**, отображаемым в виде «Черного Ящика», не может быть

ассоциировано ни одного **Потока Операций**, однако, **Потоки Сообщений** могут присоединяться к границам такого **Пула** (см. фигуру 9.3).



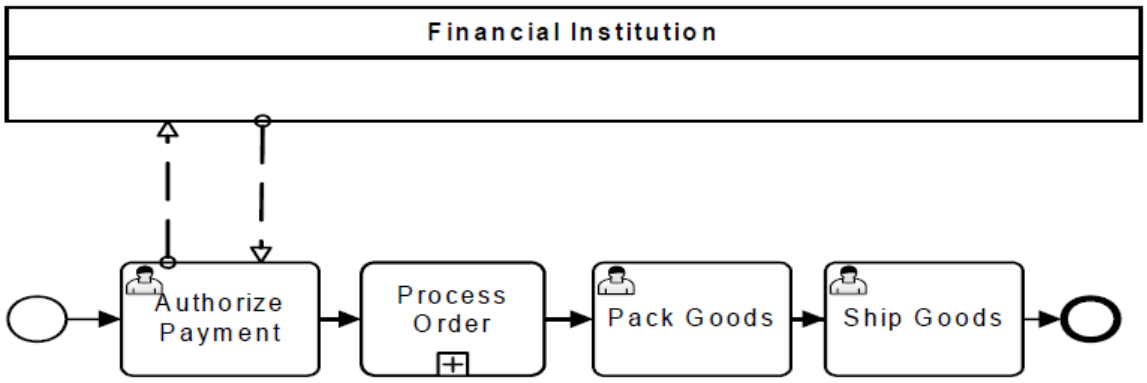
Фигура 9.3 – Потоки Сообщений, присоединенные к границам двух Пулов

Действия внутри **Пула**, отображаемого в качестве «Белого Ящика», организованы посредством **Потоков Операций**. **Потоки Сообщений** в данном случае могут пересекать границы **Пула** для соединения с соответствующими **Действиями** (см. фигуру 9.4).



Фигура 9.4 – Потоки Сообщений, соединяющие Элементы Потока внутри Пулов

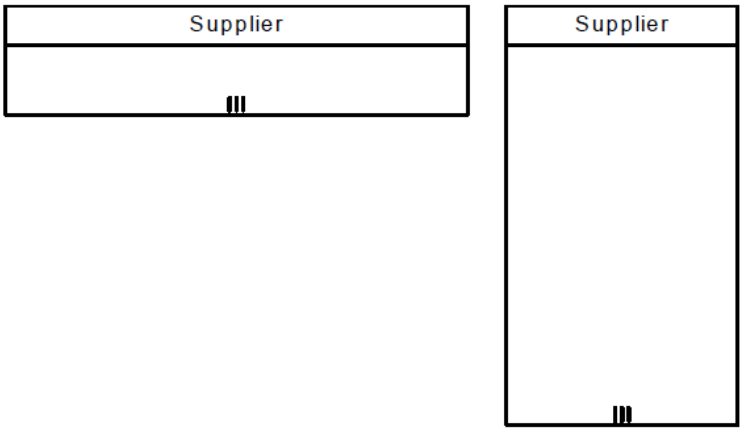
Взаимодействие может содержать два (2) или более **Пулов** (т.е. *Участников*). Однако с точки зрения разработчиков модели, **Процесс**, представляющий собой выполняемую работу, рассматривается как «внутренний» (internal) и НЕ ТРЕБУЕТ заключения в границы **Пула**, в то время как остальные **Пулы** на Диаграмме ДОЛЖНЫ отображаться с границами (см. фигуру 9.5).



Фигура 9.5 – Основной (Внутренний) Пул, не имеющий границ

В **BPMN** для **Пулов** предусмотрен маркер - маркер многоэкземплярности, который МОЖЕТ отображаться в **Пуле** (см. фигуру 9.6). Данный маркер используется в том случае, если *Участник*, указанный в **Пуле**, является *множественным*. Для получения более подробной информации о множественности *Участников* см. соответствующий пункт следующего подраздела.

- *Многоэкземплярный* маркер **Пула** ДОЛЖЕН быть выполнен в виде трех параллельно расположенных вертикальных линий.
- Маркер, если он используется, ДОЛЖЕН располагаться в центре нижней части графического элемента **Пула**.

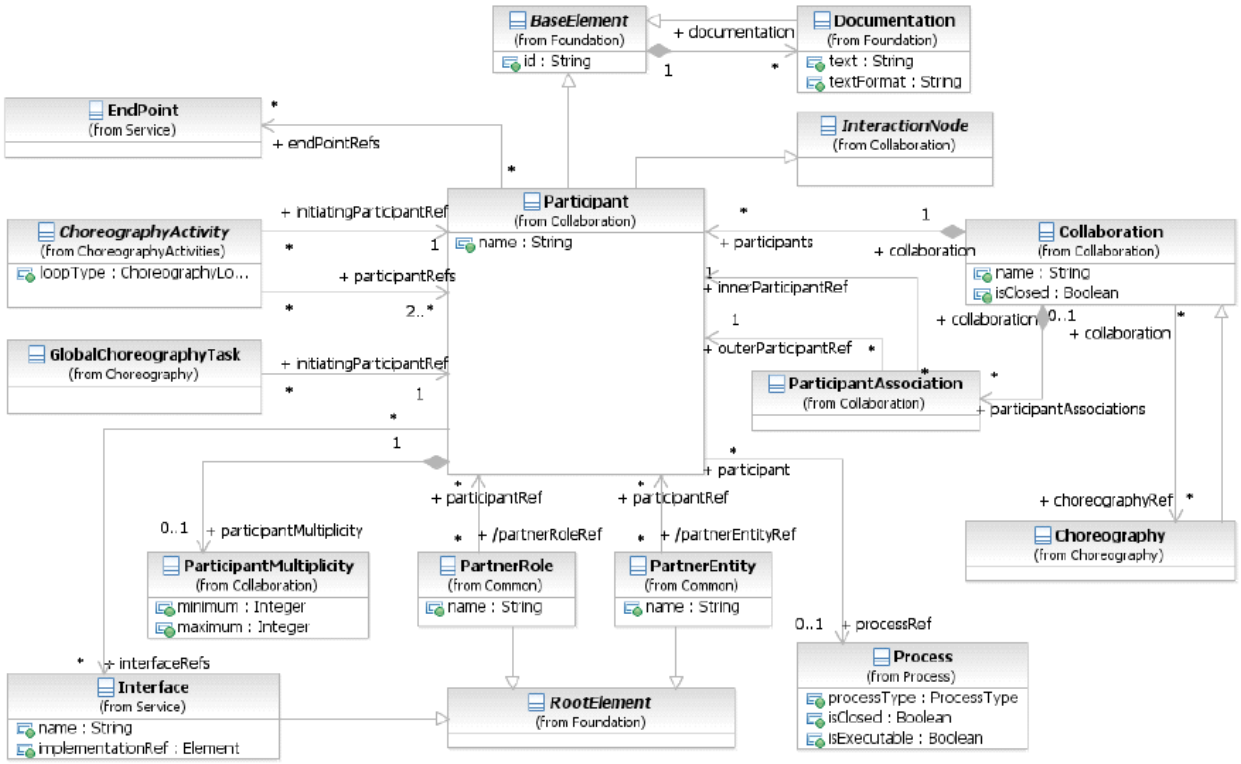


Фигура 9.6 – Пулы с маркерами Множественного Участника

9.2.1. Участники (Participants)

Участник представляет собой конкретную Партнерскую Сущность (PartnerEntity, например, компанию) и/или менее узкую Партнерскую Роль (PartnerRole, например, покупателя, продавца, производителя), являющиеся *Участниками* **Взаимодействия**. Как правило, *Участник* отвечает за выполнение **Процесса**, заключенного в его **Пуле**. Однако **Пул** МОЖЕТ существовать и без **Процесса**.

На фигуре 9.7 представлена диаграмма классов элемента *Participant* и его отношений с другими элементами **BPMN**. Если *Участник* указан, это значение хранится во **Взаимодействии**, содержащем подтипы **Хореографии** (**Choreography**), `Глобального Обмена Сообщениями (GlobalConversation)` и `Глобальной Задачи Хореографии (GlobalChoreographyTask)`.



Фигура 9.7 – Диаграмма классов элемента Participant

Элемент *Participant* наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.2 содержит информацию о дополнительных атрибутах и ассоциациях элемента *Participant*.

Таблица 9.2 – Атрибуты и ассоциации элемента Participant

Название атрибута	Описание/использование
name: string [0..1]	Текстовое описание *Участника*. `Имя` *Участника* может отображаться напрямую либо быть заменено ассоциированной `Партнерской Ролью (PartnerRole)` или `Партнерской Сущностью (PartnerEntity)`. Также возможен вариант, когда `имя Партнерской Роли` и `имя Партнерской Сущности` отображаются одновременно.
processRef: Process [0..1]	Посредством атрибута `processRef` указывается **Процесс**, который используется Участником во *Взаимодействии*. **Процесс** отображается в Пуле данного *Участника*.
partnerRoleRef: PartnerRole [0..*]	Посредством атрибута `partnerRoleRef` указывается `Партнерская Роль`, принадлежащая *Участнику* Взаимодействия. Для *Участника* МОЖЕТ быть указана как `Партнерская Роль`, так и `Партнерская Сущность`. Данный атрибут образован от атрибута `participantRefs` `Партнерской Роли`.
partnerEntityRef: PartnerEntity [0..*]	Посредством атрибута `partnerEntityRef` указывается `Партнерская Сущность`, принадлежащая *Участнику* Взаимодействия. Для

	Участника МОЖЕТ быть указана как `Партнерская Роль`, так и `Партнерская Сущность`. Данный атрибут образован от атрибута `participantRefs Партнерской Сущности`.
interfaceRef: Interface [0..*]	Посредством данной ассоциации указываются `Интерфейсы (Interfaces)`, которые поддерживает *Участник*. Определение `Интерфейса` см. в подразделе 8.4.1)
participantMultiplicity: participant- Multiplicity [0..1]	Ассоциация `participantMultiplicityRef` используется для определения *Участника*, который представляет собой более одного (1) экземпляра *Участника* в данном взаимодействии. Ниже содержится более подробное описание `ParticipantMultiplicity`.
endPointRefs: EndPoint [0..*]	Данный атрибут используется для указания адреса (или ссылки на конечную точку) конкретного сервиса для реализации *Участника*.

Партнерская Сущность (PartnerEntity)

`Партнерская Сущность (PartnerEntity)` является одним из типов *Участника* (см. подраздел выше).

Элемент `PartnerEntity` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.3 содержит информацию о дополнительных атрибутах и ассоциациях элемента `PartnerEntity`.

Таблица 9.3 – Атрибуты и ассоциации элемента PartnerEntity

Название атрибута	Описание/использование
name: string	Текстовое описание `Партнерской Сущности`.
participantRef: Participant [0..*]	Данный атрибут определяет то, каково участие `Партнерской Сущности` во **Взаимодействии** или **Хореографии** (**Choreography**).

Партнерская Роль (PartnerRole)

`Партнерская Роль (PartnerRole)` является одним из типов *Участника* (см. подраздел выше).

Элемент `PartnerRole` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.4 содержит информацию о дополнительных атрибутах и ассоциациях элемента `PartnerRole`.

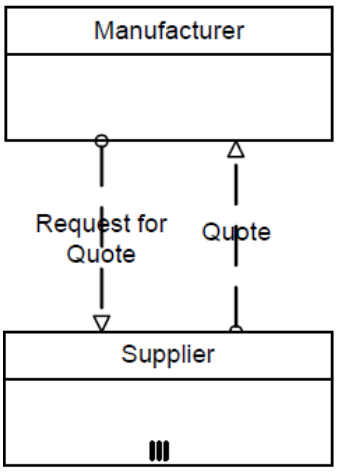
Таблица 9.4 – Атрибуты и ассоциации элемента PartnerRole

Название атрибута	Описание/использование
name: string	Текстовое описание `Партнерской Роли`.
participantRef: Participant [0..*]	Данный атрибут определяет то, каково участие `Партнерской Роли` во **Взаимодействии** или **Хореографии** (**Choreography**).

Множественность Участника

Элемент `ParticipantMultiplicity` используется для указания на множественность `Участника`.

К примеру, производитель может получить запрос от множества поставщиков, участвующих во **Взаимодействии**.



Фигура 9.8 – Пул с Множественным Участником

На фигуре 9.9 изображена диаграмма классов для Множественного Участника.

Фигура 9.9 – Диаграмма классов для Множественного Участника

Маркер *многоэкземплярности* отображается в центре нижней части **Пула** (*Участника*, см. фигуру 9.9) или **Секции Участника** в **Действии Хореографии** (см. раздел 11.4). Он используется в случае, если элемент `ParticipantMultiplicity` ассоциирован с *Участником*, а значение атрибута `maximum` не указано или равно «два и больше».

Таблица 9.5 содержит информацию об атрибутах элемента `ParticipantMultiplicity`.

Таблица 9.5 – Атрибуты элемента ParticipantMultiplicity

Название атрибута	Описание/использование
minimum: integer = 0	Посредством атрибута `minimum` указывается минимальное количество Участников, которые ДОЛЖНЫ участвовать во **Взаимодействии**. Если указано также и значение атрибута `maximum`, оно ДОЛЖНО БЫТЬ больше или равно значению данного атрибута.
maximum: integer [0..1] = 1	Посредством атрибута `maximum` указывается максимальное количество Участников, которые МОГУТ участвовать во **Взаимодействии**. Данное значение ДОЛЖНО БЫТЬ равно «1» или больше, А ТАКЖЕ ДОЛЖНО БЫТЬ равно или больше значения атрибута `minimum`.

Таблица 9.6 содержит информацию об атрибутах *Экземпляра* (*Instance*)элемента `ParticipantMultiplicity`.

Таблица 9.6 – Атрибуты Экземпляра элемента ParticipantMultiplicity

Название атрибута	Описание/использование
numParticipants: integer [0..1]	Текущее значение множественности *Участника* для данного *Экземпляра* **Хореографии** или **Взаимодействия**.

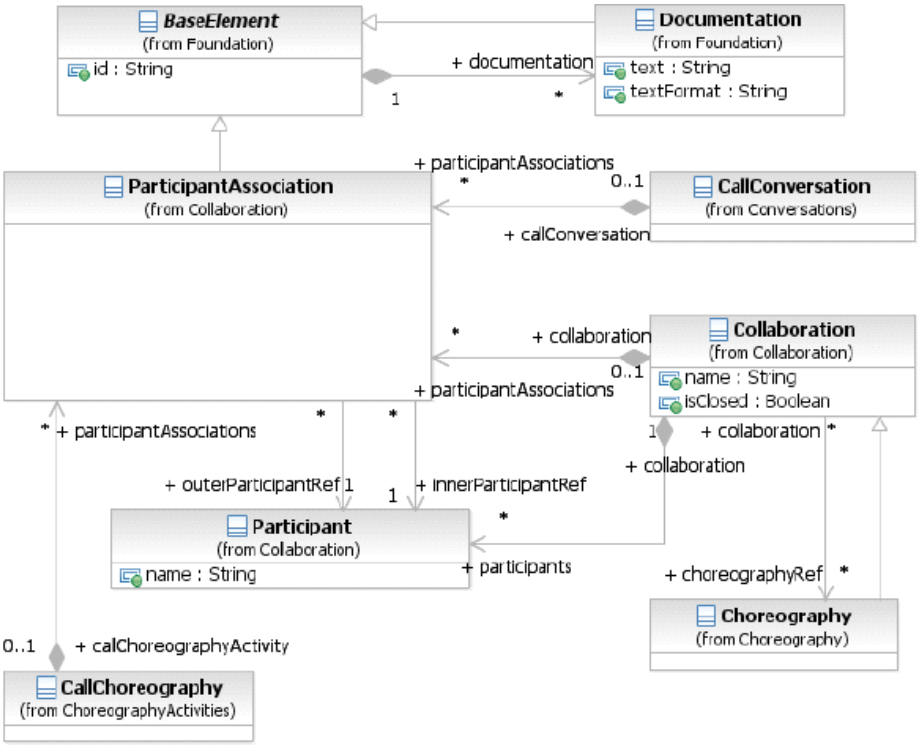
Ассоциация Участника (ParticipantAssociation)

Данные элементы используются для установки соответствия между двумя элементами, содержащими *Участников*. Время от времени могут возникать ситуации, когда *Участники* разных Диаграмм могут быть определены по-разному (т.к. они создавались по-отдельности), но при этом данные *Участники* представляют собой одно и то же. Элемент `ParticipantAssociation` служит для установки соответствия между такими *Участниками*.

Элемент `ParticipantAssociation` используется в случае, если одна (внешняя) диаграмма с *Участниками* в составе содержит другую (внутреннюю) диаграмму также с *Участниками* в составе. **BPMN** выделяет четыре случая, когда необходимо применение элемента `ParticipantAssociation`:

1. **Взаимодействие** ссылается на **Хореографию** (**Choreography**) для добавления её между **Пулов** (*Участников*) **Взаимодействия**. *Участники* **Хореографии** (внутренней диаграммы) должны соответствовать *Участникам* **Взаимодействия** (внешней диаграммы).
2. **Обмен Сообщениями типа Вызов** (**Call Conversation**) ссылается на **Взаимодействие** или `Глобальный Обмен Сообщениями (GlobalConversation)`. *Участники* **Взаимодействия** (внутренней диаграммы) должны соответствовать *Участникам*, на которых ссылается **Обмен Сообщениями типа Вызов** (внешний элемент). Любой **Обмен Сообщениями типа Вызов** содержит собственный набор элементов `ParticipantAssociations`.
3. **Хореография типа Вызов** (**Call Choreography**) ссылается на **Хореографию** или `Глобальную Задачу Хореографии (GlobalChoreographyTask)`. *Участники* **Хореографии** или `Глобальной Задачи Хореографии (внутренней диаграммы)` должны соответствовать *Участникам*, на которых ссылается **Хореография типа Вызов** (внешний элемент). Любая **Хореография типа Вызов** содержит собственный набор элементов `ParticipantAssociations`.
4. **Действие Вызов** (**Call Activity**), содержащееся внутри **Процесса** с *определяющим* **Взаимодействием**, ссылается на другой **Процесс** с *определяющим* **Взаимодействием**. *Участники определяющего* **Взаимодействия** вызываемого **Процесса** (внутренней диаграммы) должны соответствовать *Участникам определяющего* **Взаимодействия** вызывающего **Процесса** (внешней диаграммы).

Элемент `ParticipantAssociation` может быть присвоен внешней диаграмме или одному из её элементов. На фигуре 9.10 изображена диаграмма классов элемента `ParticipantAssociation`.



Фигура 9.10 - Диаграмма классов элемента ParticipantAssociation

Элемент `ParticipantAssociation` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.7 содержит информацию о дополнительных ассоциациях элемента `ParticipantAssociation`.

Таблица 9.7 – Ассоциации элемента ParticipantAssociation

Название атрибута	Описание/использование
innerParticipantRef: Participant	Посредством данного атрибута определяется *Участник* элемента, на который ссылаются (например, **Хореография**, которая будет использована во **Взаимодействии**) и который соответствует родительскому элементу (например, **Взаимодействию**).
outerParticipantRef: Participant	Посредством данного атрибута определяется *Участник* родительского элемента (например, **Взаимодействие** ссылается на **Хореографию**), соответствующий элементу, на который ссылаются (например, **Хореография**).

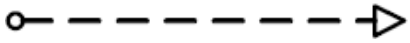
9.2.2. Дорожки (Lanes)

Дорожки (**Lane**) разграничивают внутреннее пространство **Процесса** (как правило, **Пула**) и простираются на всю длину уровня **Процесса**, как вертикально (см. фигуру 10.123), так и горизонтально (см. фигуру 10.124). Для получения более подробной информации о **Дорожках** см. раздел 10.7.

9.3. Поток Сообщений (Message Flow)

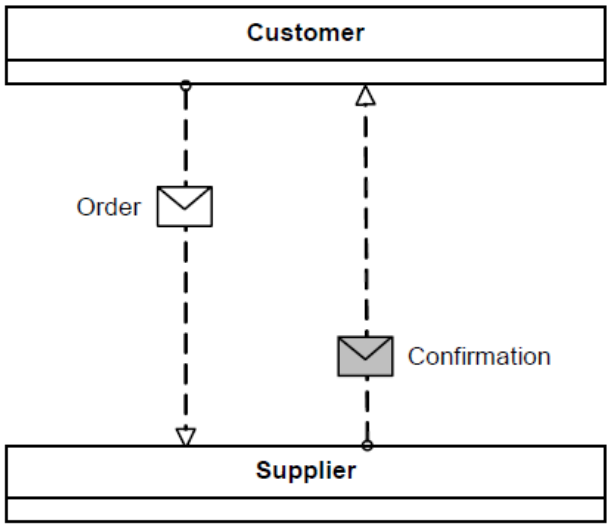
Поток Сообщений (**Message Flow**) используется для отображения обмена сообщениями между двумя `Участниками (Participants)`, готовыми эти **Сообщения** отсылать и получать.

- **Поток Сообщений** ДОЛЖЕН соединять двух разных **Пула** путем присоединения либо к границам **Пулов**, либо к Элементам Потока (Flow Objects), содержащимся в них. **Потоки Сообщений** НЕ ДОЛЖНЫ соединять объекты внутри **Пулов**.
- **Поток Сообщений** представляет собой линию, ограниченную маленьким кругом на одном конце (начало линии) и стрелкой на другом (конец линии). И круг, и стрелка отображаются без заливки. Сама линия ДОЛЖНА БЫТЬ выполнена пунктиром (см. фигуру 9.11).
    - Текст, цвет, размер, а также линии, используемые для изображения Потока Сообщений, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».



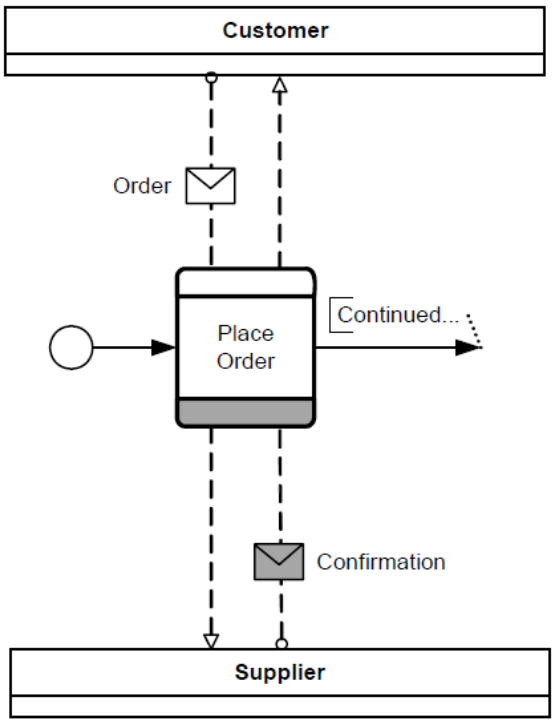
Фигура 9.11 – Поток Сообщений

На Диаграмме **Взаимодействия** (вид **Процесса Хореографии** в сочетании с **Процессами** Оркестровки) **Поток Сообщений** может быть расширен с целью показать **Сообщение**, поступающее от одного `Участника` к другому (см. фигуру 9.12).



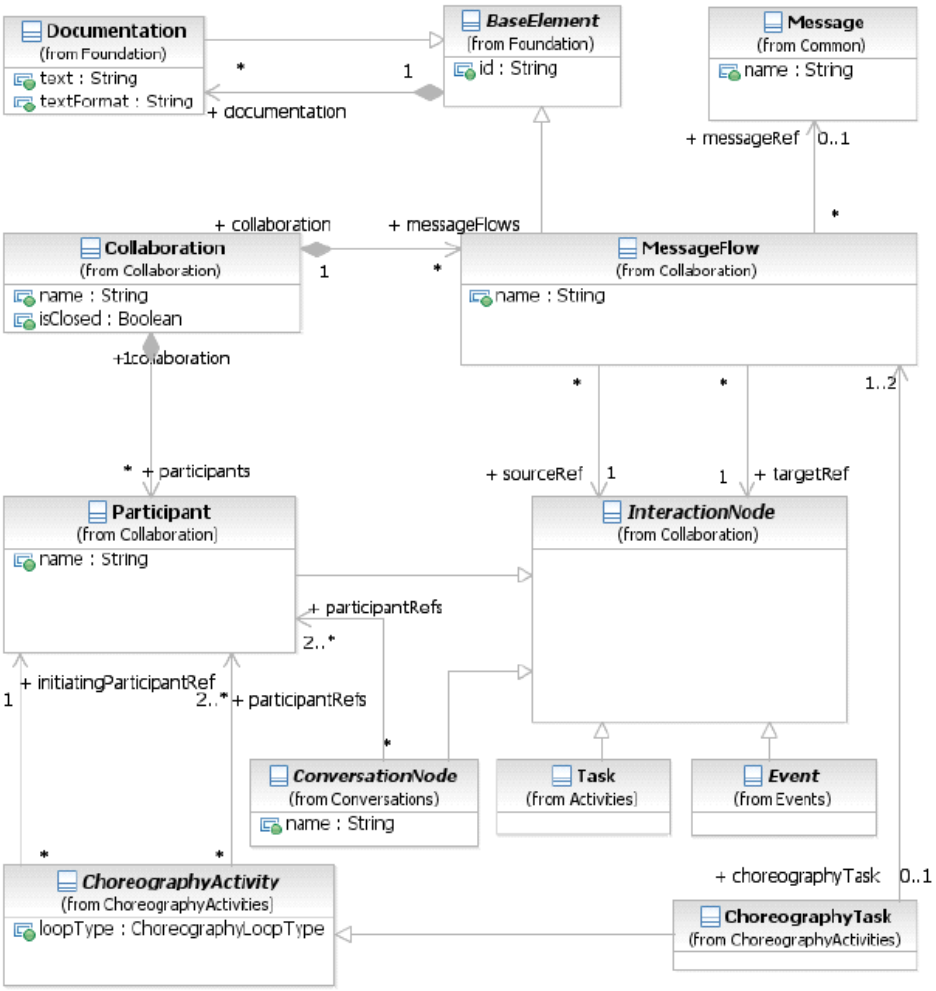
Фигура 9.12 – Поток Сообщений с Присоединенным Сообщением

В случае, если в состав **Взаимодействия** входит **Хореография** (**Choreography**), то **Поток Сообщений**, поскольку он соединяет двух `Участников`, проходит через **Задачу Хореографии** (**Choreography Task**), см. фигуру 9.13.



Фигура 9.13 – Поток Сообщений, проходящий через Задачу Хореографии

На фигуре 9.14 представлена диаграмма классов Потока Сообщений, а также его отношения с другими элементами **BPMN**. Если **Поток Сообщений** определен, это значение хранится либо во **Взаимодействии**, либо в **Хореографии** или `Глобальной Задаче Хореографии (GlobalChoreographyTask).`



Фигура 9.14 – Диаграмма классов элемента Message Flow

Элемент **Message Flow** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.8 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Message Flow**.

Таблица 9.8 – Атрибуты и ассоциации элемента Message Flow

Название атрибута	Описание/использование
name: string	Текстовое описание **Потока Сообщений**.
sourceRef: InteractionNode	Определяет `Узел Взаимодействия (InteractionNode)`, от которого отходит **Поток Операций**. Из элементов, относящихся к `Узлам Взаимодействия`, только **Пулы**/*Участники*, **Действия** (**Activities**) и **События** (**Events**) могут являться *источниками* **Потока Сообщений**.
targetRef: InteractionNode	Определяет `Узел Взаимодействия (InteractionNode)`, к которому направлен **Поток Операций**. Из элементов, относящихся к `Узлам Взаимодействия`, только **Пулы**/*Участники*, **Действия** (**Activities**) и **События** (**Events**) могут являться *целями* **Потока Сообщений**.
messageRef: Message [0..1]	Посредством ассоциации `messageRef` указывается **Сообщение**, проходящее через **Поток**

	Сообщений (для получения более подробной информации см. подраздел 8.3.11).

9.3.1. Узел Взаимодействия (Interaction Node)

В отличие от индивидуальных ассоциаций элементов, соединяющихся с **Потоками Сообщений** (см. раздел выше), элемент `InteractionNode` используется для указания другого элемента в качестве источника или цели ассоциаций **Потока Сообщений** (см. фигуру 9.14). Соединяться с Потоками Сообщений могут лишь **Пулы**/*Участники*, **Действия** (**Activities**) и **События** (**Events**). Элемент `InteractionNode` также используется для указания других элементов в качестве источников или целей **Ссылок на Обмен Сообщениями** (**Conversation Links**), см. подраздел 9.4.6.

Элемент `InteractionNode` не может иметь никаких атрибутов и ассоциаций, а также не наследует от других элементов **BPMN**. Поскольку **Пулы**/*Участники*, **Действия** и **События** обладают собственными атрибутами и ассоциациями, а также наследуют от других элементов, элементу `InteractionNode` не требуется никаких дополнительных атрибутов и ассоциаций.

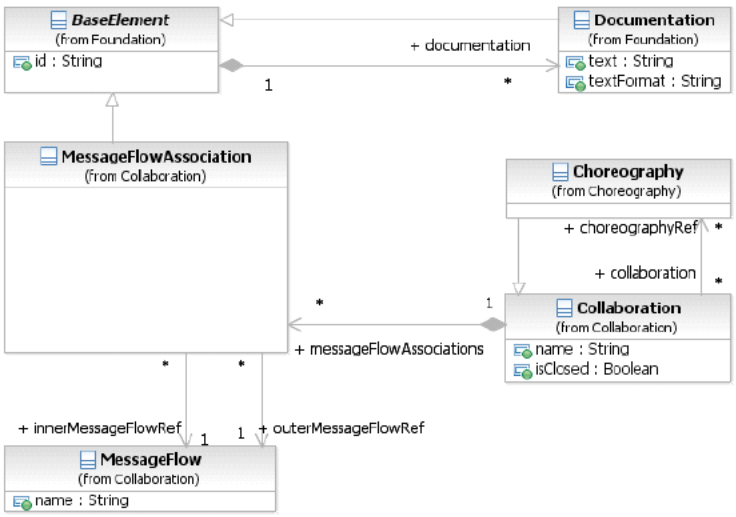
9.3.2. Ассоциации Потока Сообщений

Описанные здесь элементы используются для установки соответствия между двумя элементами, содержащими **Потоки Сообщений**. Элемент `MessageFlowAssociation` применяется для установки соответствия между **Потоками Сообщений**.

Элемент `MessageFlowAssociation` используется в случае, если одна (внешняя) диаграмма с **Потоками Сообщений** содержит другую (внутреннюю) диаграмму, в состав которой также входят **Потоки Сообщений**. Данная ассоциация применяется в случае, когда:

1. **Взаимодействие** ссылается на **Хореографию** (**Choreography**) для добавления её между **Пулов** (*Участников*) **Взаимодействия**. **Потоки Сообщений Хореографии** (внутренней диаграммы) должны соответствовать **Потокам Сообщений Взаимодействия** (внешней диаграммы).
2. **Взаимодействие** ссылается на **Обмен Сообщениями**, содержащий **Потоки Сообщений**. **Потоки Сообщений** данного **Обмена Сообщениями** могут выступать в роли частичных запросов для **Взаимодействия**. Поэтому **Потоки Сообщений Обмена Сообщениями** (внутренняя диаграмма) должны соответствовать **Потокам Сообщений Взаимодействия** (внешняя диаграмма).
3. **Хореография** ссылается на **Обмен Сообщениями**, содержащий **Потоки Сообщений**. **Потоки Сообщений** данного **Обмена Сообщениями** могут выступать в роли частичных запросов для **Хореографии**. Поэтому **Потоки Сообщений Обмена Сообщениями** (внутренняя диаграмма) должны соответствовать **Потокам Сообщений Хореографии** (внешняя диаграмма).

На фигуре 9.15 представлена диаграмма классов элемента `MessageFlowAssociation`.



Фигура 9.15 – Диаграмма классов элемента MessageFlowAssociation

Элемент `MessageFlowAssociation` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.9 содержит информацию о дополнительных атрибутах и ассоциациях элемента `MessageFlowAssociation`.

Таблица 9.9 – Атрибуты и ассоциации элемента MessageFlowAssociation

Название атрибута	Описание/использование
innerMessageFlowRef: Message Flow	Посредством данного атрибута определяется **Поток Сообщений** элемента, на который ссылаются (например, **Хореография**, которая будет использована во **Взаимодействии**) и который соответствует родительскому элементу (например, **Взаимодействию**).
outerMessageFlowRef: Message Flow	Посредством данного атрибута определяется **Поток Сообщений** родительского элемента (например, **Взаимодействие** ссылается на **Хореографию**), соответствующий элементу, на который ссылаются (например, **Хореография**).

9.4. Обмен Сообщениями (Conversations)

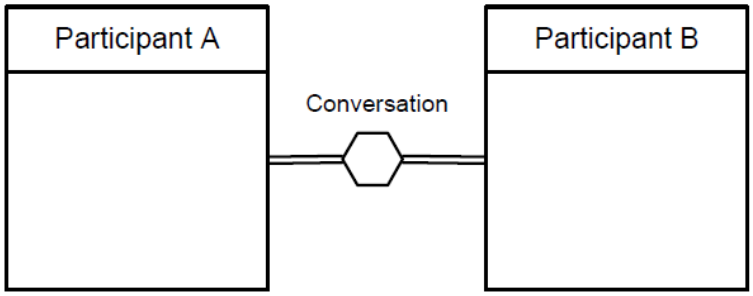
Обмен Сообщениями (**Conversation**) является частным случаем использования диаграммы **Взаимодействия** и её неформальным описанием. Выражаясь общими терминами, **Обмен Сообщениями** представляет собой упрощенный вариант **Взаимодействия**, однако, диаграмма **Обмена Сообщениями** все же поддерживает все характеристики **Взаимодействия**. В частности, *Участники* (**Пулы**) диаграммы **Обмена Сообщениями** могут содержать **Процессы** для отображения связи между **Обменом Сообщениями** и **Действиями** (**Activities**).

Диаграмма **Обмена Сообщениями** содержит два дополнительных графических элемента, которые не отображаются на других диаграммах **BPMN**:

1. **Узел Взаимодействия** (**Conversation Node**). Сюда входят **Обмен Сообщениями**, **Подчиненный Обмен Сообщениями** (**Sub-Conversation**) и **Обмен Сообщениями типа Вызов** (**Call Conversation**).
2. **Ссылка на Обмен Сообщениями** (**Conversation Link**).

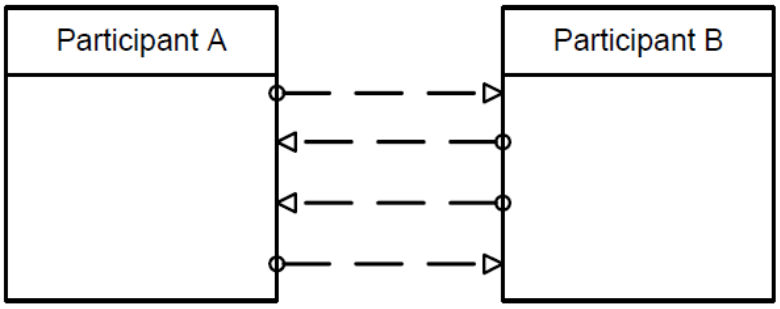
Обмен Сообщениями представляет собой логическое оформление (группировку) обмена **Сообщениями** (**Потоков Сообщений**), имеющих общую *Корреляцию* (*Correlation*). Посредством **Обмена Сообщениями** также отображается логическое отношение, характерное для обмена **Сообщениями**. В действительности логическое отношение часто связано с важными бизнес-объектами, например, «Заказом», «Транспортировкой и Доставкой», «Выставлением счета». Соответственно, **Обмен Сообщениями** ассоциирован с набором пар "имя-значение" или `Ключом корреляции` (`Correlation Key`, например, «Идентификатор Заказа» или «Идентификатор Доставки»), который записан в **Сообщениях**, подлежащих обмену. В данном случае **Сообщение** может быть направлено в конкретный *экземпляр* **Процесса**, ответственный за получение и обработку данного **Сообщения**.

На фигуре 9.16 представлен простой пример диаграммы **Обмена Сообщениями**.



Фигура 9.16 - Диаграмма Обмена Сообщениями

На фигуре 9.17 представлен один из вариантов расположенной выше диаграммы. В данном случае узел **Обмена Сообщениями** расширен с продолжением в собственные **Потоки Сообщений**. Обратите внимание, что диаграмма имеет такой же вид, что и диаграмма простого **Взаимодействия** (см. фигуру 9.3).



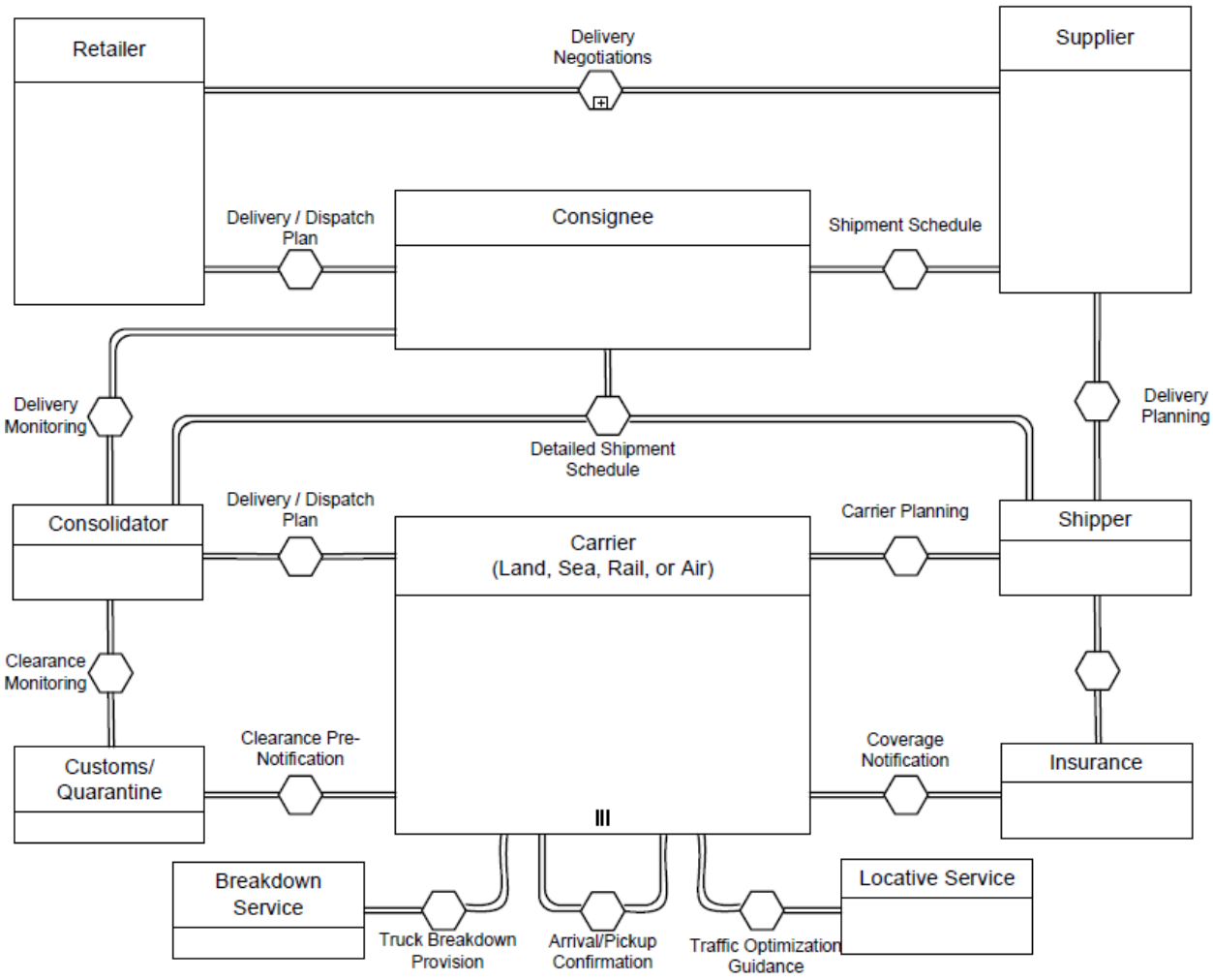
Фигура 9.17 - Диаграмма Обмена Сообщениями, расширенного в Потоки Сообщений

Случаи обмена **Сообщениями** связаны друг с другом и являются отражением удаленных бизнес-сценариев. Отношения иногда оказываются простыми, как, например, запрос, за которым следует ответ (такой пример может быть описан в качестве компонента структурного интерфейса какого-либо сервиса, например, определения операции WSDL). Однако в коммерческих операциях, управляемых **Бизнес-процессами**, отношения могут быть сложными, когда задействуется взаимный продолжительный обмен **Сообщениями**. В таких случаях отношения могут представлять собой не двусторонние, а комплексные, многосторонние **Взаимодействия**. Рассмотрим пример из логистики. Пополнение запасов товаров подразумевает следующие сценарии: формирование заказов, назначение проводника для доставки товаров по нескольким заказам, прохождение таможни/карантин, оплата и рассмотрение отводов.

Говоря об *оркестровке* **Процесса**, следует помнить, что **Обмены Сообщениями** являются значимыми для **Хореографии** (**Choreography**), однако, не отображаются в ней. Отличие состоит в том, что **Хореография** представляет собой проекцию комплексного (с несколькими Участниками) **Обмена Сообщениями**. Причиной этого является то, что обмен **Сообщениями**, моделируемый с использованием **Действий Хореографии** (**Choreography Activities**), подразумевает наличие нескольких Участников. Эта схема отличается от

оркестровки **Процесса**, где элементы отправки и получения **Сообщения** принадлежат одному Участнику. Ещё одним отличием является то, что понятие **Обмена Сообщениями** сохраняется и в **Хореографии**, и в *оркестровке*. Таким образом, обмен **Сообщениями** будет, в итоге, выполняться и в *оркестровке* **Процесса**.

Поскольку **Взаимодействие** представляет собой проекцию обмена **Сообщениями** в режиме нисходящего моделирования (в период проектирования), то абстрактное представление всех **Обменов Сообщениями**, связанных с моделируемой областью, доступно во всех диаграммах **Обменов Сообщениями**. На диаграмме **Обмена Сообщениями** (такой, как на фигуре 9.18) в виде шестиугольников показаны **Обмены Сообщениями** между *Участниками*. Благодаря этому обеспечивается вид «с высоты птичьего полета» различных **Обменов Сообщениями**, связанных с определенной областью.



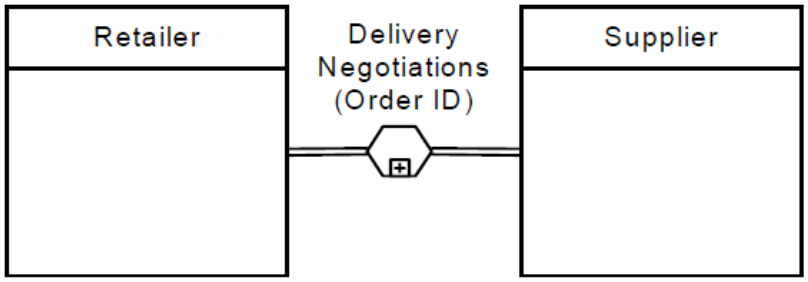
Фигура 9.18 – Диаграмма Обмена Сообщениями, изображающая несколько Обменов Сообщениями между Участниками одной области

На фигуре 9.18 изображены 13 обособленных **Обменов Сообщениями** между взаимодействующими *Участниками* (рассматривается область логистики). Участники «*Retailer*» (продавец) и «*Supplier*» (поставщик) участвуют в **Обмене Сообщениями** «*Delivery Negotiations*» (переговоры о доставке), а *Участник* «*Consignee*» (грузополучатель) ведет переговоры с *Участниками* «*Retailer*» и «*Supplier*» посредством **Обменов Сообщениями** «*Delivery/Dispatch Plan*» (план отправки и доставки) и «*Shipment Schedule*» (схема доставки) соответственно. В **Обмен Сообщениями** МОГУТ БЫТЬ вовлечены более двух участников (например, в **Обмене Сообщениями** «*Detailed Shipment Schedule*» (детальная схема доставки) могут участвовать «*Consignee*», «*Consolidator*» (консолидатор) и «*Shipper*»). Для того, чтобы указать количество вовлеченных в **Обмен Сообщениями** Участников, необходимо указать ассоциации между этими Участниками и **Обменом Сообщениями**. К примеру, в **Обмене Сообщениями** «*Deliver Negotiations*» один *экземпляр* Участника

«*Retailer*» ведет переговоры с другим (одним) *экземпляром* Участника «*Supplier*». Однако один экземпляр Участника «*Shipper*» ведет обсуждение с несколькими экземплярами Участника «*Carrier*» (на диаграмме такой Участник обозначен многоэкземплярным маркером Пула) в **Обмене Сообщениями** «*Carrier Planning*» (планирование перевозки). Обратите внимание, что в рамках **Обмена Сообщениями** множественность (многоэкземплярность) подразумевает наличие одного и более экземпляров (а не от нуля и более).

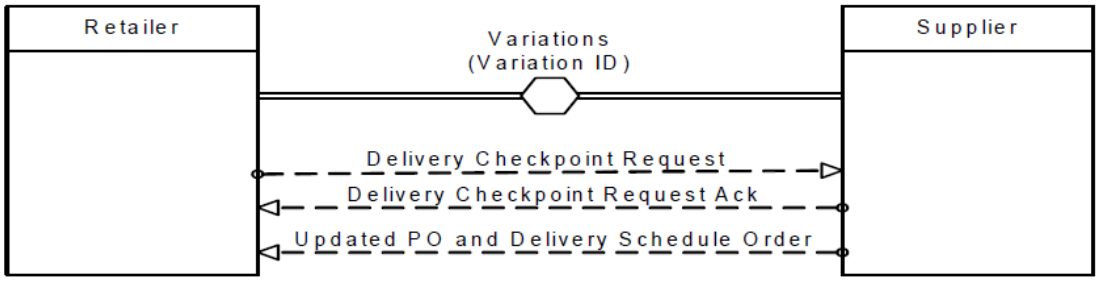
Поведение различных **Обменов Сообщениями** моделируется в отдельных **Хореографиях** (**Choreographies**), где детализируется последовательность обмена **Сообщениями**. В действительности тесно связанные **Обмены Сообщениями** могут использоваться совместно в одной и той же модели **Хореографии**, к примеру, обмен **Сообщениями** в «*Delivery Negotiation*» ведет к **Обменам Сообщениями** «*Shipment Schedule*», «*Delivery Planning*» и «*Delivery/Dispatch*», которые, в свою очередь, могут использоваться вместе в одной **Хореографии**. При этом они также по отдельности могут входить в состав разных **Хореографий.**

На фигуре 9.19 изображено подмножество более крупного **Обмена Сообщениями**, представленного на фигуре 9.18. На фигурах 9.20 и 9.21 более детально показан **Подчиненный Обмен Сообщениями** «Delivery Negotiations». Он растягивает **Обмен Сообщениями** за счет **Потоков Сообщений**, благодаря которым обеспечивается структура **Обмена Сообщениями** и не нарушается последовательность в нем. На фигуре 9.19 также изображен `Ключ Корреляции (CorrelationKey)`, входящий в состав **Потоков Сообщений** данного **Обмена Сообщениями**. К примеру, «*Order Id*» (ID заказа) необходим для всех **Сообщений**, составляющих **Потоки Сообщений** в «*Delivery Negotiation*». Некоторые из этих **Потоков Сообщений** требуют наличия *Variation Id* (это необходимо для работы с видами поставок на основании продукции).



Фигура 9.19 – Пример Подчиненного Обмена Сообщениями

На фигуре 9.20 отображен **Подчиненный Обмен Сообщениями**, представленный на фигуре 9.19, который расширен с помощью **Потоков Сообщений** и **Обмена Сообщениями** более высокого уровня.



Фигура 9.20 – Пример Подчиненного Обмена Сообщениями, расширенного Обменом Сообщениями и Потоками Сообщений

На фигуре 9.21 отображен **Обмен Сообщениями**, представленный на фигуре 9.20, который также расширен с помощью набора **Потоков Сообщений** в сочетании с предыдущими **Потоками Сообщений**. Обратите внимание, что вновь открытые **Потоки Сообщений**, принадлежащие **Обмену Сообщениями** более низкого уровня, взаимосвязаны посредством `Ключа Корреляции (CorrelationKey)` как **Обмена Сообщениями** более низкого уровня (*Variation Id*), так и **Подчиненного Обмена Сообщениями** более высокого уровня (*Order Id*).

Фигура 9.21 – Пример полностью развернутого Подчиненного Обмена Сообщениями

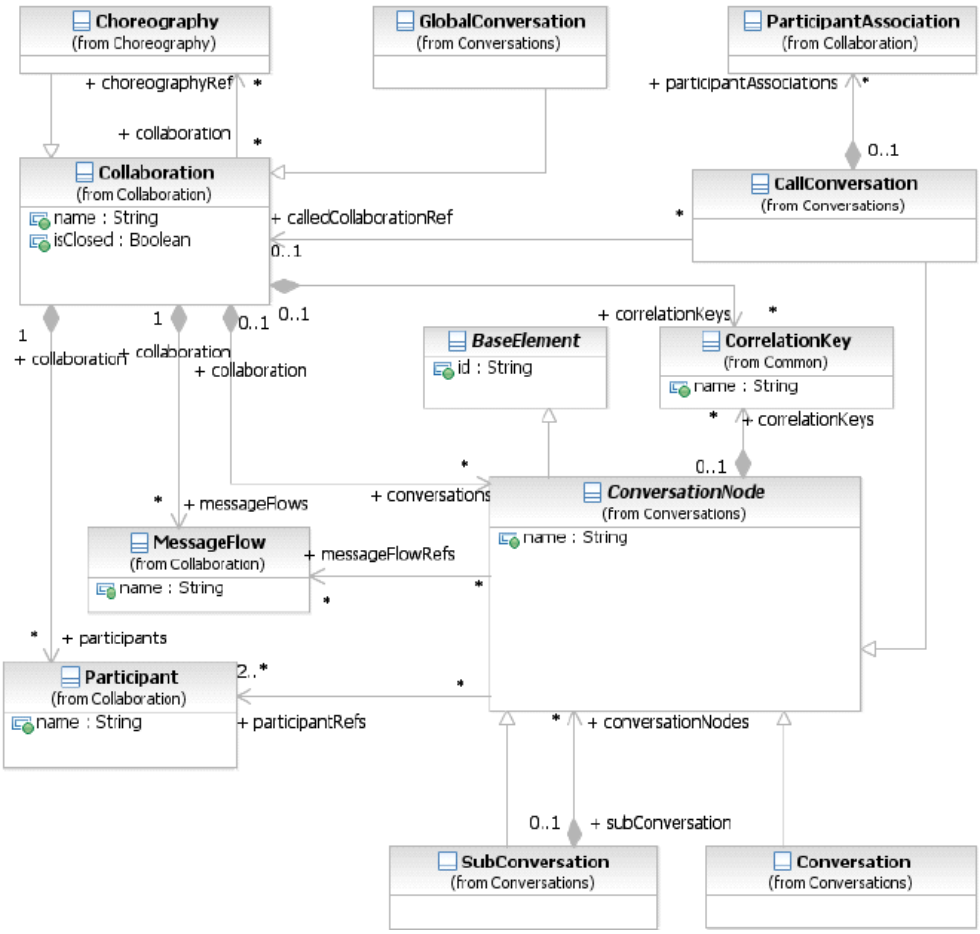
На фигуре 9.19 изображен **Обмен Сообщениями** с иерархической структурой, которая просматривается в одном наборе **Потоков Сообщений**, находящемся в родительско-дочерних отношениях с другим. В частности, после «Planned Order Variations» на родительском уровне (использовался Order Id) **Потоки Сообщений** дочернего уровня следуют до «*Retailer Order and Delivery Variations Ack*» (использовались *Variation Id* и Order Id). Оставшиеся **Потоки Сообщений** (использовался *Order Id*) находятся на родительском уровне. Дочерний **Обмен Сообщениями** является частью родительского **Обмена Сообщениями**. Вложенный **Обмен Сообщениями** графически отмечен знаком «+», указывающим на то, что **Подчиненный Обмен Сообщениями** или **Обмен Сообщениями типа Вызов** вызывает **Взаимодействие**. Вложенный **Обмен Сообщениями** может охватить любое количество уровней.

Отношения подчинения между **Обменами Сообщениями** могут частично перекрываться. Это происходит тогда, когда в двух или более **Обменах Сообщениями** происходят общие для них двоих отправка и получение **Сообщений**. Как изображено на фигуре 9.18, **Сообщение** отсылается как часть «Detailed Shipment Schedule» (используется Carrier Schedule Id) с целью активации «Delivery Monitoring» (используется Shipment Id). В ходе операции «Delivery Monitoring» **Сообщение** может быть отправлено в «Detailed Shipment Schedule» с целью запросить информацию об изменениях в случае, если возникнут какие-то проблемы при транспортировке.

Разделение и *слияние* относятся к сценариям перекрытия. *Разделение* **Обмена Сообщениями** возникает тогда, когда обмен **Сообщением**, являющимся частью данного **Обмена Сообщениями**, происходит между двумя или более *Участниками*, которые одновременно активируют новый удаленный **Обмен Сообщениями** (между этими же или другими *Участниками*). При разделении **Обмена Сообщениями** ни последующего обмена **Сообщениями**, ни последующего слияния **Обменов Сообщениями** не происходит. Примером может служить **Обмен Сообщениями** «Delivery Planning», ведущий к «Carrier Planning» и «Special Cover». *Слияние* **Обмена Сообщениями** происходит тогда, когда несколько этих элементов объединяются в один **Обмен Сообщениями**. При этом в исходных **Обменах Сообщениями** отправка и принятие **Сообщений** прекращается, что означает завершение **Обменов Сообщениями**. Итак, *разделение* и *слияние* представляют собой *рефакторинг* **Обменов Сообщениями** и подразумевают разделение одного **Обмена Сообщениями** на параллельные и последующее их объединение.

9.4.1. Узел Обмена Сообщениями (Conversation Node)

`ConversationNode` представляет собой абстрактный суперкласс для всех элементов, которые могут содержать элементы **Обмена Сообщениями** диаграммы **Взаимодействия**. Такими элементами являются **Обмен Сообщениями** (**Conversation**, см. подраздел 9.4.2), **Подчиненный Обмен Сообщениями** (**Sub-Conversation**, см. подраздел 9.4.3) и **Обмен Сообщениями типа Вызов** (**Call Conversation**, см. подраздел 9.4.4).



Фигура 9.22 – Метамодель взаимосвязанных элементов Узла Обмена Сообщениями

Элементы `ConversationNode` соединяется с *Участниками* посредством **Ссылок на Обмен Сообщениями** (**Conversation Links**, см. подраздел 9.4.6).

Элемент `ConversationNode` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.10 содержит информацию о дополнительных атрибутах и ассоциациях элемента `ConversationNode`.

Таблица 9.10 – Ассоциации элемента ConversationNode

Название атрибута	Описание/использование
name: string [0..1]	Текстовое описание элемента `ConversationNode`.
participantRefs: Partici-pant [2..*]	Предоставляет список *Участников*, задействованных в `Узле Обмена Сообщениями` согласно списку родительского **Обмена Сообщениями** `Узла Обмена Сообщениями`. Данная ссылка отображается посредством элемента **Ссылка на Обмен Сообщениями** (**Conversation Links**, см. подраздел 9.4.6).
messageFlowRefs: MessageFlow [0..*]	Ссылка на все **Потоки Сообщений** (и, соответственно, на **Сообщения**), сгруппированные посредством **Обмена Сообщениями**.
correlationKeys: CorrelationKey [0..*]	Список Ключей Корреляции `Узла Обмена Сообщениями`, используемые для группировки

	Потоков Сообщений Узла Обмена Сообщениями.

9.4.2. Обмен Сообщениями (Conversation)

Обмен Сообщениями (**Conversation**) представляет собой простой элемент, предназначенный для диаграммы **Обмена Сообщениями** (**Взаимодействия**). Является набором **Потоков Сообщений** (**Message Flows**), сгруппированных по общему принципу и\или в соответствии с Ключом Корреляции (CorrelationKey).

- **Обмен Сообщениями** (**Conversation**) представляет собой шестиугольник, который ДОЛЖЕН БЫТЬ выполнен тонкой одинарной линией (см. фигуру 9.23).

Фигура 9.23 – Обмен Сообщениями

Элемент **Conversation** наследует атрибуты и ассоциации элемента ConversationNode (см. таблицу 9.10), однако, не может иметь каких-либо других атрибутов или ассоциаций.

9.4.3. Подчиненный Обмен Сообщениями (Sub-Conversation)

Подчиненный Обмен Сообщениями (**Sub-Conversation**) является Узлом Обмена Сообщениями (ConversationNode). Представляет собой иерархическое деление внутри родительского **Взаимодействия**. Данный графический элемент расположен внутри **Взаимодействия**, однако, он может быть «открытым» для того, чтобы посредством него были показаны детали нижнего уровня **Обмена Сообщениями**, состоящего из **Потоков Сообщений** (**Message Flows**), **Обменов Сообщениями** и/или других **Подчиненных Обменов Сообщениями**. *Участники* (*Participants*) **Подчиненного Обмена Сообщениями** являются *Участниками* его родительского **Обмена Сообщениями**.

- **Подчиненный Обмен Сообщениями** представляет собой шестиугольник, который ДОЛЖЕН БЫТЬ выполнен тонкой одинарной линией (см. фигуру 9.24).
    - Маркер **Подчиненного Обмена Сообщениями** ДОЛЖЕН представлять собой небольшой квадрат с расположенным в нем знаком «+» и находиться в центре нижней части графического элемента **Подчиненного Обмена Сообщениями**.

Фигура 9.24 – Составной элемент Обмена Сообщениями

Элемент **Sub-Conversation** наследует атрибуты и ассоциации элемента ConversationNode (см. таблицу 9.10). Таблица 9.11 содержит информацию о дополнительных ассоциациях элемента **Sub-Conversation**.

Таблица 9.11 – Ассоциации элемента Sub-Conversation

Название атрибута	Описание/использование
conversationNodes: ConversationNode [0..*]	Объединяющая связь модели Узла Обмена

	Сообщениями допускает присутствие в **Подчиненном Обмене Сообщениями** других Узлов Обмена Сообщениями. Благодаря этому осуществляется группировка **Потоков Сообщений Подчиненного Обмена Сообщениями** и установка ассоциаций с коррелятивной информацией.

9.4.4. Обмен Сообщениями типа Вызов (Call Conversation)

Обмен Сообщениями типа Вызов указывает место **Обмена Сообщениями** (**Взаимодействия**), где используется глобальный **Обмен Сообщениями** или элемент GlobalConversation.

- В случае, если **Обмен Сообщениями типа Вызов** вызывает Глобальный Обмен Сообщениями, то отображаться он будет так же, как стандартный **Обмен Сообщениями**, однако, его границы ДОЛЖНЫ БЫТЬ выполнены жирной линией (см. фигуру 9.25).
- В случае, если **Обмен Сообщениями типа Вызов** вызывает **Взаимодействие**, то отображаться он будет так же, как **Подчиненный Обмен Сообщениями**, однако, его границы ДОЛЖНЫ БЫТЬ выполнены жирной линией (см. фигуру 9.26).

Фигура 9.25 - Обмен Сообщениями типа Вызов, вызывающий Глобальный Обмен Сообщениями

Фигура 9.26 - Обмен Сообщениями типа Вызов, вызывающий Взаимодействие

Элемент **Call Conversation** наследует атрибуты и ассоциации элемента ConversationNode (см. таблицу 9.10). Таблица 9.12 содержит информацию о дополнительных ассоциациях элемента **Call Conversation**.

Таблица 9.12 – Ассоциации элемента Call Conversation

Название атрибута	Описание/использование
calledCollaborationRef: Collaboratioin [0..1]	Элемент, который необходимо вызвать, МОЖЕТ являться либо **Взаимодействием**, либо **Глобальным Обменом Сообщениями**. Вызываемый элемент НЕ ДОЛЖЕН быть **Хореографией** или Глобальной Задачей Хореографии (оба этих элемента являются подтипами **Взаимодействия**).
participantAssociations: Participant Association [0..*]	Посредством данного атрибута *Участники* родительского **Обмена Сообщениями** получают список соответствий от *Участников* соответствующего Глобального Обмена Сообщениями или **Обмена Сообщениями**.

Примечание – атрибут `messageFlowRef Узла Обмена Сообщениями` не применяется в **Обмене Сообщениями типа Вызов**.

9.4.5. Глобальный Обмен Сообщениями (Global Conversation)

`Глобальный Обмен Сообщениями (GlobalConversation)` является повторно используемым простым элементом **Обмена Сообщениями**, который вызывается из любого **Взаимодействия** или **Обмена Сообщениями типа Вызов**.

Элемент `GlobalConversation` наследует атрибуты и ассоциации элемента **Collaboration** (см. таблицу 9.1), однако, не может иметь каких-либо других дополнительных атрибутов или ассоциаций.

`Глобальный Обмен Сообщениями` представляет собой ограниченный тип **Взаимодействия**, т.е. «пустое **Взаимодействие**».

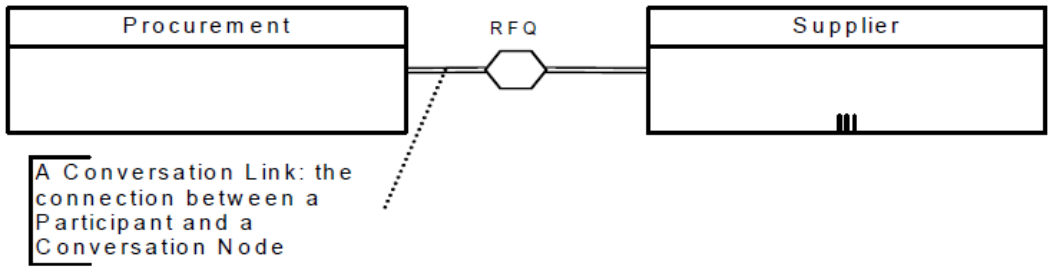
- `Глобальный Обмен Сообщениями` не должен содержать `Узлов Обмена Сообщениями`.

Поскольку в `Глобальном Обмене Сообщениями` не содержится никаких Элементов Потока (Flow Elements), для него не требуется использование элементов `MessageFlowAssociations (Ассоциаций с Элементами Потока), ParticipantAssociations (Ассоциаций с Участниками), ConversationAssociations (Ассоциаций с Обменом Сообщениями)` или **Артефактов** (**Artifacts**). Чаще всего `Глобальный Обмен Сообщениями` представляет собой повторно используемых *Участников* (*Participants*), **Потоков Сообщений** (**Message Flows**) или `Ключей Корреляции (CorrelationKeys)`. Атрибут `choreographyRef` **Взаимодействия** не может быть применен в `Глобальном Обмене Сообщениями`.

9.4.6. Ссылка на Обмен Сообщениями (Conversation Link)

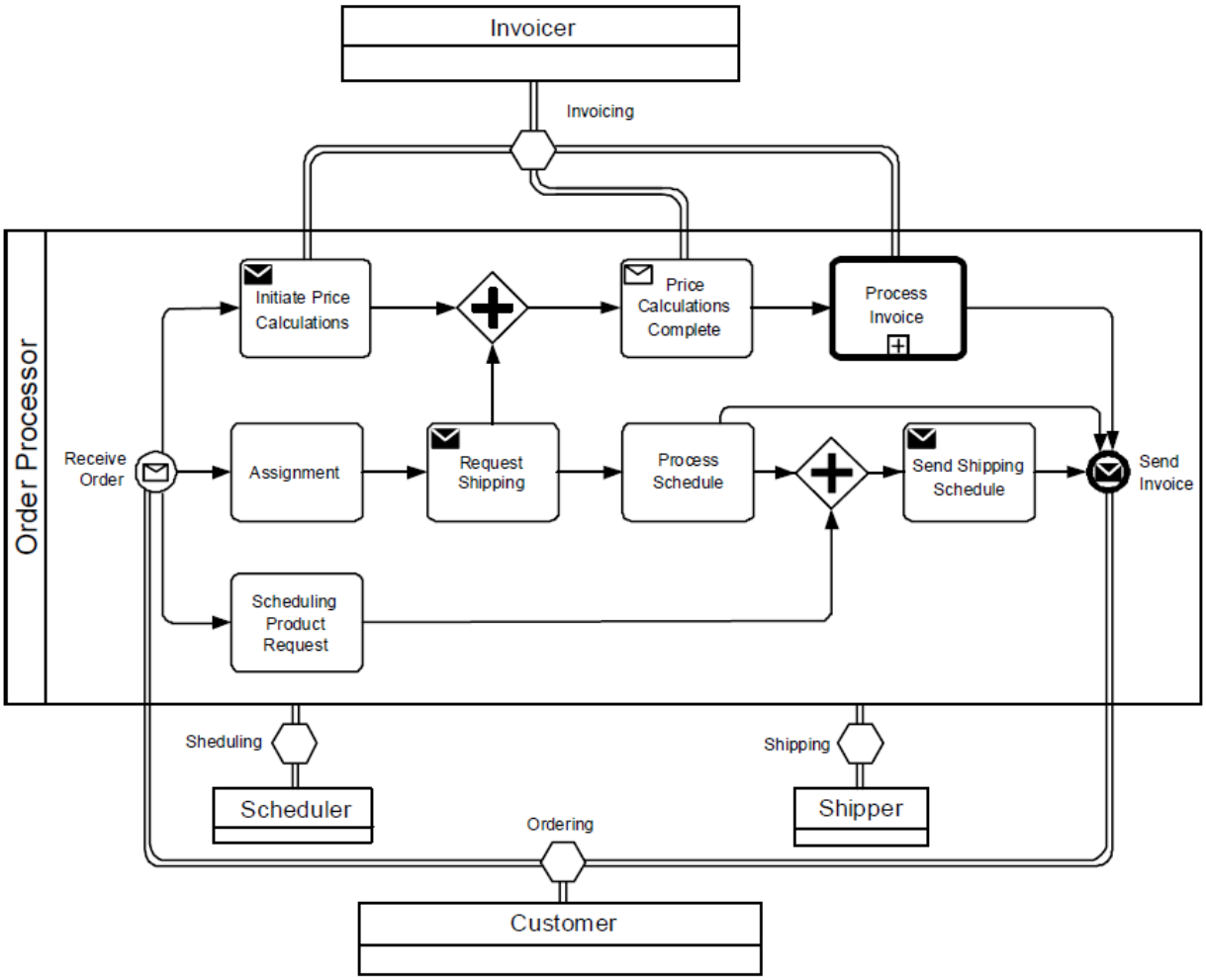
Ссылки на Обмен Сообщениями (**Conversation Links**) используются для установления связи между `Узлами Обмена Сообщениями (ConversationNodes)` и *Участниками* (*Participants*), т.е. **Пулами** (**Pools**). Связь может быть направлена как от *Участников*, так к ним (см. фигуру 9.27).

- Графический элемент **Ссылка на Обмен Сообщениями** ДОЛЖЕН БЫТЬ выполнен двойной тонкой линией.

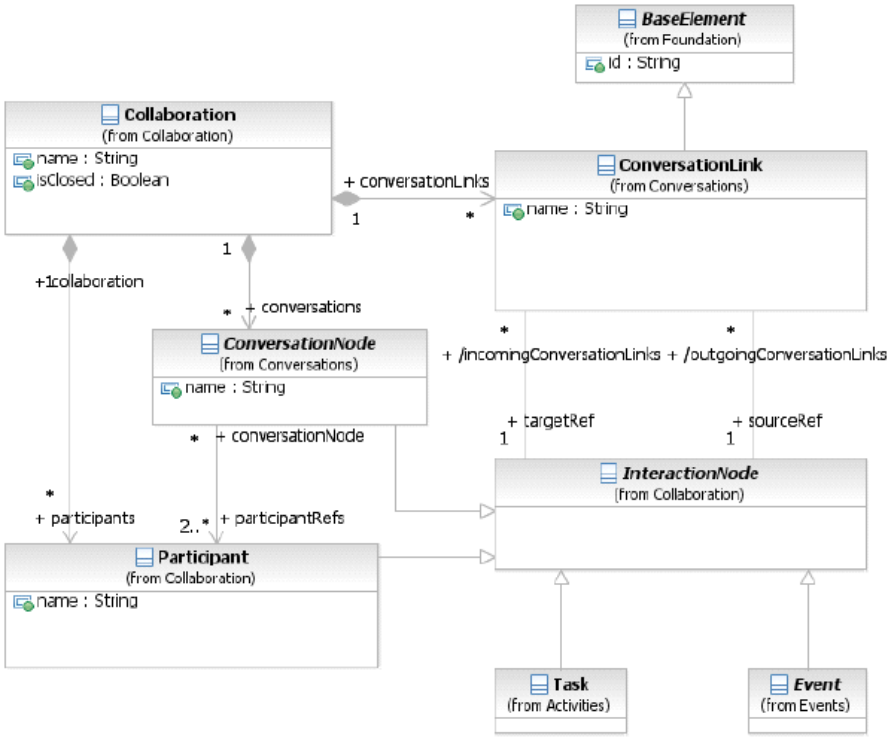


Фигура 9.27 - Ссылка на Обмен Сообщениями

Как показано на фигуре 9.28, **Процесс** появляется в **Пулах** (Участниках) диаграммы **Обмена Сообщениями**. **Обмены Сообщениями** типа выставление счетов и оформление заказов содержат ссылки, ведущие к **Действиям** (**Activities**) и **Событиям** (**Events**) **Процесса**, относящегося к Обработке Заказа. Другие два **Обмена Сообщениями** не имеют «расширенных» ссылок. **Ссылки на Обмен Сообщениями**, которые ведут к **Действиям**, не являющимся **Задачами** типа **Отправка** и **Получение**, означают, что на каком-либо уровне вложения с помощью этих **Действий** будут отправлены и получены **Сообщения** данного **Обмена Сообщениями**.



Фигура 9.28 – Ссылки на Обмен Сообщениями, ведущие к Действиям и Событиям



Фигура 9.29 – Метамодель элементов, связанных с элементом Ссылка на Обмен Сообщениями

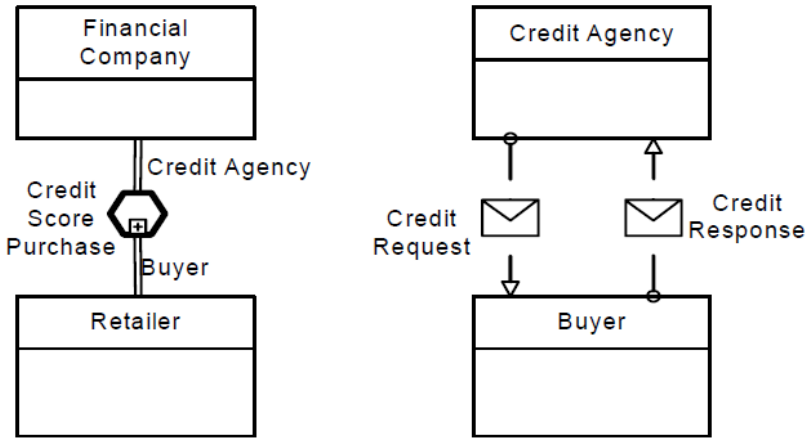
Элемент **Conversation Link** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.13 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Conversation Link**.

Таблица 9.13 – Атрибуты и ассоциации элемента Conversation Link

Название атрибута	Описание/использование
name: string [0..1]	Посредством данного атрибута указывается название **Ссылки на Обмен Сообщениями**.
sourceRef: InteractionNode	Посредством данного атрибута указывается `Узел Обмена Сообщениями (InteractionNode)`, от которого ведет **Ссылка на Обмен Сообщениями**. **Ссылка на Обмен Сообщениями** ДОЛЖНА вести лишь к одному `Узлу Обмена Сообщениями`. В случае, если значение данного атрибута не равно «`ConversationNode`», то значение атрибута `targetRef` ДОЛЖНО БЫТЬ равно «`ConversationNode`».
targetRef: InteractionNode	Посредством данного атрибута указывается `Узел Обмена Сообщениями (InteractionNode)`, к которому ведет **Ссылка на Обмен Сообщениями**. **Ссылка на Обмен Сообщениями** ДОЛЖНА вести лишь к одному `Узлу Обмена Сообщениями`. В случае, если значение данного атрибута не равно «`ConversationNode`», то значение атрибута `sourceRef` ДОЛЖНО БЫТЬ равно «`ConversationNode`».

Ссылки на Обмен Сообщениями, используемые в **Обменах Сообщениями типа Вызов**, служат для указания имен *Участников* во вложенном или глобальном **Взаимодействии** (указано в элементах

`ParticipantAssociations`). К примеру, на фигуре 9.30 (слева направо) отображены **Взаимодействие** и **Обмен Сообщениями типа Вызов**, ведущий к **Взаимодействию**. **Ссылки на Обмен Сообщениями**, отображаемые слева, указывают на то, какие *Участники* вызываемого **Взаимодействия** (справа) соотносятся с *Участниками* вызывающего **Взаимодействия** (слева). К примеру, *Участники*, относящиеся к расположенному справа «Credit Agency» (кредитному агентству), соотносятся с *Участником* «Financial Company» (финансовой компании) слева. Посредством элементов `ParticipantAssociations` (на диаграмме не отображаются) каждый из *Участников* расположенного слева **Взаимодействия** привязывается к *Участнику* **Взаимодействия** справа. Эти элементы могут быть использованы для отображения имен *Участников* во вложенном или глобальном **Взаимодействии**.



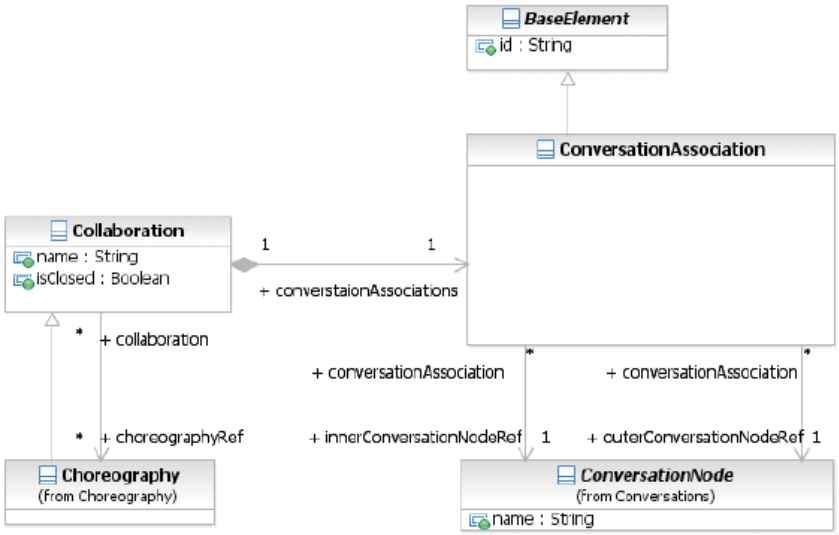
Фигура 9.30 – Ссылки Обмена Сообщениями типа Вызов

9.4.7. Ассоциация Обмена Сообщениями (Conversation Association)

`Элемент Ассоциация Обмена Сообщениями (ConversationAssociation)` используется на диаграммах **Взаимодействий** и **Хореографий** (**Choreographies**) с целью использования повторно выполняемых **Обменов Сообщениями** в **Потоках Сообщений** (**Message Flows**) вышеуказанных диаграмм.

`Элемент Ассоциация Обмена Сообщениями` используется тогда, когда на диаграммах имеются ссылки на **Обмен Сообщениями** для предоставления коррелятивной информации из **Сообщения** и/или для логической группировки **Потоков Сообщений**. Таким образом, данный элемент используется в ситуации, когда:

- **Взаимодействие** ссылается на **Хореографию** для включения её между **Пулами** (*Участниками*) данного **Взаимодействия**. `Узлы Обмена Сообщениями (ConversationNodes)`, принадлежащие **Хореографии** (внутренняя диаграмма), должны соответствовать `Узлам Обмена Сообщениями (ConversationNodes)` **Взаимодействия** (внешняя диаграмма).



Фигура 9.31 – Диаграмма классов элемента ConversationAssociation

Элемент `ConversationAssociation` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 9.14 содержит информацию о дополнительных ассоциациях элемента `ConversationAssociation`.

Таблица 9.14 – Ассоциации элемента ConversationAssociation

Название атрибута	Описание/использование
innerConversationNodeRef: ConversationNode [0..1]	Посредством данного атрибута указываются `Узлы Обмена Сообщениями` элемента (например, **Хореографии**, используемой во **Взаимодействии**), который соответствует родительскому элементу (например, **Взаимодействию**).
outerConversationNodeRef: ConversationNode [0..*]	Посредством данного атрибута указываются `Узлы Обмена Сообщениями` родительского элемента (например, **Взаимодействия**, ссылающегося на **Хореографию**), который соотносится с соответствующим элементом (например, **Хореографией**).

9.4.8. Корреляция (Correlations)

Корреляцией называется механизм назначения **Сообщений** на соответствующие *экземпляры* **Процесса**. Указывается для **Потоков Сообщений** (**Message Flows**), принадлежащих **Обмену Сообщениями**. Корреляции используются для указания **Обменов Сообщениями**, расположенными между **Процессами**. Такие **Обмены Сообщениями** являются достаточно простыми образцами **Обмена Сообщениями**, а значит:

- Данные, содержащиеся в **Обмене Сообщениями**, хорошо известны и определены посредством участвующих **Процессов**, что, однако, не делает расположенные ниже системы типов идентичными. Достаточно того, что идея данных известна на (потенциально очень высоком) бизнес-уровне.
- **Обмен Сообщениями** отображается в виде простого обмена **Сообщениями** между **Процессами**, другие варианты рассматриваться НЕ ДОЛЖНЫ.
- Данные **Обмена Сообщениями** принимаются **Задачами** отправки и получения (к примеру, Заказ, отправленный **Задачей Процесса**, должен быть получен по-меньшей мере одной **Задачей** участвующего **Процесса**).

- Корреляция как таковая указывается в полях корреляции, которые подразумевают подмножество данных, необходимых для использования в корреляции (например, если такие данные содержат заказ, то поле корреляции должно быть отмечено ID (идентификатором) заказа).

В некоторых приложениях удобнее использовать большее количество **Сообщений**, передаваемых *Участниками* (*Participants*) выполняемого Взаимодействия, чем то, что содержится в модели **Взаимодействия**. Это позволяет *Участникам* обмениваться другими **Сообщениями** по мере необходимости, и при этом не придется изменять само **Взаимодействие**. Если значение атрибута `isClosed` **Взаимодействия** равно «false» (либо данное значение вообще не установлено), *Участники* МОГУТ отсылать друг другу **Сообщения** без использования в данном **Взаимодействии** дополнительных **Потоков Сообщений** (**Message Flows**). Если же значение атрибута `isClosed` **Взаимодействия** равно «true», то *Участники* НЕ МОГУТ отсылать друг другу **Сообщения** без использования в данном **Взаимодействии** дополнительных **Потоков Сообщений.** Если в состав **Взаимодействия** входит **Хореография** (**Choreography**), то значение атрибута `isClosed` ДОЛЖНО БЫТЬ таким же, как во **Взаимодействии.** Ограничение немоделирумого обмена **Сообщениями**, указанное посредством атрибута `isClosed`, применимо лишь во **Взаимодействии**, содержащем ограничение. Элементы `PartnerEntities (Партнерские Сущности)` и `PartnerRoles (Партнерские Роли)`, принадлежащие *Участникам*, МОГУТ отсылать друг другу **Сообщения** в рамках других **Хореографий** (**Choreographies**), **Взаимодействий** и **Обменов Сообщениями**.

9.5. Процесс в составе Взаимодействия (Process within Collaboration)

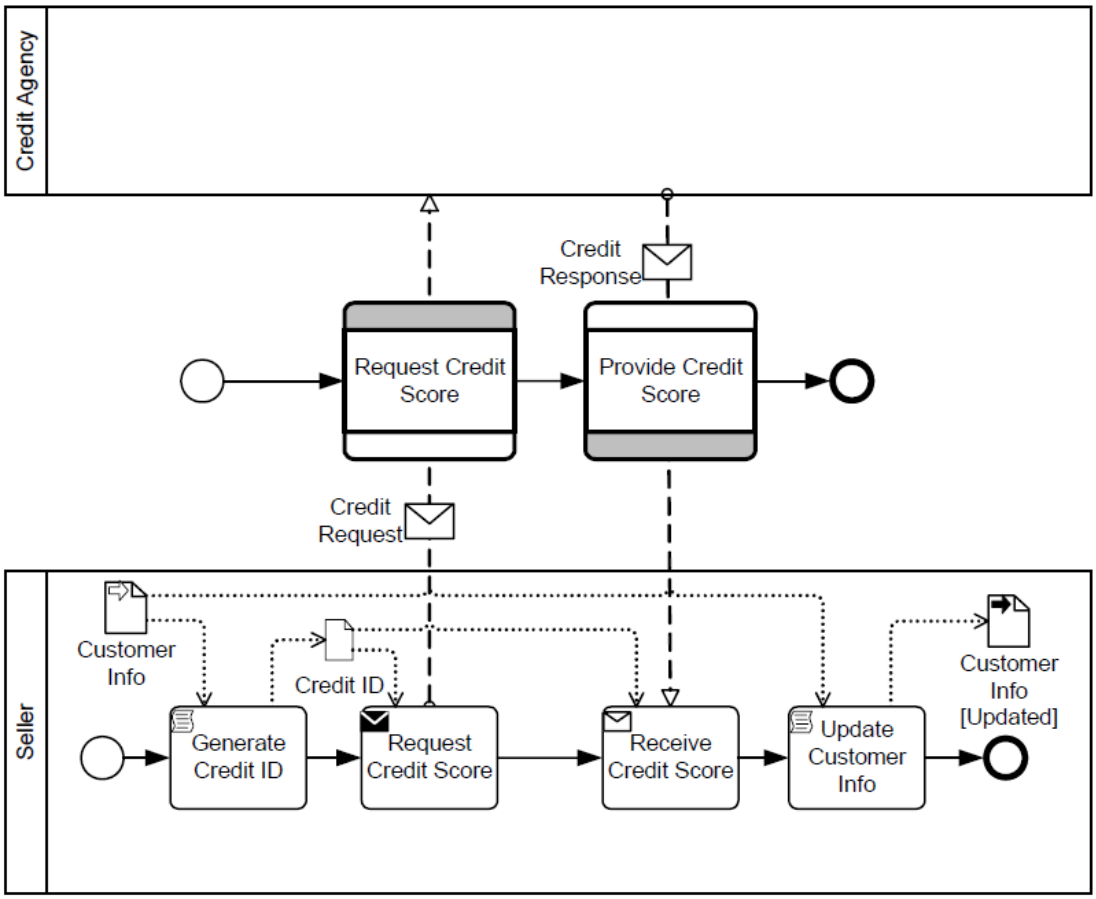
Процессы могут входить в состав диаграммы **Взаимодействия**. *Участник*/**Пул**, содержащийся во **Взаимодействии**, может заключать в себе **Процесс** (хотя это НЕ ОБЯЗАТЕЛЬНО). В качестве примера см. фигуру 9.4.

Если **Дорожка** (**Lane**) **Процесса** представляет собой **Обмен Сообщениями**, то *Элементы Потока* (*Flow Elements*), содержащиеся в ней и отправляющие или отсылающие **Сообщения**, ДОЛЖНЫ выполнять вышеперечисленные действия в качестве **Обмена Сообщениями**, для отображения которого служит данная **Дорожка**.

9.6. Хореография в составе Взаимодействия (Choreography within Collaboration)

Хореографии (**Choreographies**) могут входить в состав диаграммы **Взаимодействия**. Во **Взаимодействии** указывается то, каким образом устанавливается соответствие между *Участниками* (*Participants*) и **Потоками Сообщений** (**Message Flows**) **Хореографии** и *Участниками* и **Потоками Сообщений Взаимодействия**. Для этой цели во **Взаимодействии** используются элементы `ParticipantAssociations` и `MessageFlowAssociations`.

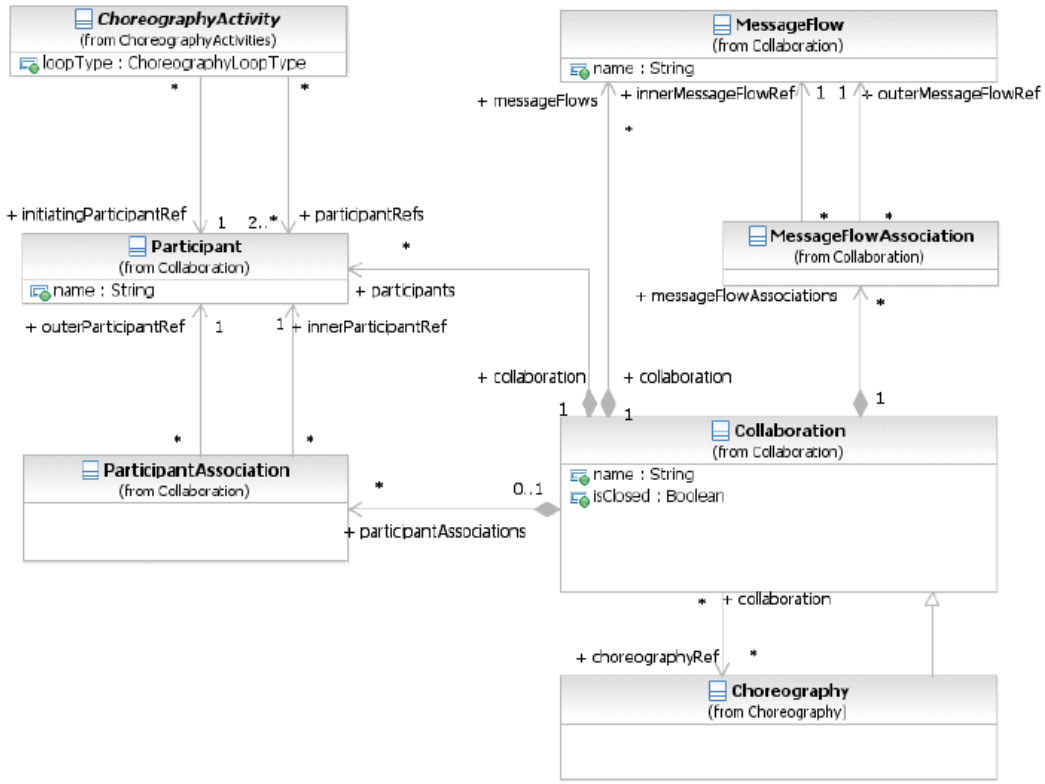
Для установки соответствий между *Участниками* элементы `innerParticipant` и `ParticipantAssociation` ссылаются на *Участника* **Хореографии**, в то время как элемент `outerParticipant` ссылается на *Участника* **Взаимодействия**, содержащего данную **Хореографию**. Такое соответствие устанавливается и между **Секциями Участников** (**Participant Bands**) **Задач Хореографии** (**Choreography Activities**) данной **Хореографии** и **Пулами** (**Pools**) **Взаимодействия**. Таким образом, использование имен в **Секциях Участников** становится НЕОБЯЗАТЕЛЬНЫМ (см. фигуру 9.32).



Фигура 9.32 – Пример Хореографии в составе Взаимодействия

Для установки соответствий между **Потоками Сообщений** элементы `innerMessageFlow` и `MessageFlowAssociation` ссылаются на **Потоки Сообщений Хореографии**, в то время как элемент `outerMessageFlow` ссылается на **Поток Сообщений Взаимодействия**, содержащего данную **Хореографию**. Такое соответствие устанавливается между **Потоками Сообщений Хореографии** (которые не отображаются) и **Потоками Сообщений Взаимодействия** (которые отображаются). Это позволяет **Потокам Сообщений Взаимодействия** «подключаться» с помощью соответствующего **Действия Хореографии** (см. фигуру 9.32).

`Ассоциации Участников (ParticipantAssociations)` могут образовываться исходя из `Партнерских Сущностей (partnerEntities)` и `Партнерских Ролей (partnerRoles)` *Участников*. К примеру, если на **Действие Хореографии** назначен *Участник* с той же `Партнерской Сущностью`, что и *Участник* **Взаимодействия**, содержащего это **Хореографию**, то оба этих *Участника* могут стать `внутренними (inner)` и `внешними` Участниками `(outerParticipants)` `Ассоциации Участников (ParticipantAssociation)`. Подобным образом, **Потоки Сообщений**, ссылающиеся на одно и то же **Сообщение Действия Хореографии** типа **Вызов** и **Взаимодействия**, могут быть автоматически синхронизированы с помощью `Ассоциации Потока Сообщений (MessageFlowAssociation)`, но только при условии, что данное **Сообщение** входит в состав только одного **Потока Сообщений**.



Фигура 9.33 – Хореография в составе диаграммы классов Взаимодействия

9.7. Представление XML-схемы для пакета Collaboration

Таблица 9.15 – XML-схема для элемента Call Conversation

```
<xsd:element name="callConversation" type="tCallConversation" substitutionGroup="conversationNode"/>
<xsd:complexType name="tCallConversation">
      <xsd:complexContent>
            <xsd:extension base="tConversationNode">
                  <xsd:sequence>
                        <xsd:element ref="participantAssociation" minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="calledCollaborationRef" type="xsd:QName" use="optional"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.16 – XML-схема для элемента Collaboration

```
<xsd:element name="collaboration" type="tCollaboration" substitutionGroup="rootElement"/>
<xsd:complexType name="tCollaboration">
      <xsd:complexContent>
            <xsd:extension base="tRootElement">
                  <xsd:sequence>
                        <xsd:element name="choreography" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="participant" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="messageFlow" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="artifact" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="conversationNode" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element name="conversationLink" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element name="conversationAssociation" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="participantAssociation" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element name="MessageFlowAssociation" type="tMessageFlowAssociation" minOccurs="0" maxOc-
                              curs="unbounded"/>
                        <xsd:element ref="correlationKey" minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string"/>
```

```
            <xsd:attribute name="isClosed" type="xsd:boolean" default="false"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.17 – XML-схема для элемента Conversation

```
<xsd:element name="conversation" type="tConversation" substitutionGroup="conversationNode"/>
<xsd:complexType name="tConversation">
    <xsd:complexContent>
        <xsd:extension base="tConversationNode"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.18 – XML-схема для элемента ConversationAssociation

```
<xsd:element name="conversationAssociation" type="tConversationAssociation"/>
<xsd:complexType name="tConversationAssociation">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="innerConversationNodeRef" type="xsd:QName" use="required"/>
            <xsd:attribute name="outerConversationNodeRef" type="xsd:QName" use="required"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.19 – XML-схема для элемента ConversationAssociation

```
<xsd:element name="conversationLink" type="tConversationLink"/>
<xsd:complexType name="tConversationLink">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="name" type="xsd:string" use="optional"/>
            <xsd:attribute name="sourceRef" type="xsd:QName" use="required"/>
            <xsd:attribute name="targetRef" type="xsd:QName" use="required"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.20 – XML-схема для элемента ConversationNode

```
<xsd:element name="conversation" type="tConversation" substitutionGroup="rootElement"/>
<xsd:complexType name="tConversation">
    <xsd:complexContent>
        <xsd:extension base="tCallableElement">
            <xsd:sequence>
        <xsd:element ref="conversationNode" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="participant" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="artifact" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="messageFlow" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="messageFlowRef" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="correlationKey" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.21 – XML-схема для элемента Conversation Node

```
<xsd:element name="conversationNode" type="tConversationNode"/>
<xsd:complexType name="tConversationNode" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="messageFlowRef" type="xsd:QName" minOccurs="0" maxOc-
                        curs="unbounded"/>
                <xsd:element name="participantRef" type="xsd:QName" minOccurs="0" maxOc-
                        curs="unbounded"/>
            </xsd:sequence>
```

```
                <xsd:attribute name="conversationRef" type="xsd:QName"/>
                <xsd:attribute name="correlationKeyRef" type="xsd:QName"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.22 – XML-схема для элемента Global Conversation

```
<xsd:element name="globalConversation" type="tGlobalConversation" substitutionGroup="collaboration"/>
<xsd:complexType name="tGlobalConversation">
        <xsd:complexContent>
            <xsd:extension base="tCollaboration"/>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.23 – XML-схема для элемента MessageFlow

```
<xsd:element name="messageFlow" type="tMessageFlow"/>
<xsd:complexType name="tMessageFlow">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:attribute name="name" type="xsd:string" use="optional"/>
                <xsd:attribute name="sourceRef" type="xsd:QName" use="required"/>
                <xsd:attribute name="targetRef" type="xsd:QName" use="required"/>
                <xsd:attribute name="messageRef" type="xsd:QName"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.24 – XML-схема для элемента MessageFlowAssociation

```
<xsd:element name="messageFlowAssociation" type="tMessageFlowAssociation"/>
<xsd:complexType name="tMessageFlowAssociation">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:attribute name="innerMessageFlowRef" type="xsd:QName" use="required"/>
                 <xsd:attribute name="outerMessageFlowRef" type="xsd:QName" use="required"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.25 – XML-схема для элемента Participant

```
<xsd:element name="participant" type="tParticipant"/>
<xsd:complexType name="tParticipant">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:sequence>
                    <xsd:element name="interfaceRef" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/
                                >
                    <xsd:element name="endPointRef" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/
                                >
                    <xsd:element ref="participantMultiplicity" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string"/>
                <xsd:attribute name="processRef" type="xsd:QName" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.26 – XML-схема для элемента ParticipantAssociation

```
<xsd:element name="participantAssociation" type="tParticipantAssociation"/>
<xsd:complexType name="tParticipantAssociation">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:sequence>
                    <xsd:element name="innerParticipantRef" type="xsd:QName" use="required"/>
```

```
                    <xsd:element name="outerParticipantRef" type="xsd:QName" use="required"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.27 – XML-схема для элемента ParticipantMultiplicity

```
<xsd:element name="participantMultiplicity" type="tParticipantMultiplicity"/>
<xsd:complexType name="tParticipantMultiplicity">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:attribute name="minimum" type="xsd:int"/>
                <xsd:attribute name="maximum" type="xsd:int"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.28 – XML-схема для элемента PartnerEntity

```
<xsd:element name="partnerEntity" type="tPartnerEntity" substitutionGroup="rootElement"/>
<xsd:complexType name="tPartnerEntity">
        <xsd:complexContent>
            <xsd:extension base="tRootElement">
                <xsd:attribute name="name" type="xsd:string"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 9.29 – XML-схема для элемента PartnerRole

```
<xsd:element name="partnerRole" type="tPartnerRole" substitutionGroup="rootElement"/>
<xsd:complexType name="tPartnerRole">
        <xsd:complexContent>
            <xsd:extension base="tRootElement">
                <xsd:attribute name="name" type="xsd:string"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

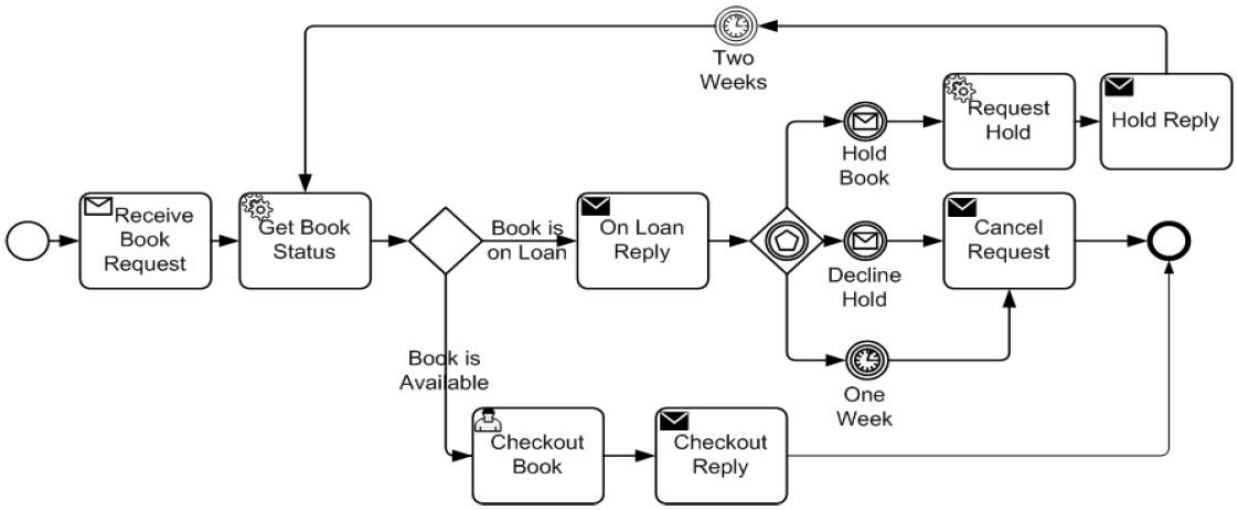
Таблица 9.30 – XML-схема для элемента Sub-Conversation

```
<xsd:element name="subConversation" type="tSubConversation" substitutionGroup="conversationNode"/>
<xsd:complexType name="tSubConversation">
        <xsd:complexContent>
            <xsd:extension base="tConversationNode">
                <xsd:sequence>
                    <xsd:element ref="conversationNode" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

10. Процесс

Примечание: описанные в данной главе рекомендации по моделированию и выполнению **Процессов BPMN** НЕОБХОДИМЫ для соответствия требованиями моделирования **Процессов BPMN** или общим стандартам **BPMN**. Однако данная глава НЕ ОБЯЗАТЕЛЬНО входит в состав документов соответствия стандартам моделирования **Хореографии Процесса BPMN**, соответствия стандартам выполнения **Процесса BPMN**, соответствия стандартам выполнения **Процесса BPEL BPMN**.

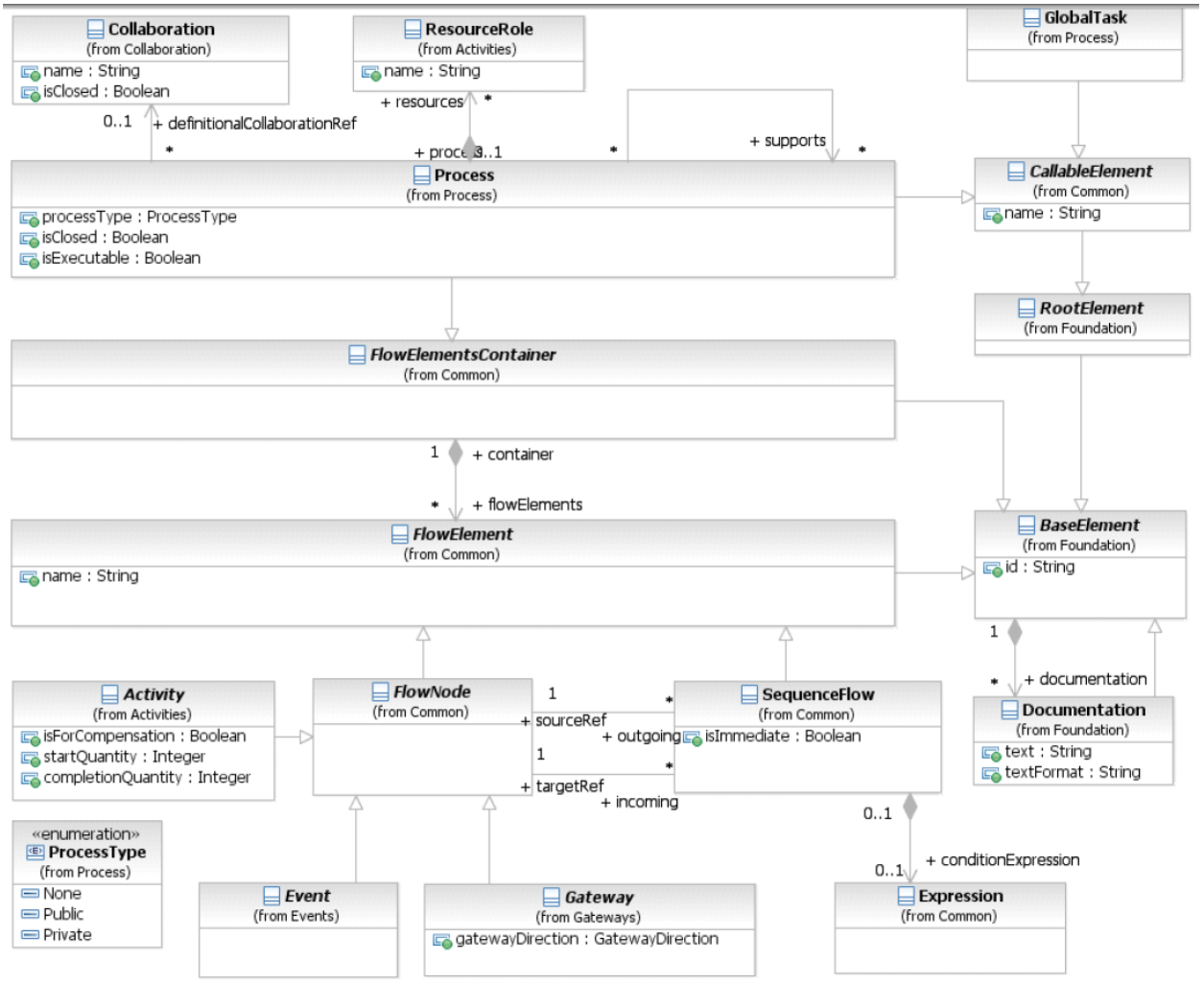
Процесс BPMN представляет собой последовательный поток **Действий** в организации, направленный на выполнение работы. Графически он изображается в виде диаграммы, содержащей такие Элементы потока, как **Действия**, **События**, **Шлюзы**, а также **Поток операций**, который и определяет конечную семантику выполнения (см. фигуру 10.1). Для описания работы организации могут использоваться **Процессы** любого уровня сложности, начиная от **Процессов** масштаба целой организации до **Процессов**, выполняемых одним человеком. Сгруппированные низкоуровневые **Процессы** служат для достижения общей бизнес-цели.



Фигура 10.1 – Пример Процесса

Обратите внимание, что термин «**Процесс**», используемый в **BPMN**, более точно передает суть набора *элементов потока*. Терминами «**Взаимодействие**» и «**Хореография**» обозначаются взаимоотношения между **Процессами**.

Пакет **Процесса** содержит классы, используемые для моделирования потока **Действий**, **Событий** и **Шлюзов**, а также служащие для описания последовательности их выполнения в ходе Процесса (см. фигуру 10.2). Если **Процессу** дано определение, оно помещается в `Definitions`.

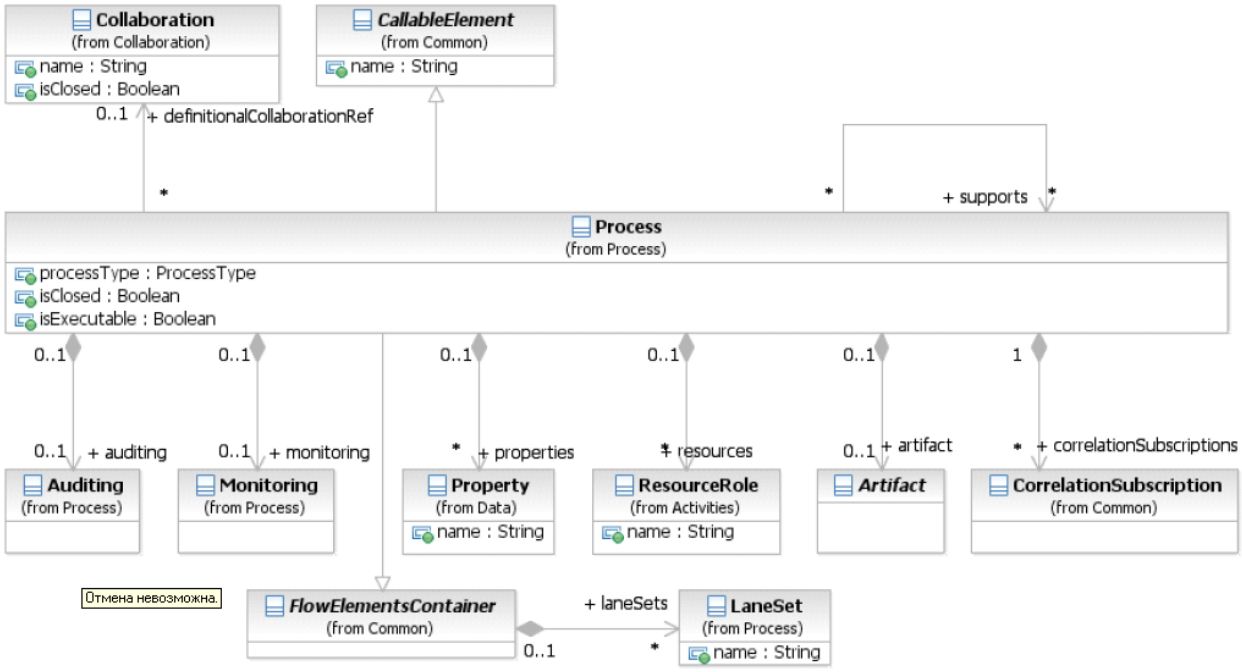


Фигура 10.2 – Диаграмма классов элемента Process

Процесс относится к классу `CallableElement`, что позволяет ему ссылаться на другие **Процессы** и быть повторно использованным ими благодаря использованию **Действия Вызов**. Процесс МОЖЕТ ссылаться на ряд элементов `Interfaces`, определяющих его внешнее поведение.

Процесс может использоваться повторно, а также импортироваться и использоваться в составе других `Definitions`.

Детали атрибутов и ассоциаций модели **Процесса** изображены на фигуре 10.3.



Фигура 10.3 – Диаграмма классов деталей Процесса

Элемент **Process** наследуют атрибуты и ассоциации элементов `CallableElement` (см. таблицу 10.24) и `FlowElementContainer` (см. таблицу 8.45). Таблица 10.1 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Process**.

Таблица 10.1 – Атрибуты и ассоциации элемента Process

Название атрибута	Описание/использование
processType: ProcessType = none { None | Private | Public }	Атрибут `processType` обеспечивает наличие дополнительной информации о том, насколько абстрактен данный **Процесс** (насколько публичным он является). Публичный **Процесс** содержит только те элементы потока, которые являются важными для внешнего заказчика. Содержимое **Процесса** не моделируется. Такие **Процессы** доступны для ознакомления и могут быть использованы во **Взаимодействии**. Обратите внимание, что в **BPMN 1.2** публичный процесс назван абстрактным. Приватный **Процесс** является внутренним для данной организации. По умолчанию значение данного атрибута равно «none».
isExecutable: boolean [0..1]	Данный атрибут является опциональным и указывает, является ли **Процесс** выполняемый. Выполняемым называется приватный **Процесс**, моделируемый для выполнения в соответствии с семантическими правилами, изложенными в главе 14. Разумеется, в ходе разработки такого **Процесса** невозможно исключить отрезки, в

	которых не окажется достаточно информации для того, чтобы **Процесс** мог являться выполняемым. Невыполняемый **Процесс** - это приватный **Процесс**, моделируемый для подробного или не очень (решает разработчик модели) документирования поведения **Процесса**. Таким образом, информация, необходимая для выполнения **Процесса** (например, выражения формальных условий), как правило, не включается в невыполняемые **Процессы**. В публичных **Процессах** значения атрибутов не имеют такой же семантики, как если бы они уже имели значения *«false»*. Это означает, что в данном случае значение МОЖЕТ БЫТЬ не *«true»*.
auditing: Auditing [0..1]	Данный атрибут определяет метод проверки схожих свойств.
monitoring: Monitoring [0..1]	Данный атрибут определяет метод отслеживания схожих свойств.
artifacts: Artifact [0..*]	Данный атрибут предоставляет список **Артефактов**, содержащихся в **Процессе**.
IsClosed: boolean = false	Данный атрибут указывает, будут ли взаимоотношения (такие, как отправка и получение **Сообщений** и **Событий**), не запланированные в **Процессе**, осуществляться в точке завершения/выполнения данного **Процесса**. Если выбрано значение *«true»*, то взаимоотношения МОГУТ и НЕ осуществиться. Если выбрано значение *«false»*, то взаимоотношения МОГУТ быть.
supports: Process [0..*]	Посредством данного атрибута разработчики модели могут показать, что все действия, ведущие к выполнению или завершению одного **Процесса**, также верны и для другого **Процесса**. Это означает, что действия, включенные в состав первого **Процесса**, не будут отличаться от действий второго **Процесса**.
properties: Property [0..*]	Данный атрибут указывает на то, что в **Процесс** МОГУТ БЫТЬ добавлены *свойства (properties)*, определенные разработчиком модели. Все **Задачи** и **Подпроцессы** ДОЛЖНЫ иметь доступ к таким *свойствам*.
resources: ResourceRole [0..*]	Данный атрибут определяет ресурсы (исполнителей), выполняющие **Процесс** или являющиеся за него ответственными. Ресурсом (например, исполнителем) может являться одно лицо, группа лиц, роль или должность в организации, а также сама организация. Обратите внимание, что ресурсы, назначенные для какого-либо **Процесса**, не обуславливают назначенных ресурсов для выполнения

	Действий, входящих в состав данного **Процесса**. Для получения более подробной информации о назначении ресурсов.
correlationSubscriptions: CorrelationSubscription [0..*]	Элементы `correlationSubscriptions` являются особенностью основанной на контексте корреляции (см. 8.3.3). Используются для установления связи между входящими **Сообщениями** и контекстом **Процесса**. В **Процессе** МОЖЕТ содержаться несколько таких атрибутов.
definitionalCollaborationRef: Collaboration [0..1]	В случае **Процесса**, взаимодействующего с другими *Участниками*, область определения понятия **Взаимодействие** зависит от данного **Процесса**. **Взаимодействие** здесь подразумевает *Участников*, с которыми взаимодействует **Процесс**, а именно – что должен выполнять (отправка или получение **Задач, Сообщение**) тот или иной *Участник*, что определяется посредством **Потоков сообщений**. Нет необходимости отображать область определения **Взаимодействия** на диаграмме. Также **Взаимодействие** может подразумевать добавление в **Процесс** информации об **Обмене сообщениями.**

Также необходимо указать, что у экземпляра имеются атрибуты, на значения которых МОГУТ ссылаться *выражения (Expressions)*, см. таблицу 10.2. Эти **Процесса** значения могут быть доступны только в случае, если **Процесс** находится на стадии выполнения.

Таблица 10.2 – Атрибуты экземпляра Процесса

Название атрибута	Описание/использование
state: string = None	Для ознакомления с допустимыми значениями см. фигуру 13.2 (Жизненный цикл Действия BPMN) в подразделе 13.2.2.

10.1. Основные понятия Процесса

10.1.1. Типы процессов BPMN

Моделирование **Бизнес-процессов** предназначено для передачи самой разнообразной информации широкой аудитории.
BPMN охватывает множество типов моделирования и позволяет создавать **Бизнес-процессы** полного цикла (end-to-end). Существует три основных типа **процессов BPMN**:

1. *Приватный невыполняемый (внутренний)* **Бизнес-процесс**.
2. *Приватный выполняемый (внутренний)* **Бизнес-процесс**.
3. *Публичный* **Процесс**.

10.1.1.1. Приватный (Внутренний) Бизнес-процесс.

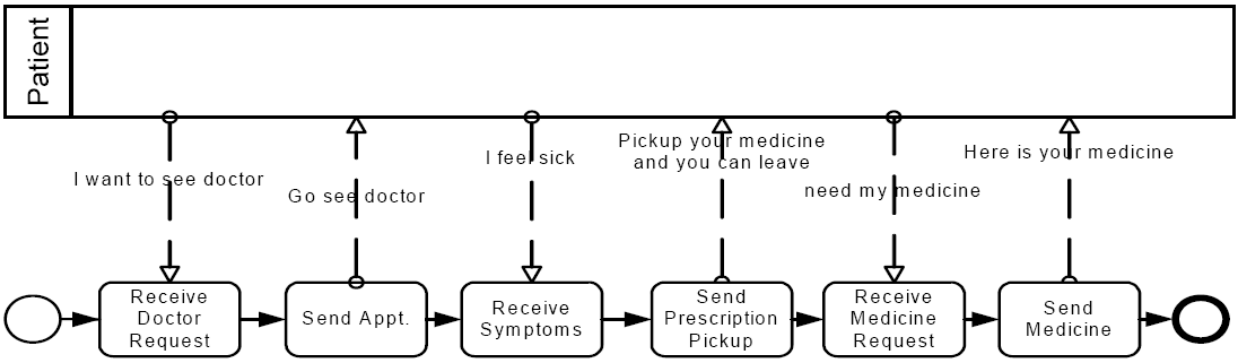
Приватный **Бизнес-процесс** относится к *внутренним* **Процессам** со специфической структурой. Такие **Процессы**, как правило, называются потоками работ или **Процессами BPM** (см. фигуру 10.4). Другое название этого **Процесса** обычно используется для веб-сервисов и звучит как *Оркестровка* сервисов (*Orchestration* of services). Существует два типа *приватных* **Процессов**: *выполняемые* и *невыполняемые*. *Выполняемым* называется **Процесс**, смоделированный для выполнения согласно семантике, описанной в Главе 14. Разумеется, в ходе разработки такого **Процесса** невозможно исключить отрезки, в которых не окажется достаточно информации для того, чтобы **Процесс** мог являться *выполняемым*. *Невыполняемый* **Процесс** это *приватный* **Процесс**, моделируемый для подробного или не очень (решает разработчик модели) документирования поведения **Процесса**. Таким образом, информация, необходимая для выполнения **Процесса** (например, *выражения формальных условий*), как правило, не включается в *невыполняемые* **процессы**.

Если при моделировании **Процесса** используется нотация Зон ответственности (swimlanes-like notation) (например, **Взаимодействие**, см. ниже), то *приватный* **Бизнес-процесс** находится в отдельно взятом **Пуле**. Поток элементов данного **Процесса** также находится в **Пуле** и не может выходить за его границы. Поток **Сообщений**, в свою очередь, может пересекать границы **Пула** для того, чтобы изобразить взаимоотношения между отдельно взятыми *приватными* **Бизнес-процессами**.

Фигура 10.4 – Пример приватного Бизнес-процесса.

10.1.1.2. Публичный Процесс

Публичный **Процесс** подразумевает взаимодействие между *приватным* **Бизнес-процессом** и каким-либо другим **Процессом** или *Участником* (см. фигуру 10.5). *Публичный* **Процесс** содержит лишь те **Действия** или их последовательность, которые используются для изображения взаимоотношений с другими *Участниками*. Другие внутренние **Действия** *приватного* **Бизнес-процесса** в *публичном* **Процессе** не отображаются. Таким образом, *публичный* **Процесс** используется для передачи внешнему миру **Сообщений** или их последовательности, необходимых для взаимодействия с данным **Бизнес-процессом**. *Публичный* **Процесс** может быть моделирован как отдельно от **Взаимодействия**, так и внутри него, причем последнее используется для отображения обмена **Сообщениями** между **Действиями** *публичного* **Процесса** и другими *Участниками*. Обратите внимание, что в **BPMN 1.2** *публичный* **Процесс** назван абстрактным.



Фигура 10.5 – Пример публичного Процесса

10.1.2. Использование общих для BPMN элементов

Некоторые графические элементы **BPMN** являются общими как для **Процесса**, так и для **Хореографии** и **Взаимодействия**. Все они могут использоваться на такого рода диаграммах. Следующие подразделы посвящены описанию использования в **Хореографии Сообщений**, **Потока сообщений**, *Участников*, `Потока операций`, *Корреляций*, `Выражений`, а также *Сервисов*.

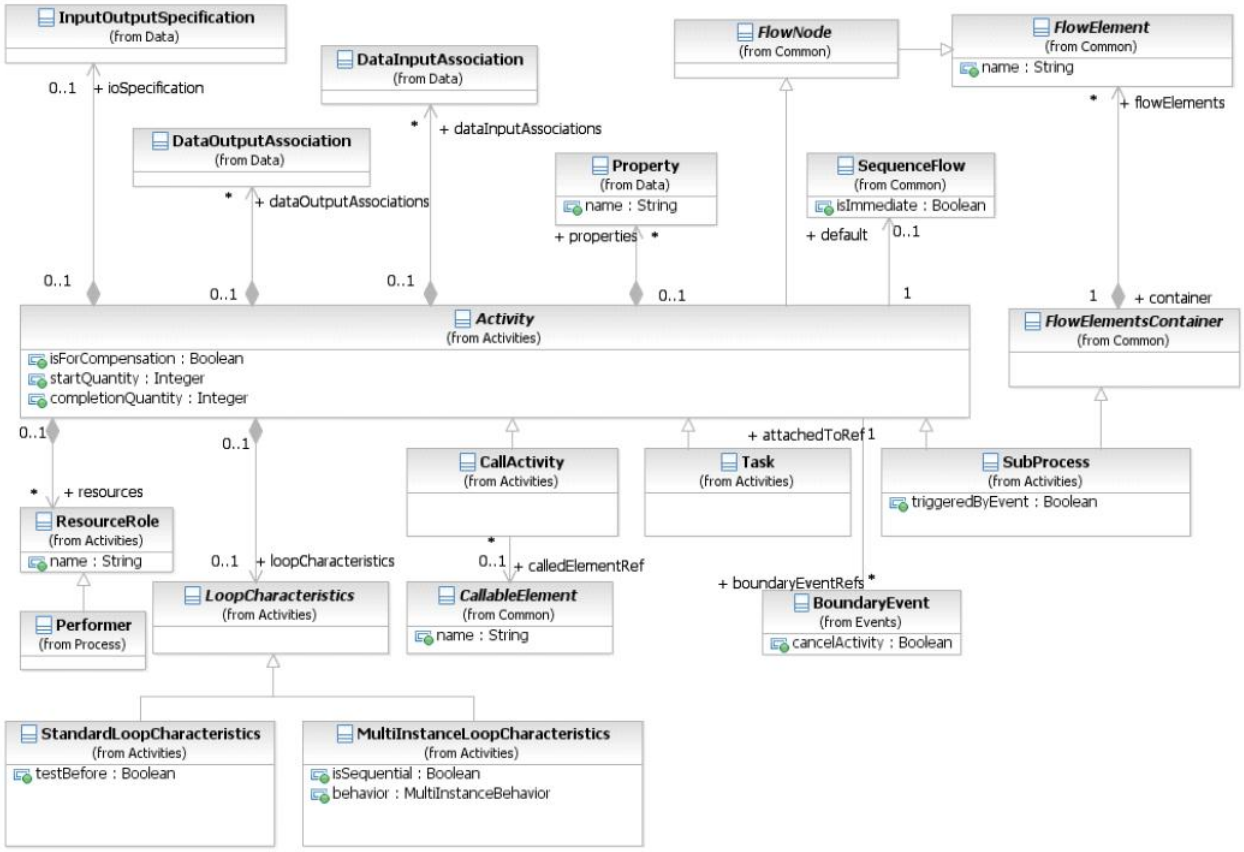
Ключевые графические элементы **Шлюз** и **Событие** также могут использовать как в **Процессе**, так и в **Хореографии**. Поскольку упомянутые выше элементы оказывают влияние на ход **Процесса** или **Хореографии**, их описанию посвящена значительная часть данной главы (см. разделы **События** и **Шлюзы**).

10.2. Действия

Действие представляет собой деятельность, выполняемую внутри **Бизнес-процесса**. **Действие** может быть как элементарным, так и неэлементарным (составным). Диаграмма **Бизнес-процесса** может содержать все существующие виды действий: **Задачу**, **Подпроцесс**, а также **Вызов**, позволяющий включать в состав диаграммы повторно используемые **Задачи** и **Процессы**. Поскольку для изображения **Процесс**а используется набор графических элементов, а не один из них, следующие подразделы данного документа содержат подробное описание таких элементов, как **Подпроцесс** и **Задача**.

Действия – это точки выполнения работ в ходе **Процесса**. Они относятся к выполняемым элементам **Процесса BPMN**.

Класс **Activity** относится к абстрактным элементам и является сабклассингом элемента `FlowElement` (см. фигуру 10.6). Отдельно взятые подклассы **Действия** определяют дополнительные правила семантики, что в последующем определяет характерное для данного класса **Действие**.



Фигура 10.6 – Диаграмма классов элемента Activity

Класс **Действия** – это абстрактный суперкласс, объединяющий все отдельно взятые типы **Действия**.

Элемент **Activity** наследует атрибуты и ассоциации элемента `FlowElement` (см. таблицу 8.44). Таблица 10.3 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Activity**.

Таблица 10.3 - Атрибуты и ассоциации элемента Activity

Название атрибута	Описание/Использование
isForCompensation: boolean = false	Данный атрибут представляет собой индикатор, согласно которому определяется, будет ли данное **Действие** использовано в целях *компенсации*. Значение *«false»* означает, что выполняемое **Действие** - результат Стандартного потока операций. Значение *«true»* указывает на то, что **Действие** предполагается выполнять только в том случае, если в Потоке операций присутствует *Компенсация* и она запущена в собственной области видимости.
loopCharacteristics: LoopCharac-teristics [0..1]	**Действие** МОЖЕТ выполняться однократно или несколько раз. При повторном использовании **Действие** ДОЛЖНО иметь атрибут `loopCharacteristics`, определяющий критерии цикличности. При этом атрибут `isExecutable` **Процесса** должен иметь значение «*true*».
resources: ResourceRole [0..*]	Данный атрибут определяет ресурс, выполняющий данное **Действие** или отвечающий за его выполнение. Ресурсом (например, исполнителем) может являться одно лицо, группа лиц, роль или должность в организации, а также сама организация.
default: SequenceFlow [0..1]	Данный атрибут указывает на **Поток операций**, который получит *Токен* в случае, если ни одно из выражений `conditionExpressions` для других *Исходящих* **Потоков операций** не имеет значение *«true»*. **Поток операций** по умолчанию не должен содержать `условных Выражений`. (`conditionExpression`). Все эти выражения ДОЛЖНЫ БЫТЬ проигнорированы.
ioSpecification: Input OutputSpecification [0..1]	Данный атрибут определяет *входные* и *выходные* данные **Действия**, а также атрибуты `InputSets` и `OutputSets` для него.
properties: Property [0..*]	`Свойства` определяются разработчиком модели и по желанию МОГУТ БЫТЬ добавлены к атрибутам **Действия**. Хранятся в самом элементе **Действия**.
boundaryEventRefs: BoundaryEvent [0..*]	Данный атрибут ссылается на **Промежуточное**

	событие, присоединенное к границам **Действия**.
dataOutputAssociations: DataOutputAssociation [0..*]	Опциональная ссылка на элементы `DataInputAssociation`, определяющие то, каким образом будут заполняться входные данные элемента `InputOutputSpecification` **Действия**.
dataOutputAssociations: DataOutputAssociation [0..*]	Опциональная ссылка на элемент `DataOutputAssociations`.
startQuantity: integer = 1	По умолчанию данный атрибут имеет значение «1». Его значение НЕ МОЖЕТ быть меньше «1». Определяет количество *Токенов*, которые ДОЛЖНЫ поступить до начала выполнения **Действия**. Обратите внимание, что любое значение данного атрибута, кроме «1», подходит для моделирования продвинутым разработчиком и должно использоваться с осторожностью.
completionQuantity: integer = 1	По умолчанию данный атрибут имеет значение «1». Его значение НЕ МОЖЕТ быть меньше 1. Определяет количество *Токенов*, которые ДОЛЖНЫ БЫТЬ запущены **Действием**. Число полученных *Токенов* отсылается любому *Исходящему* **Потоку операций** (если выполнены какие-либо из условий **Потока операций**). Обратите внимание, что любое значение данного атрибута, кроме «1», подходит для моделирования продвинутым разработчиком и должно использоваться с осторожностью.

Также, и *экземпляры* **Действия** имеют свои атрибуты, на значения которых МОГУТ ссылаться `выражения`. Эти значения доступны лишь в момент выполнения **Действия**.

Таблица 10.4 содержит информацию об атрибутах *экземпляров* **Действия**.

Таблица 10.4 - Атрибуты экземпляра Действия

Название атрибута	Описание/использованию
state: string = None	Для ознакомления с допустимыми значениями см. фигуру 13.2 (Жизненный цикл Действия BPMN) в подразделе 13.2.2.

Соединение Потока операций

Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они МОГУТ служить *источниками* и *целями* **Потока операций**, обратитесь к подразделу 2.5.

- **Действие** МОЖЕТ являться целью **Потока операций** и иметь любое количество *Входящих* **Потоков операций.** *Входящие* **Потоки операций** МОГУТ исходить как от альтернативных, так и от параллельных маршрутов.
    - Если **Действие** не имеет ни одного *Входящего* **Потока операций**, такое **Действие** ДОЛЖНО отображаться тогда, когда отображается **Процесс**.
        - Для вышеприведенных правил существуют два исключения: **Событие Компенсация** и **Подпроцесс**, работающий с **Событиями**.

Примечание: Если **Действие** соединено с несколькими *Входящими* **Потоками операций**, то такое явление называется Неконтролируемым потоком операций. Этот термин означает, что **Действие** будет отображаться по прибытии *Токена* по одному из имеющихся маршрутов. Прибытие *Токенов* по другим маршрутам не будет иметь значения. Если другой *Токен* прибывает по тому же самому маршруту, тогда запускается отдельный экземпляр **Действия**. Если необходимо, чтобы контроль **Потока операций** все же осуществлялся, он должен сходиться в **Шлюзе**, предшествующем данному **Действию** (для более подробной информации о **Шлюзах** см. подраздел **Шлюзы**).

- **Действие** МОЖЕТ являться источником **Потока операций** и иметь любое количество *Исходящих* **потоков операций**. Если **Действие** имеет несколько *Исходящих* **потоков операций**, это означает, что для каждого **Потока операций** создается свой параллельный маршрут (например, для каждого **Потока операций**, *отходящего* от **Действия,** создаются свои *Токены*).
    - Если **Действие** не имеет *Исходящих* **потоков операций**, оно является конечной точкой одного или более маршрутов **Процесса**. При завершении **События** и отсутствии каких-либо других действующих параллельных маршрутов **Процесс** ДОЛЖЕН БЫТЬ закончен.
        - Для вышеприведенных правил существуют два исключения: **Событие Компенсация** и **Событийный Подпроцесс**.

Соединение Потока сообщений

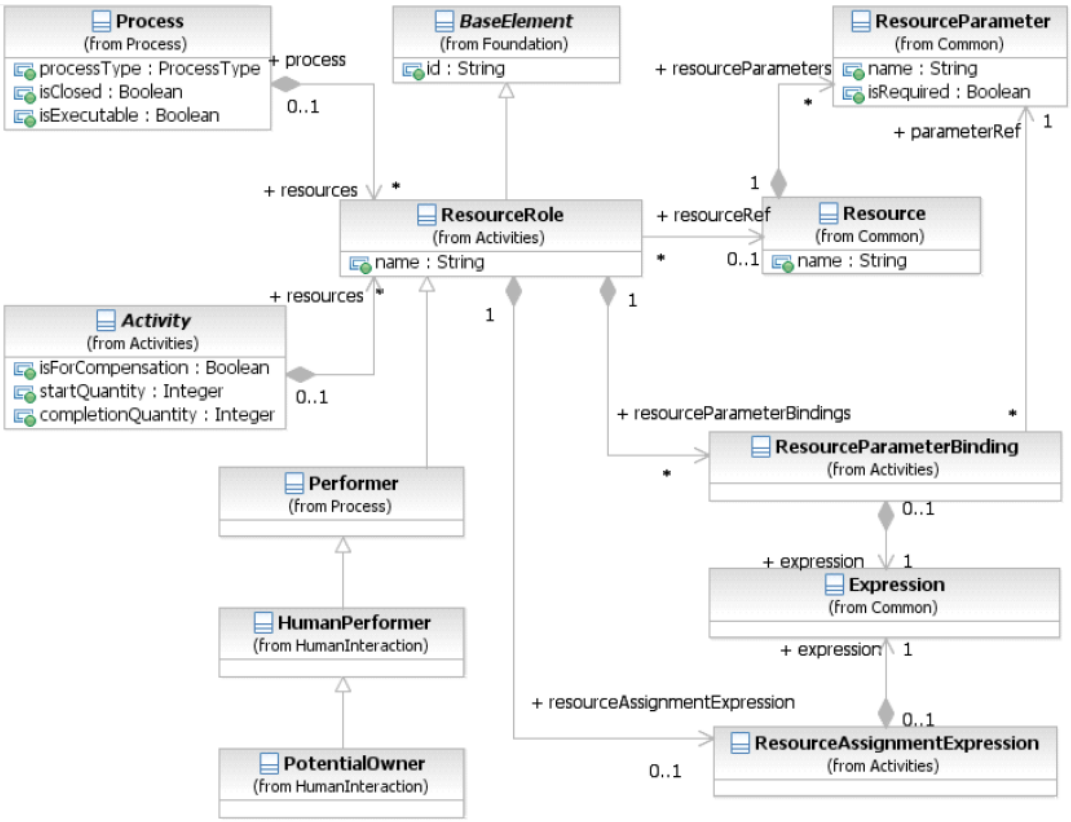
Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они МОГУТ служить *источниками* и *целями* **Потока сообщений**, обратитесь к пункту 7.5.1 «Правила Соединения Потоков Операций».

Примечание: Все **Потоки сообщений** ДОЛЖНЫ соединять два разных **Пула**. Они МОГУТ БЫТЬ присоединены как к границам **Пула**, так и к элементам потока, находящимся внутри этого **Пула**. Они НЕ ДОЛЖНЫ соединять два элементам внутри одного **Пула**.

- **Действие** МОЖЕТ являться целью **Потока сообщений** и иметь от нуля (0) и более *Входящих* **Потоков сообщений**.
- **Действие** МОЖЕТ являться источником **Потока сообщений** и иметь от нуля (0) и более *Исходящих* **Потоков сообщений**.

10.2.1. Распределение ресурсов

В следующих параграфах содержится информация о том, каким образом определяются `ресурсы` (Resources) для **Действия**. Фигура 10.7 отображает диаграмму классов элементов **BPMN**, используемых для распределения `ресурсов`.



Фигура 10.7 – Диаграмма классов для распределения ресурсов

Элемент ResourceRole

Элемент `ResourceRole` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.5 содержит информацию о дополнительных ассоциациях элемента `ResourceRole`.

Таблица 10.5 - Ассоциации элемента ResourceRole.

Название атрибута	Описание/использование
resourceRef: Resource [0..1]	Данный `ресурс` связан с **Действием** и не должен быть указан, если используется атрибут `resourceAssignmentExpression`.
resourceAssignmentExpression: Resource-AssignmentExpression [0..1]	Данный атрибут определяет выражения, используемые для распределения `ресурсов` (см. информацию ниже). Не должен быть указан, если используется атрибут `resourceRef`.
resourceParameterBindings: Resource-ParameterBinding [0..*]	Данный атрибут определяет `параметры` связующих элементов, используемых для распределения `ресурсов` (см. информацию ниже). Используется только в том случае, если определен атрибут `resourceRef`.

Expression Assignment Выражения для распределения ресурсов

`Ресурсы (Resources)` могут быть определены для **Действия**, использующего определенные `выражения`. Такие `выражения` ДОЛЖНЫ возвращать типы данных, связанные с `ресурсами`, например данные о

Пользователях или Группах. Различные выражения могут возвращать данные о множестве ресурсов. Для всех них определяются соответствующие подклассы класса ResourceRole, например, потенциальные владельцы. Семантика здесь определяется подклассом.

Элемент ResourceAssignmentExpression наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 10.6 содержит информацию о дополнительных ассоциациях элемента ResourceAssignmentExpression.

Таблица 10.6 - Ассоциации элемента ResourceAssignmentExpression

Название атрибута	Описание/использование
expression: Expression	Элемента ResourceAssignmentExpression ДОЛЖЕН содержать выражение (Expression), используемое в рабочем цикле для назначения ресурса (или ресурсов) для элемента ResourceRole.

Назначение параметризованного ресурса

Ресурсы поддерживают параметры запросов, поступающих к Ресурсу в рабочем цикле. Эти параметры МОГУТ относиться к данным об *экземплярах* **Задач**, использующим выражения. В ходе выполнения запроса инфраструктурой определяется, какой из параметров, определенный Ресурсом, используется. МОГУТ БЫТЬ использованы несколько параметров, от «0» и более. Они могут являться более значимым, чем параметры, значения которых были определены в ходе использования Ресурса. Данный документ не содержит описания механизма использования **Задачи** и Ресурса. Определение количества запросов ресурса необходимо для определения группы Ресурсов, например, людей, назначенных на выполнение **Действия**. Невыполненные запросы оцениваются как запросы, возвращающие пустой набор данных. Запросы возвращают один или несколько (группу) Ресурсов.

Элемент ResourceParameterBinding наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 10.7 содержит информацию о дополнительных ассоциациях элемента ResourceParameterBinding.

Таблица 10.7 - Ассоциации элемента ResourceParameterBinding.

Название атрибута	Описание/использование
parameterRef: ResourceParameter	Ссылается на параметр, определенный Ресурсом.
expression: Expression	Выражение, используемое для вычисления значения, используемого для связки с атрибутом типа ResourceParameter.

10.2.2. Исполнитель

Класс Performer определяет ресурс, выполняющий **Действие** или являющийся ответственным за его выполнение. Ресурсом (например, исполнителем) может являться одно лицо, группа лиц, роль или должность в организации, а также сама организация.

Элемент `Performer` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством взаимодействия с элементом `ResourceRole`. Однако у данного элемента не может быть дополнительных атрибутов или ассоциаций.

10.2.3. Задача

Задача представляет собой *элементарное* **Действие**, включенное в состав **Процесса**. Используется в случае, если **Процесс** не детализируется далее в данной Модели. Как правило, после выполнения **Задачи** её исполнителем являются конечный пользователь и/или приложения.

Задачи, как и **Процессы**, отображаются на диаграмме **Процесса** в виде прямоугольников с закругленными углами (см. фигуру 10.8).

- **Задача** представляет собой прямоугольник с закругленными углами, который ДОЛЖЕН БЫТЬ выполнен одинарной тонкой линией.
    - Текст, цвет, размер, а также линии, используемые для изображения **Задачи**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».
        - Граница данного графического элемента, выполненная жирной линией, ДОЛЖНА подразумевать использование графического элемента в качестве **Действия Вызов** (`Глобальные Задачи`).
        - Граница данного графического элемента, выполненная пунктиром, ДОЛЖНА подразумевать использование графического элемента в качестве **Подпроцесса**, работающего с **Событиями**. Такая граница неприемлема для отображения **Задач**.
        - Граница данного графического элемента, выполненная двойной линией, ДОЛЖНА подразумевать использование графического элемента в качестве **Подпроцесса Транзакции**. Такая граница неприемлема для отображения **Задач**.

Фигура 10.8 – Графический элемент «Задача»

BPMN различает три типа маркеров **Задачи**: Маркер **Цикла** (**Loop** Marker), **Многоэкземплярный** Маркер (**Multiple Instance** Marker), а также Маркер **Компенсации** (**Compensation** Marker). **Задача** может содержать от одного до двух вышеуказанных Маркеров (см. Фигуру 10.9).

- Маркер *Цикла* ДОЛЖЕН БЫТЬ выполнен в виде небольшой стрелки, острие которой загнуто в направлении, противоположном направлению самой стрелки.
    - Маркер *Цикла* МОЖЕТ сочетаться с Маркером *Компенсации*.
- *Многоэкземплярный* Маркер ДОЛЖЕН БЫТЬ выполнен в виде трех параллельных вертикальных линий. Для более подробной информации о *многоэкземплярных* **Действиях** см. подраздел 10.2.8.
    - Если *Многоэкземплярный* Маркер используется для последовательного, а не параллельного, выполнения **Задачи**, линии должны отображаться горизонтально (см. фигуру 10.49).
    - *Многоэкземплярный* Маркер МОЖЕТ сочетаться с Маркером *Компенсации*.
- Маркер *Компенсации* ДОЛЖЕН БЫТЬ выполнен в виде двух треугольников, повернутых влево (как кнопка перемотки назад на проигрывателе). Для более подробной информации о *Компенсации* см. подраздел 10.6

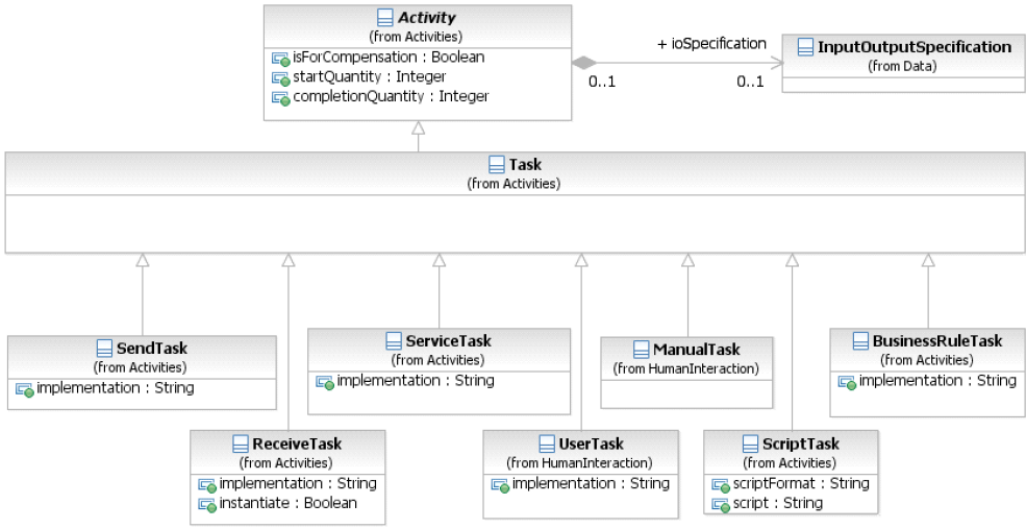
- Маркер *Компенсации* МОЖЕТ сочетаться как с Маркером *Цикла*, так и с *Многоэкземплярным* Маркером.

Все существующие Маркеры ДОЛЖНЫ БЫТЬ сгруппированы и располагаться в центре нижней части графического элемента **Задачи**.

Маркер Цикла **Многоэкземплярный Маркер** **Маркер Компенсции**

Фигура 10.9 – Маркеры Задачи

Фигура 10.10 отображает диаграмму классов элемента **Задача**



Фигура 10.10 - Диаграмма классов элемента Задача

Задача наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3), помимо которых других у **Задачи** быть не может.

10.2.3.1. Типы Задач

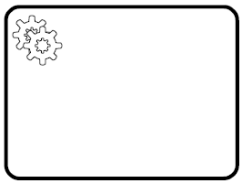
BPMN выделяет несколько типов **Задач**, что позволяет описывать различия в присущем им поведении, характерные для каждого из типов. Список типов **Задач** МОЖЕТ БЫТЬ пополнен за счет добавления различных маркеров. **Задача**, тип которой не указан, называется **Абстрактная Задача** (в **BPMN** 1.2 такая **Задача** носит название **Неопределенная Задача** (**None Task**)). Фигура 10.8 представляет собой графическое изображение **Абстрактной Задачи**.

Сервисная задача (Service Task)

Сервисная задача представляет собой **Задачу**, предназначенную для оказания услуги, которая может являться как веб-сервисом (Web service), так и автоматизированным приложением.

Графически **Сервисная Задача** отображается в виде прямоугольника с закругленными углами (установленное в **BPMN** отображение графического элемента **Задача**) и отличается от других типов **Задач** наличием определенного маркера, расположенного в левом верхнем углу фигуры **Сервисной Задачи** (см. фигуру 10.11).

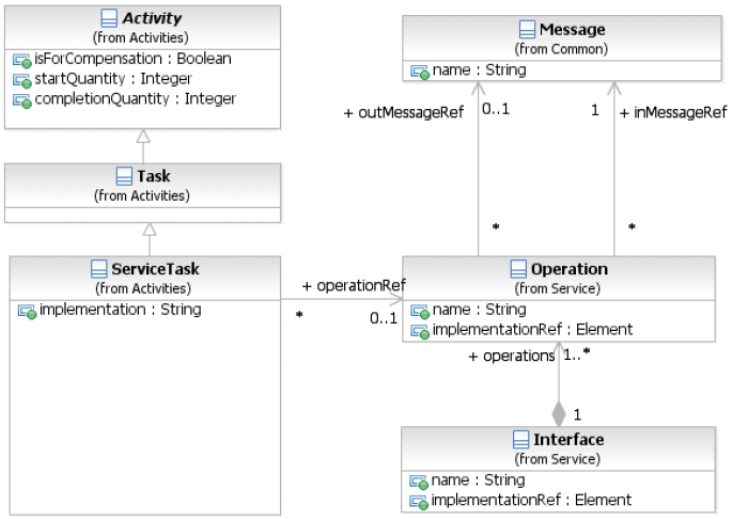
Таким образом, на диаграмме **Сервисная Задача** – это прямоугольник с закругленными углами, границы которого ДОЛЖНЫ БЫТЬ выполнены одинарной тонкой линией. Содержит маркер, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.11).



Фигура 10.11 – Графический элемент Сервисная Задача

Сервисная Задача наследует атрибуты и ассоциации **Activity** (см. таблицу 10.3). Необходимо отметить, что в случае, если **Задача** данного типа ссылается на `Операцию` (`Operation`), должны быть приняты во внимание следующие ограничения: **Сервисная Задача** имеет лишь один набор `входных` и максимум один набор `выходных` данных. Для неё подразумевается один единственный **Ввод Данных** со значением `ItemDefinition`, равным значению, определенному **Сообщением**, на которое ссылается атрибут `inMessageRef` связанной `Операции`. Если же `Операция` определяет выходные **Сообщения**, то **Сервисная Задача** будет иметь один **Вывод Данных** со значением атрибута `ItemDefinition`, равным значению, определенному **Сообщением**, на которое ссылается атрибут `outMessageRef` связанной `Операции`

Текущий *Участник*, предоставляющий услугу, указывается посредством соединения **Сервисной Задачи** и *Участника* с помощью **Потоков Сообщений**. В **Процессе** такое соединение отображается в рамках **Взаимодействия** (см. таблицу 10.1).



Фигура 10.12 – Диаграмма классов Сервисной Задачи

Элемент **ServiceTask** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3). Таблица 10.8 содержит информацию о дополнительных ассоциациях элемента **ServiceTask**.

Таблица 10.8 – Ассоциации элемента ServiceTask

Название атрибута	Описание/использование
implementation: string = ##webService	С помощью данного атрибута определяется технология, используемая для отправки и получения **Сообщений**. Значение "##unspecified" используется для указания того, что технология ещё не выбрана, значение "##WebService" – для указания технологии веб-сервисов, а введенный URL – для указания какой-либо другой технологии или протокола (coordination protocol). Технологией, определенной по умолчанию, является технология веб-сервиса.
operationRef: Operation [0..1]	Данный атрибут указывает на операцию, запускаемую **Сервисной Задачей**.

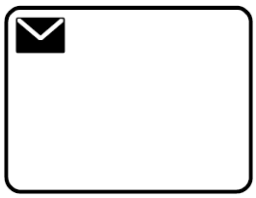
Отправка сообщений (Send Task)

Отправка сообщений представляет собой простую **Задачу**, суть которой заключается в отправлении **сообщения** внешнему `Участнику`, имеющего отношение к данному **Бизнес-процессу**. Задача считается выполненной в случае, если **Сообщение** было отправлено хотя бы один раз.

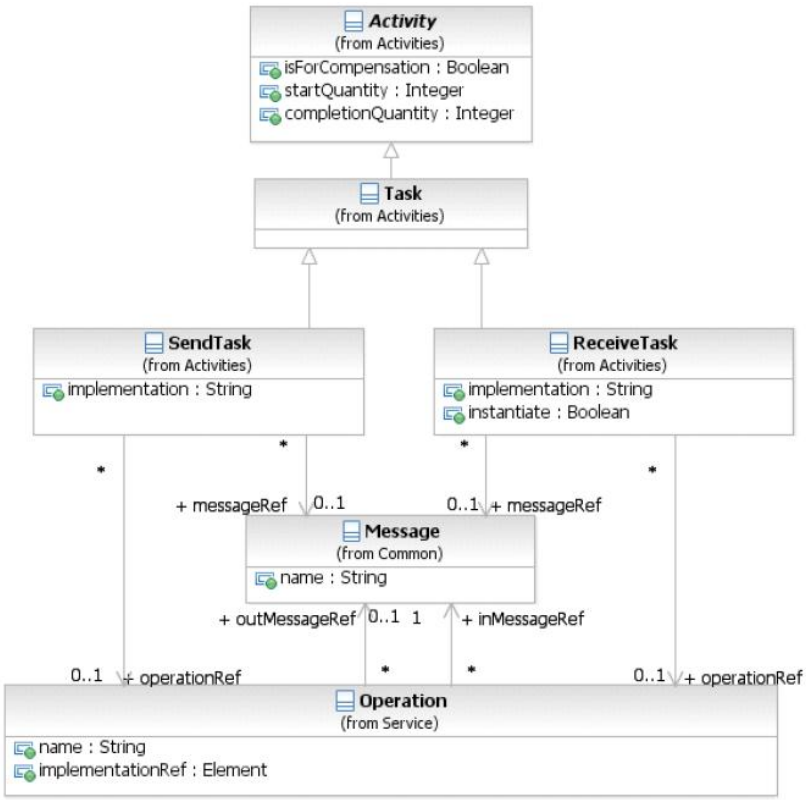
Текущий *Участник*, которому отправляется **Сообщение**, указывается посредством соединения **Отправки сообщений** и *Участника* с помощью **Потоков Сообщений**. В **Процессе** такое соединение отображается в рамках **Взаимодействия** (см. таблицу 10.1).

Графически **Отправка сообщений** отображается в виде прямоугольника с закругленными углами (установленное в **BPMN** отображение графического элемента **Задача**) и отличается от других типов **Задач** наличием маркера в виде конвертика с темной заливкой (такой же маркер имеет определяющее результат **Событие Сообщение**). Маркер расположен в левом верхнем углу фигуры **Задачи** данного типа.

Таким образом, на диаграмме **Отправка сообщений** – это прямоугольник с закругленными углами, границы которого ДОЛЖНЫ БЫТЬ выполнены одинарной тонкой линией. Содержит маркер в виде конвертика с темной заливкой, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.13).



Фигура 10.13 - Графический элемент Отправка сообщений



Фигура 10.14 – Диаграмма классов элементов SendTask и ReceiveTask

Элемент **SendTask** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3). Необходимо отметить, что в случае, если **Задача** данного типа ссылается на **Сообщение**, должны быть приняты во внимание следующие ограничения: **Отправка сообщений** имеет максимум один набор `входных` и один **Ввод Данных**. Если **Ввод Данных** осуществляется, то для данной **Задачи** ДОЛЖЕН БЫТЬ определен атрибут `ItemDefinition`, значение которого равно тому, что определено связанным **Сообщением**. Во время выполнения **Отправки сообщений** данные автоматически перемещаются из **Ввода Данных** данной **Задачи** в **Сообщение**, предназначенное для отправки. Если **Ввод Данных** не осуществляется, то данные из **Процесса** в **Сообщение** не попадают.

Таблица 10.9 содержит информацию о дополнительных ассоциациях элемента **SendTask**.

Таблица 10.9 – Ассоциации элемента SendTask

Название атрибута	Описание/использование
messageRef: Message [0..1]	Атрибут `messageRef` МОЖЕТ иметь значение «**Message**», указывающее на то, что **Сообщение** будет отправлено **Задачей**. В данном контексте **Сообщение** – это то же самое, что и `выходное` сообщение (в веб-сервисе). На диаграмме МОГУТ отображаться один или несколько связанных с **Задачей** `Исходящих` **Потоков Сообщений**. Однако в данной ситуации отображение **Потоков Сообщений** НЕ ОБЯЗАТЕЛЬНО**. Сообщение** относится ко всем *Исходящим* **Потокам Сообщений** и отсылается по всем *Исходящим* **Потокам Сообщений** по завершении отдельно взятого *экземпляра*

	Задачи данного типа.
operationRef: Operation	Данный атрибут указывает на операцию, запускаемую **Задачей Отправка сообщений**.
implementation: string = ##webService	С помощью данного атрибута определяется технология, используемая для отправки и получения **Сообщений**. Значение "##unspecified" используется для указания того, что технология ещё не выбрана, значение "##WebService" – для указания технологии веб-сервисов, а введенный URL – для указания какой-либо другой технологии или протокола (coordination protocol). Технологией, определенной по умолчанию, является технология веб-сервиса.

Получение сообщений (Receive Task)

Получение **Сообщений** представляет собой простую **Задачу**, суть которой заключается в получении **Сообщения** от внешнего *Участник*а, имеющего отношение к данному **Бизнес-Процессу**. **Задача** считается выполненной в случае, если **Сообщение** было получено хотя бы один раз.

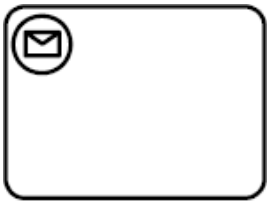
Текущий *Участник*, от которого получено **Сообщение**, указывается посредством соединения **Получения Сообщений** и *Участника* с помощью **Потоков Сообщений**. В **Процессе** такое соединение отображается в рамках **Взаимодействия** (см. таблицу 10.1).

Данный тип **Задач** обычно используется для запуска **Процесса** (получение сообщения -> самозапуск процесса). Обязательными условиями этого являются наличие у **Задачи** атрибута `instantiate`, значение которого ДОЛЖНО БЫТЬ установлено как *«true»*, и ОТСУТСТВИЕ Входящих **Потоков Операций**.

Графически **Получение Сообщений** отображается в виде прямоугольника с закругленными углами (установленное в **BPMN** отображение графического элемента **Задача**) и отличается от других типов **Задач** наличием маркера в виде конвертика без заливки (такой же маркер имеет обрабатывающее триггер **Событие Сообщение**). Маркер расположен в левом верхнем углу фигуры **Задачи** данного типа.

Таким образом, на диаграмме **Получение Сообщений** – это прямоугольник с закругленными углами, границы которого ДОЛЖНЫ БЫТЬ выполнены одинарной тонкой линией. Содержит маркер в виде конвертика без заливки, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.15). В случае, если значение атрибута `instantiate` для данной **Задачи** установлено как *«true»*, то маркер выглядит как маркер **Стартового События Сообщение** (см. фигуру 10.16).

Фигура 10.15 - Графический элемент Получение сообщений



Фигура 10.16 - Графический элемент Задачи Получение сообщений, используемой для запуска Процесса

Элемент **ReceiveTask** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3). Необходимо отметить, что в случае, если **Задача** данного типа ссылается на **Сообщение**, должны быть приняты во внимание следующие ограничения: **Получение Сообщений** имеет максимум один набор выходных и один **Вывод Данных**. Если **Вывод Данных** осуществляется, то для данной **Задачи** ДОЛЖЕН БЫТЬ определен атрибут `ItemDefinition`, значение которого равно тому, что определено связанным **Сообщением**. Во время выполнения **Получения Сообщений** данные автоматически перемещаются из **Сообщения** в **Вывод Данных**. Если **Вывод Данных** не осуществляется, то содержимое **Сообщения** не попадет из этой **Задачи** в **Процесс**.

Таблица 10.10 содержит информацию о дополнительных ассоциациях элемента **ReceiveTask**.

Таблица 10.10 – Атрибуты и ассоциации элемента ReceiveTask

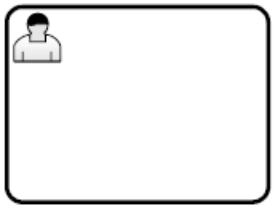
Название атрибута	Описание/использование
messageRef: Message [0..1]	Атрибут `messageRef` МОЖЕТ иметь значение «**Message**», указывающее на то, что **Сообщение** будет получено **Задачей**. В данном контексте **сообщение** – это то же самое, что и *входное* сообщение (в веб-сервисе). На диаграмме МОГУТ отображаться один (1) или несколько связанных с **Задачей Входящих Потоков Сообщений**. Однако в данной ситуации отображение **Потоков сообщений** НЕ ОБЯЗАТЕЛЬНО. **Сообщение** относится ко всем *Входящим* **Потокам Сообщений**, но приходит только по одному (1) *Входящему* **Потоку Сообщений** для каждого отдельно взятого *экземпляра* **Задачи** данного типа.
instantiate: boolean = false	Данный атрибут указывает на то, что **Получение сообщений** является механизмом, запускающим **Процесс**. Если данная **Задача** отображается как самое первое из **Действий**, включенных в **Процесс**, значение данного атрибута МОЖЕТ БЫТЬ «*true*» (например, если к **Задаче** не подходит Входящий **поток операций**). Многоэкземплярные **Задачи** также МОГУТ иметь этот атрибут со значением «*true*».
operationRef: Operation	Данный атрибут определяет операцию, с помощью которой **Задача Получение Сообщение** принимает **Сообщения**.
implementation: string = ##webService	С помощью данного атрибута определяется технология, используемая для отправки и получения **Сообщений**. Значение "##unspec-

	ified" используется для указания того, что технология ещё не выбрана, значение "##WebService" – для указания технологии веб-сервисов, а введенный URL – для указания какой-либо другой технологии или протокола (coordination protocol). Технологией, определенной по умолчанию, является технология веб-сервиса.

Пользовательская задача (User Task)

Пользовательская Задача представляет собой **Задачу**, типичную для технологического процесса, где человек выступает в роли исполнителя и выполняет **Задачи** при содействии других людей или программного обеспечения. Для выполнения **Задачи** каждый человек назначается лицом, ответственным за список **Задач**.

Графически **Пользовательская Задача** отображается в виде прямоугольника с закругленными углами (установленное в **BPMN** отображение графического элемента **Задача**), который ДОЛЖЕН БЫТЬ выполнен одинарной тонкой линией и отличаться от других типов **Задач** наличием маркера в виде верхней части фигуры человека (см. фигуру 10.17).



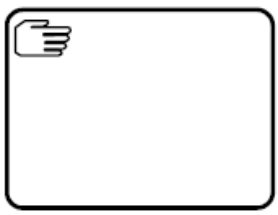
Фигура 10.17 - Графический элемент Пользовательская задача

Для получения более подробной информации о **Пользовательской Задаче** см. раздел 10.2.4, посвященный участию людей в **Бизнес-Процессе**.

Ручное выполнение (Manual Task)

Ручное выполнение представляет собой **Задачу**, выполнение которой подразумевает действия человека и исключает использование каких-либо автоматизированных механизмов исполнения или приложений. Примером такого типа **Задач** может служить установка телефонного аппарата на территории заказчика специалистом по облуживанию телефонов.

Графически **Ручное Выполнение** отображается как прямоугольник с закругленными углами (установленное в **BPMN** отображение графического элемента **Задача**), границы которого ДОЛЖНЫ БЫТЬ выполнены одинарной тонкой линией. Содержит маркер в виде руки, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.18).



Фигура 10.18 - Графический элемент Ручное выполнение

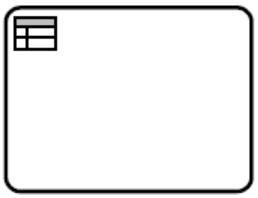
Для получения более подробной информации о **Ручном Выполнении** см. раздел 10.2.4, посвященный участию людей в **Бизнес-Процессе**.

Бизнес-правило (Business Rule)

Бизнес-правило представляет собой инструмент, используемый в **Процессе** для обеспечения доступа к Механизму Бизнес-Правил, а также для получения на выходе предоставленной этим механизмом информации об изменениях в **Бизнес-Процессе**. Использование атрибута `InputOutputSpecification` позволяет отправлять данные о **Процессе** в Механизм Бизнес-Правил и получать обратно другие данные.

Графически **Бизнес-правило** отображается как прямоугольник с закругленными углами (установленное в **BPMN** отображение графического элемента **Задача**). Содержит маркер, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.11). Маркер расположен в левом верхнем углу фигуры **Задачи** данного типа.

Таким образом, на диаграмме **Бизнес-правило** – это прямоугольник с закругленными углами, границы которого ДОЛЖНЫ БЫТЬ выполнены одинарной тонкой линией. Содержит маркер, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.19).



Фигура 10.19 - Графический элемент Бизнес-правило

Элемент **BusinessRuleTask** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3). Таблица 10.11 содержит информацию о дополнительных атрибутах элемента **BusinessRuleTask**.

Таблица 10.11 – Атрибуты и ассоциации элемента BusinessRuleTask

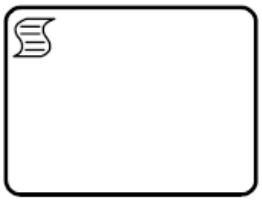
Название атрибута	Описание/использование
implementation: string = ##unspecified	С помощью данного атрибута определяется технология, используемая для выполнения **Задачи Бизнес-правило**. Значение "##unspecified" используется для указания того, что технология ещё не выбрана, значение "##WebService" – для указания технологии веб-сервисов, а введенный URL – для указания какой-либо другой технологии или протокола (coordination protocol). Технология по умолчанию не определена.

Задача-сценарий (Script Task)

Задача-Сценарий выполняется механизмом исполнения **Бизнес-Процесса**. Разработчик модели **Процесса** или её «внедренец» создают сценарий, который распознается механизмом исполнения. В нужный момент, когда **Задача** должна быть выполнена, запускается сценарий. **Задача** считается выполненной в случае, если был выполнен запущенный для неё сценарий.

Графически **Задача-сценарий** отображается как прямоугольник с закругленными углами (установленное в **BPMN** отображение графического элемента **Задача)**. Содержит маркер, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.11). Маркер расположен в левом верхнем углу фигуры **Задачи** данного типа.

Таким образом, на диаграмме **Задача-сценарий** – это прямоугольник с закругленными углами, границы которого ДОЛЖНЫ БЫТЬ выполнены одинарной тонкой линией. Содержит маркер, позволяющий отличать данный тип **Задач** от других (см. фигуру 10.20).



Фигура 10.20 - Графический элемент Задача-сценарий

Элемент **ScriptTask** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3). Таблица 10.12 содержит информацию о дополнительных атрибутах для элемента **ScriptTask**.

Таблица 10.12 – Атрибуты элемента ScriptTask

Название атрибута	Описание/использование
scriptFormat: string [0..1]	Данный атрибут определяет формат сценария. Его значение ДОЛЖНО БЫТЬ указано в формате MIME-тип. Указывать значение для данного атрибута необходимо, если сценарий для **Задачи** уже существует.
script: string [0..1]	Разработчик модели МОЖЕТ добавлять сценарий, когда **Задача** выполняется. Если сценарий не задан, то данный тип **Задач** действует в процессе по типу **Абстрактной Задачи**.

10.2.4. Участие людей

10.2.4.1. Задачи, требующие участия людей

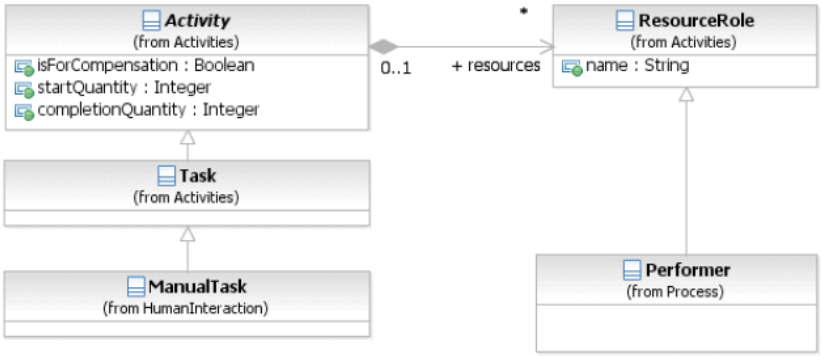
Во многих **Бизнес-Процессах** для выполнения определенных **Задач** предусмотрено участие людей. Это отображается графически на диаграммах **Бизнес-Процессов**. **BPMN** выделяет два типа **Задач**, требующих участия людей: **Ручное Выполнение** и **Пользовательская Задача**.

Пользовательская Задача выполняется и управляется в ходе выполнения **Процесса**. Атрибуты, касающиеся участия человека (например, назначения людей или визуализации информации посредством пользовательского интерфейса), могут быть очень подробными. В отличие от **Пользовательской Задачи**, **Ручное Выполнение** не выполняется и не управляется в ходе выполнения **Процесса**.

Графическое изображение.
Как **Пользовательская Задача**, так и **Ручное выполнение** отображаются на диаграмме **Бизнес-Процесса** одинаково: в виде прямоугольника с закругленными углами. Различие в отображении этих двух типов **Задач** состоит в использовании разных маркеров, подчеркивающих НЕОБХОДИМОСТЬ участия человека в выполнении **Задачи** (см. фигуры 10.17 и 10.18).

Ручное Выполнение.
Ручное Выполнение не поддается управлению никаким механизмом выполнения **Бизнес-Процесса**. Такой тип **Задач** можно отнести к неуправляемым, т.е. к **Задачам,** начало и завершение выполнения которых не

отслеживается механизмами выполнения **Бизнес-Процесса**. Примером может являться бумажная инструкция для обслуживающего телефонные аппараты специалиста по установке телефона на территории заказчика.

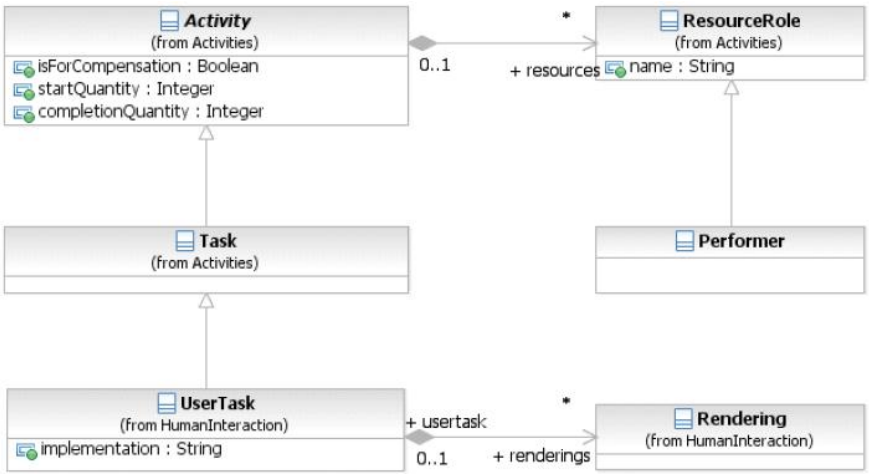


Фигура 10.21 – Диаграмма классов элемента ManualTask

Элемент **ManualTask** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3), однако, не может иметь другие дополнительные атрибуты или ассоциации.

Пользовательская Задача (User Task)

Пользовательская Задача представляет собой **Задачу**, типичную для технологического процесса, где человек выступает в роли исполнителя и выполняет **Задачи** при содействии других людей или программного обеспечения. Жизненный цикл **Пользовательской задачи** контролируется программным компонентом **бизнес-процесса** (так называемым диспетчером задач (task manager)). **Задача**, как правило, выполняется в контексте **процесса**.



Фигура 10.22 – Диаграмма классов элемента UserTask

Пользовательская Задача может быть выполнена с применением различных технологий (помимо технологии веб-сервиса), определенных атрибутом `implementation`. К примеру, **Пользовательская Задача** может выполняться с использованием спецификации WS-HumanTask путем добавления соответствующего атрибута по адресу “http://docs.oasis-open.org/ns/bpel4people/ws-humantask/protocol/20200803”.

Элемент **UserTask** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3). Таблица 10.13 содержит информацию о дополнительных атрибутах и ассоциациях элемента **UserTask**. В случае, если выполнение задачи подразумевает атрибуты, помимо указанных в таблице (например, определение параметров для предметов или описания), такие атрибуты ДОЛЖНЫ быть заимствованы из спецификации OASIS WS-HumanTask.

Таблица 10.13 – Атрибуты и ассоциации элемента UserTask

Название атрибута	Описание/использование
implementation: string = ##unspecified	С помощью данного атрибута определяется технология, используемая для выполнения **Пользовательской задачи**. Значение "##unspecified" используется для указания того, что технология ещё не выбрана, значение "##WebService" – для указания технологии веб-сервисов, а введенный URL – для указания какой-либо другой технологии или протокола (coordination protocol). Технология по умолчанию не определена.
renderings: Rendering [0..*]	Данный атрибут можно определить как метод, позволяющий приверженцам использования **BPMN** указывать атрибуты выполнения задач, используя механизм расширения **BPMN**.

Элемент **UserTask** наследует атрибуты экземпляров **Действия** (см. таблицу 8.49). Таблица 10.14 содержит информацию о таких атрибутах элемента **UserTask**.

Таблица 10.14 – Атрибуты экземпляров для элемента UserTask

Название атрибута	Описание/использование
actualOwner: string	Возвращает данные о пользователе, взявшемся выполнить **Пользовательскую задачу** или заявившем о правах на неё и ставшим её текущим владельцем. Значение является константой, которая может представлять собой имя пользователя, его e-mail или другую информацию.
taskPriority: integer	Возвращает значение приоритета **Пользовательской задачи**.

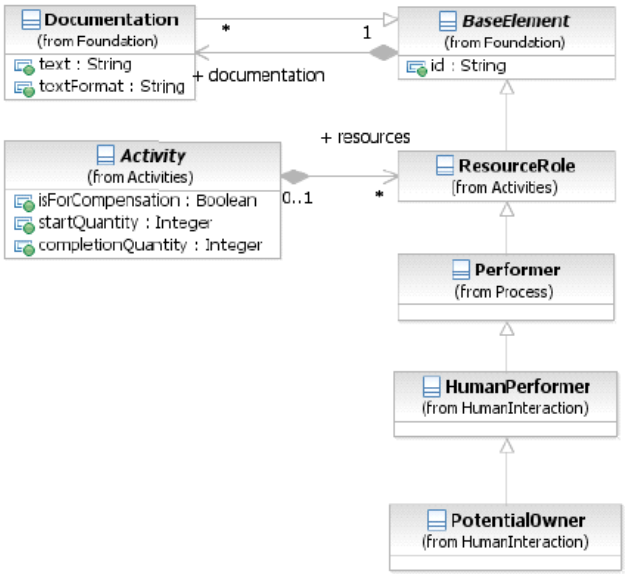
Визуализация Пользовательской задачи

Согласно **BPMN**, применение **Пользовательской задачи** подразумевает её визуализацию через пользовательские интерфейсы (клиентские формы, портлеты и т.д.). Элемент визуализации обеспечивает применение гибкого механизма для определения методов визуализации **Пользовательской задачи** посредством пользовательского интерфейса (интерфейс задачи). Данный элемент является опциональным. Методы визуализации (один или более) определяются в атрибутах **Задачи. Пользовательская задача** подразумевает гибкое её использование, независимо от того, подходит ли данный метод визуализации для данных условий или нет. Элемент Rendering является расширением для визуализации. Такие нюансы, как выбор языка, непрозрачны для элемента визуализации, т.к. приложения обычно поддерживают нескольких языков. Если же в каком-либо инструменте визуализации многоязычность не поддерживается, провайдер данного приложения может пожелать усовершенствовать данный тип визуализации для обеспечения его необходимой информацией о языке. Содержание элемента визуализации в данной спецификации не рассматривается.

Люди как исполнители

Люди в **Процессе** могут быть назначены на различные роли (в спецификации WS-HumanTask они определены как «универсальные человеческие роли»). В **BPMN 1.2** для **Действий**, как правило, указывается лишь

Исполнитель (Performer). **BPMN 2.0** позволяет добавлять в **Процесс** другие роли, помимо *Исполнителя*. Это стало возможным благодаря использованию особого элемента *HumanPerformer*, позволяющего указывать более конкретные роли (например, *Потенциальный владелец*).



Фигура 10.23 – Диаграмма классов элемента HumanPerformer

Элемент `HumanPerformer` наследует атрибуты и ассоциации элемента `ResourceRole` (см. таблицу 10.5) посредством взаимоотношения этого элемента с элементом `Performer`, но не может иметь каких-либо других дополнительных атрибутов или ассоциаций.

Потенциальный владелец

Потенциальный владелец **Пользовательской задачи** является лицом, заявляющим о правах на **Задачу** и выполняющим её. Становится текущим владельцем **Задачи**, обычно – при прямо предъявляемом требовании на права владения этой **Задачей**.

Определение XML-схемы пользовательских действий

Таблица 10.15 –XML-схема элемента ManualTask

```
<xsd:element name="manualTask" type="tManualTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tManualTask">
      <xsd:complexContent>
      <xsd:extension base="tTask"/>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.16 – XML-схема элемента UserTask

```
<xsd:element name="userTask" type="tUserTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tUserTask">
      <xsd:complexContent>
            <xsd:extension base="tTask">
                  <xsd:sequence>
                        <xsd:element ref="rendering" minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="implementation" type="tImplementation"
                   default="##unspecified"/>
            </xsd:extension>
      </xsd:complexContent>
```

```
</xsd:complexType>


<xsd:element name="rendering" type="tRendering"/>
<xsd:complexType name="tRendering">
      <xsd:complexContent>
            <xsd:extension base="tBaseElement"/>
      </xsd:complexContent>
</xsd:complexType>


<xsd:simpleType name="tImplementation">
      <xsd:union memberTypes="xsd:anyURI">
            <xsd:simpleType>
                  <xsd:restriction base="xsd:token">
                        <xsd:enumeration value="##unspecified" />
                        <xsd:enumeration value="##WebService" />
                  </xsd:restriction>
            </xsd:simpleType>
      </xsd:union>
</xsd:simpleType>
```

Таблица 10.17 – XML-схема элемента HumanPerformer

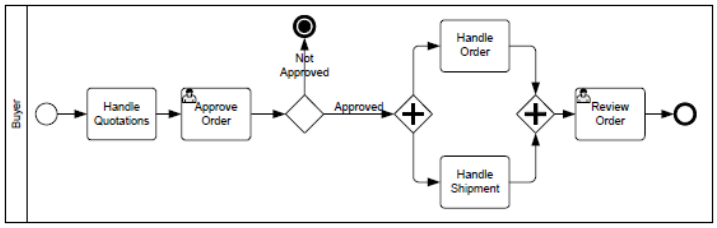
```
<xsd:element name="humanPerformer" type="tHumanPerformer" substitutionGroup="performer"/>
<xsd:complexType name="tHumanPerformer">
      <xsd:complexContent>
            <xsd:extension base="tPerformer">
                  <xsd:sequence>
                        <xsd:element ref="peopleAssignment" minOccurs="1" maxOccurs="1"/>
                  </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.18 – XML-схема элемента PotentialOwner

```
<xsd:element name="potentialOwner" type="tPotentialOwner" substitutionGroup="performer"/>
<xsd:complexType name="tPotentialOwner">
      <xsd:complexContent>
            <xsd:extension base="tHumanPerformer"/>
      </xsd:complexContent>
</xsd:complexType>
```

Примеры

Далее будет рассмотрен образец **Процесса** закупки с точки зрения Покупателя (см. фигуру 10.24).



Фигура 10.24 – Образец Процесса закупки

Данный **Процесс** состоит из двух **Пользовательских Задач**:

- **Подтверждение заказа**. После подтверждения стоимости требуется, чтобы региональный менеджер подтвердил заказ. Это необходимо для дальнейшей доставки товара.

- **Просмотр заказа**. Если заказ был доставлен покупателю хотя бы раз, сам заказ и сопутствующие документы будут снова просматриваться другими лицами.

На вышеприведенной схеме не отображена детальная информация о `ресурсе` и его назначениях. Ниже, в формате XML приведен образец **Процесса** «Покупатель», отображающий использование `ресурса` и его назначение на роль потенциального владельца.

Таблица 10.19 – XML-представление Процесса «Покупатель» в виде последовательности

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions id="Definition"
             targetNamespace="http://www.example.org/UserTaskExample"
             typeLanguage="http://www.w3.org/2001/XMLSchema"
             expressionLanguage="http://www.w3.org/1999/XPath"
              xmlns=http://www.w3.org/2001/XMLSchema
             xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
             xmlns:tns="http://www.example.org/UserTaskExample">

  <resource id="regionalManager" name="Regional Manager">
    <resourceParameter id="buyerName" isRequired="true" name="Buyer Name" type="xsd:string"/>
    <resourceParameter id="region" isRequired="false" name="Region" type="xsd:string"/>
  </resource>

  <resource id="departmentalReviewer" name="Departmental Reviewer">
    <resourceParameter id="buyerName" isRequired="true" name="Buyer Name" type="xsd:string"/>
  </resource>

  <collaboration id="BuyerCollaboration" name="Buyer Collaboration">
    <participant id="BuyerParticipant" name="Buyer" processRef="BuyerProcess"/>
  </collaboration>

   <!-- Process definition -->
  <process id="BuyerProcess" name="Buyer Process">

    <laneSet id="BuyerLaneSet">
      <lane id="BuyerLane">
        <flowNodeRef>StartProcess</flowNodeRef>
        <flowNodeRef>QuotationHandling</flowNodeRef>
        <flowNodeRef>ApproveOrder</flowNodeRef>
        <flowNodeRef>OrderApprovedDecision</flowNodeRef>
        <flowNodeRef>TerminateProcess</flowNodeRef>
        <flowNodeRef>OrderAndShipment</flowNodeRef>
        <flowNodeRef>OrderHandling</flowNodeRef>
        <flowNodeRef>ShipmentHandling</flowNodeRef>
        <flowNodeRef>OrderAndShipmentMerge</flowNodeRef>
        <flowNodeRef>ReviewOrder</flowNodeRef>
        <flowNodeRef>EndProcess</flowNodeRef>
      </lane>
    </laneSet>

   <startEvent id="StartProcess"/>


    <sequenceFlow sourceRef="StartProcess" targetRef="QuotationHandling"/>
    <task id="QuotationHandling" name="Quotation Handling"/>
    <sequenceFlow sourceRef="QuotationHandling" targetRef="ApproveOrder"/>

    <userTask id="ApproveOrder" name="ApproveOrder">
      <potentialOwner>
        <resourceRef>tns:regionalManager</resourceRef>
        <resourceParameterBinding parameterRef="tns:buyerName">
          <formalExpression>getDataInput('order')/address/name</formalExpression>
        </resourceParameterBinding>
        <resourceParameterBinding parameterRef="tns:region">
          <formalExpression>getDataInput('order')/address/country</formalExpression>
        </resourceParameterBinding>
      </potentialOwner>
    </userTask>


    <sequenceFlow sourceRef="ApproveOrder" targetRef="OrderApprovedDecision"/>
```

```
    <exclusiveGateway id="OrderApprovedDecision" gatewayDirection="Diverging"/>
    <sequenceFlow sourceRef="OrderApprovedDecision" targetRef="OrderAndShipment">
      <conditionExpression>Was the Order Approved?</conditionExpression>
    </sequenceFlow>
    <sequenceFlow sourceRef="OrderApprovedDecision" targetRef="TerminateProcess">
      <conditionExpression>Was the Order NOT Approved?</conditionExpression>
    </sequenceFlow>

    <endEvent id="TerminateProcess">
      <terminateEventDefinition id="TerminateEvent"/>
    </endEvent>


    <parallelGateway id="OrderAndShipment" gatewayDirection="Diverging"/>

    <sequenceFlow sourceRef="OrderAndShipment" targetRef="OrderHandling"/>
    <sequenceFlow sourceRef="OrderAndShipment" targetRef="ShipmentHandling"/>


    <task id="OrderHandling" name="Order Handling"/>
    <task id="ShipmentHandling" name="Shipment Handling"/>

    <sequenceFlow sourceRef="OrderHandling" targetRef="OrderAndShipmentMerge"/>
    <sequenceFlow sourceRef="ShipmentHandling" targetRef="OrderAndShipmentMerge"/>
    <parallelGateway id="OrderAndShipmentMerge" gatewayDirection="Converging"/>
    <sequenceFlow sourceRef="OrderAndShipmentMerge" targetRef="ReviewOrder"/>
    <userTask id="ReviewOrder" name="Review Order">
      <potentialOwner>
        <resourceRef>tns:departmentalReviewer</resourceRef>
       <resourceParameterBinding parameterRef="tns:buyerName">
          <formalExpression>getDataInput('order')/address/name</formalExpression>
</resourceParameterBinding>
      </potentialOwner>
    </userTask>


    <sequenceFlow sourceRef="ReviewOrder" targetRef="EndProcess"/>
    <endEvent id="EndProcess"/>

  </process>
</definitions>
```

10.2.5. Подпроцесс

Подпроцесс представляет собой **Действие**, заключающее в себе другие **Действия**, **Шлюзы**, **События** и **Потоки операций**. Графически **Подпроцесс** изображается в качестве элемента **Потока операций Процесса**. **Подпроцесс** также может изображаться «открытым» в случае, если необходимо показать другой **Процесс** внутри данного **Подпроцесса**. **Подпроцесс** определяет контекстные рамки, необходимые для обеспечения видимости атрибутов, рамки транзакции, необходимые для управления исключениями, **Событиями** или компенсацией.

Далее будут рассмотрены различные типы **Подпроцессов**.

Встроенный Подпроцесс (Embedded Sub-Process (Sub-Process))

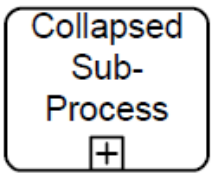
Подпроцесс, как и **Задача**, изображается в виде прямоугольника с закругленным углами.

- **Подпроцесс** представляет собой прямоугольник с закругленным углами, который ДОЛЖЕН БЫТЬ выполнен одинарной тонкой линией.
    - Текст, цвет, размер, а также линии, используемые для изображения **Подпроцесса**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм». Однако следует учитывать следующее исключение:
        - Одинарная жирная линия в изображении границ **Подпроцесса** ДОЛЖНА означать использование данного графического элемента в качестве **Действия Вызов** (**Подпроцесс**).

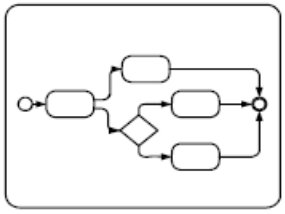
- Пунктирная линия в изображении границ **Подпроцесса** ДОЛЖНА означать использование данного графического элемента в качестве **Событийного Подпроцесса**.
- Двойная линия в изображении границ **Подпроцесса** ДОЛЖНА означать использование данного графического элемента в качестве **Подпроцесса Транзакции**.

Подпроцесс может быть свернутым (Collapsed Sub-Process), при этом его детали скрыты (см. фигуру 10.25). **Подпроцесс** также может быть развернутым (Expanded Sub-Process), при этом его детали отображаются внутри **Процесса**, в котором данный **Подпроцесс** содержится (см. фигуру 10.26). В случае, если **Подпроцесс** является свернутым, то используется маркер, позволяющий отличить **Подпроцесс** от **Задачи**.

- Маркер **Подпроцесса** ДОЛЖЕН изображаться в виде небольшого квадрата, расположенного в центре нижней части графического элемента и заключающего в себе знак «+».



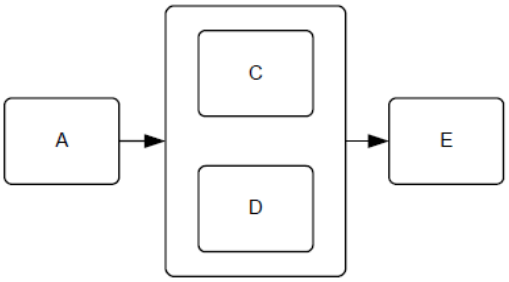
Фигура 10.25 – Графический элемент Свернутый Подпроцесс



Фигура 10.26 – Графический элемент Развернутый Подпроцесс

Подпроцессы используются для создания контекста для управления исключением, применяемым к группе **Действий**. Подобным образом выполняется управление *Компенсацией*.

Развернутый **Подпроцесс** используется для более компактного отображения группы параллельных **Действий** с использованием минимума деталей. Как показано на фигуре 10.27, Действия «C» и «D» заключены в безымянном развернутом **Подпроцессе**. Оба этих **Действия** будут выполняться параллельно. Обратите внимание, что в состав развернутого **Подпроцесса** не включены ни **Стартовое**, ни **Конечное События**. Также в нем не содержится **Потока операций**, исходящего от этих **Событий** или подходящего к ним. Такое использование развернутого **Подпроцесса** для отображения «параллельных блоков» может сделать использование **Стартового** и **Конечного Событий** необязательным.



Фигура 10.27 – Развернутый Подпроцесс, выступающий в роли «параллельного блока».

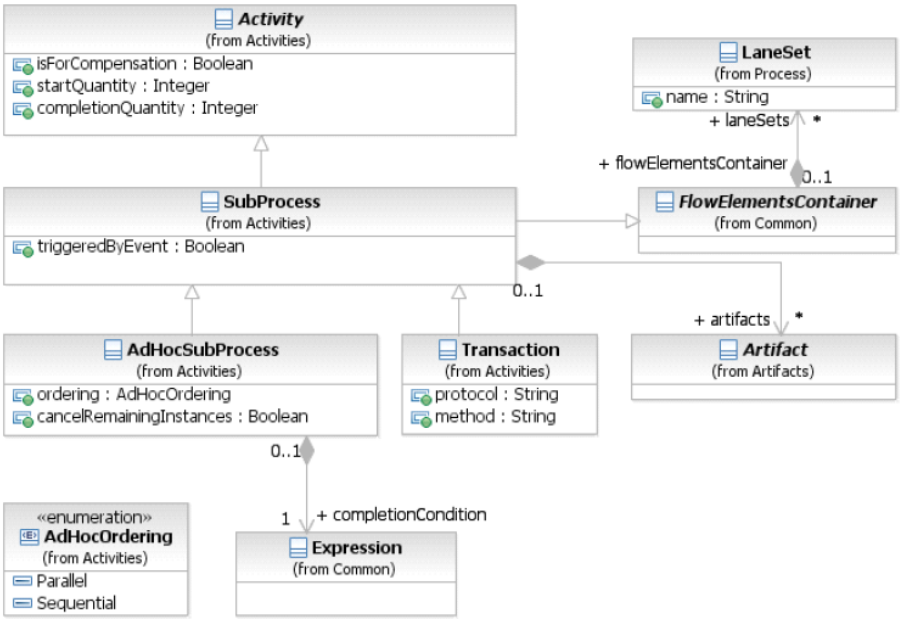
BPMN различает пять типов стандартных маркеров Свернутого **Подпроцесса**. Маркер **Подпроцесса**, изображенный на фигуре 10.25, может сочетаться с оставшимися четырьмя маркерами: Маркером *Цикла* (*Loop* Marker), *Многоэкземплярным* Маркером (*Multiple-Instance* Marker), Маркером **Компенсации** (**Compensation** Marker) и Маркером **Ad Hoc** (**Ad-Hoc** Marker). Свернутый **Подпроцесс** может содержать от одного до трех вышеуказанных Маркеров. Комбинации Маркеров могут быть любыми, кроме сочетания Маркеров *Цикличности* и *Многоэкземплярного*, - эти Маркеры не могут изображаться одновременно (см. фигуру 10.28).

- Маркер *Цикла* ДОЛЖЕН БЫТЬ выполнен в виде небольшой стрелки, острие которой загнуто в направлении, противоположном направлению самой стрелки.
    - Маркер *Цикла* МОЖЕТ сочетаться с любым другим Маркером **Подпроцесса**, кроме *Многоэкземплярного*
- *Многоэкземплярный* Маркер ДОЛЖЕН БЫТЬ выполнен в виде трех параллельных вертикальных линий.
    - *Многоэкземплярный* Маркер МОЖЕТ сочетаться с любым другим Маркером **Подпроцесса**, кроме Маркера *Цикла*.
- Маркер **Ad Hoc** ДОЛЖЕН БЫТЬ выполнен в виде тильды.
    - Маркер **Ad Hoc** МОЖЕТ сочетаться с любым другим Маркером **Подпроцесса**.
- Маркер **Компенсации** ДОЛЖЕН БЫТЬ выполнен в виде двух треугольников, повернутых влево (как кнопка перемотки назад на проигрывателе).
    - Маркер **Компенсации** МОЖЕТ сочетаться с любым другим Маркером **Подпроцесса**.
- Все вышеописанные Маркеры при совместном отображении ДОЛЖНЫ БЫТЬ сгруппированы и располагаться в центре нижней части графического элемента **Подпроцесса**.

Фигура 10.28 – Маркеры Свернутого Подпроцесса

Свернутый **Подпроцесс** в **BPMN 2.0** относится к Встроенному **Подпроцессу**, описанному в **BPMN 1.2**. Повторно используемый **Подпроцесс** в **BPMN 1.2** относится к **Действию** типа **Вызов**, описанному в **BPMN 2.0**.

Фигура 10.29 представляет собой диаграмму классов **Подпроцесса**.



Фигура 10.29 - Диаграмма классов элемента SubProcess

Элемент **SubProcess** наследует атрибуты и ассоциации элементов **Activity** (см. таблицу 10.3) и элемента `FlowElementContainer` (см. таблицу 8.45). Таблица 10.20 содержит информацию о дополнительных атрибутах элемента **SubProcess**.

Таблица 10.20 – Атрибуты элемента SubProcess

Название атрибута	Описание/использование
triggeredByEvent: boolean = false	Данный атрибут указывает на то, что данный **Подпроцесс** работает с **Событиями**. • Значение «*false*» указывает на то, что **Подпроцесс** является стандартным. • Значение «*true*» указывает на то, что **Подпроцесс** работает с **Событиями** и является причиной возникновения дополнительных ограничений.
triggeredByEvent: boolean = false	Данный атрибут обусловливает наличие списка **Артефактов**, хранящихся в **Подпроцессе**.

Reusable Sub-Process (Call Activity) Повторно используемый Подпроцесс (Вызов)

Повторно используемый **Подпроцесс**, описанный в **BPMN 1.2**, относится к **Действию** типа **Вызов**, используемому для вызова предопределенного **Подпроцесса**.

Событийный Подпроцесс (Event Sub-Process)

Событийным Подпроцессом называется специфический **Подпроцесс**, используемый внутри **Процесса** (**Подпроцесса**). Такой **Подпроцесс** имеет атрибут `triggeredByEvent` с установленным значением «*true*».

Такой **Подпроцесс** не является частью **Стандартного Потока Операцийий**, включенного в родительский **Процесс**, и не имеет *входящих* или *исходящих* **Потоков Операций**.

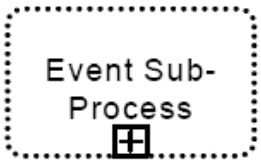
- **Событийный Подпроцесс** НЕ ДОЛЖЕН иметь *входящих* или *исходящих* **Потоков операций**.

Событийный Подпроцесс МОЖЕТ появляться (один раз или многократно) или НЕ появляться в ходе выполнения родительского **Процесса**. Отличие такого **Подпроцесса** от стандартного состоит в том, что стандартный **Подпроцесс** в качестве *триггера* использует **Поток операций**, а **Событийный Подпроцесс** - **Стартовое событие**. Всякий раз, когда какое-то **Стартовое событие** запускается во время выполнения родительского **Процесса**, запускается и **Событийный Подпроцесс**.

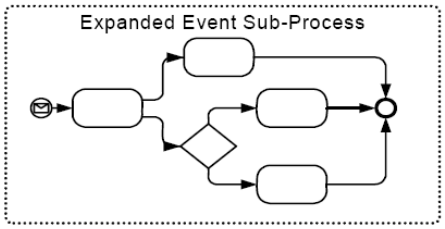
- **Стартовое Событие Событийного Подпроцесса** ДОЛЖНО иметь определенный *триггер*.
    - **Стартовое событие (EventDefinition)** ДОЛЖНО БЫТЬ одного из следующих типов: `Сообщение, Ошибка, Эскалация, Компенсация, Условие, Сигнал, Множественный.`
- **Событийный Подпроцесс** ДОЛЖЕН содержать одно или более **Стартовое Событие**.

Такой **Подпроцесс** изображается в виде прямоугольника с закругленным углами (установленное в **BPMN** отображение графического элемента **Подпроцесс**).

- **Событийный Подпроцесс** представляет собой прямоугольник с закругленным углами, который ДОЛЖЕН БЫТЬ выполнен одинарной тонкой пунктирной линией (см. фигуры 10.30 и 10.31).
    - Текст, цвет, размер, а также линии, используемые для изображения данного **Подпроцесса**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм». Однако следует учитывать следующее исключение:
        - Если **Событийный Подпроцесс** является свернутым, то **Стартовое Событие** в таком **Подпроцессе** будет являться маркером и отображаться в левом верхнем углу фигуры **Подпроцесса** (см. фигуру 10.30).



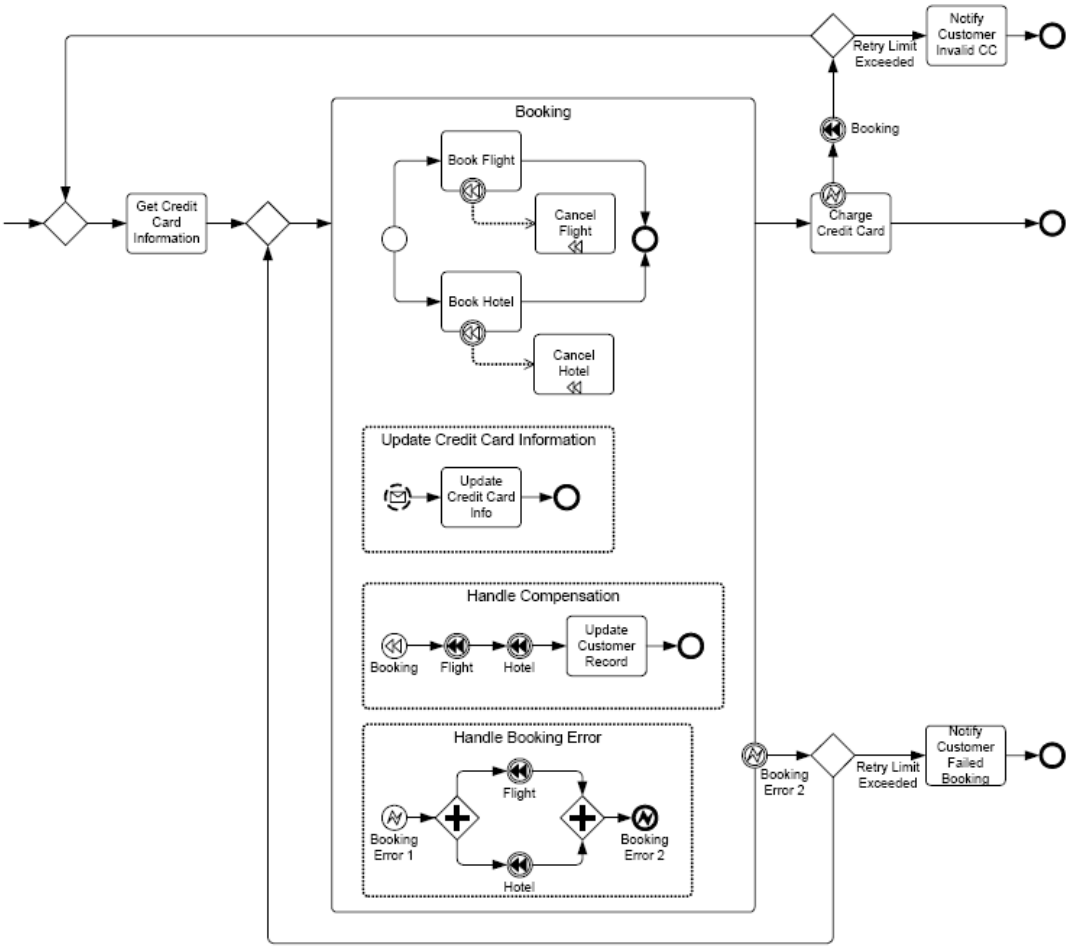
Фигура 10.30 – Графический элемент Свернутый Событийный Подпроцесс



Фигура 10.31 – Графический элемент Развернутый Событийный Подпроцесс

Запуск **Событийного Подпроцесса** может привести к следующим последствиям в родительском **Процессе**:1) Родительский **Процесс** прерывается, 2) Родительский **Процесс** продолжает выполняться (не прерывается). То, каким будет результат, определяется выбором типа **Стартового события**. Для получения более подробной информации о **Стартовых событиях**, прерывающих или не прерывающих течение родительского **Процесса**, смотрите соответствующие разделы.

На фигуре 10.32 изображен пример **Подпроцесса**, включающего в себя три **Событийных Подпроцесса**. Первый из них вызывается **Сообщением**, не прерывает родительский **Подроцесс** и может появляться многократно. Второй такой **Подпроцесс** используется в целях *компенсации* и появляется только после того, как родительский **Подпроцесс** будет завершен. Третий управляет ошибками, возникающими в ходе выполнения родительского **Подпроцесса**, и если будет запущен, то прервет его.

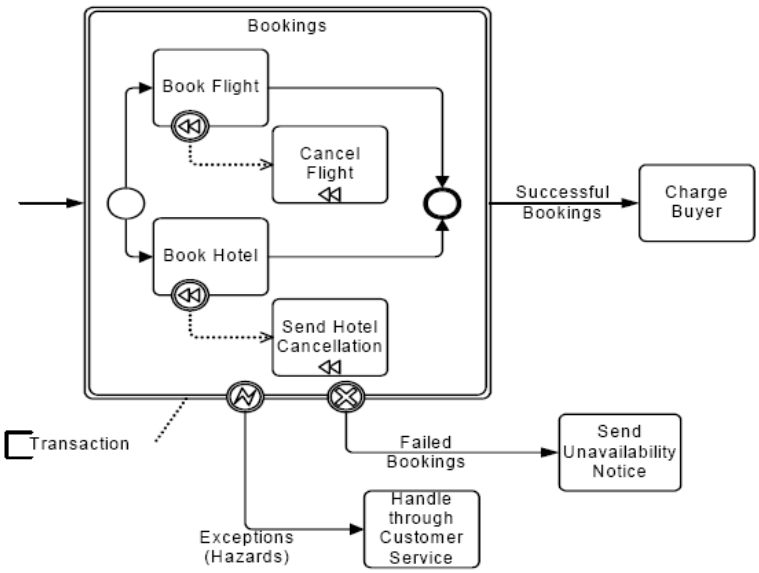


Фигура 10.32 –Событийные Подпроцессы внутри родительского Подпроцесса

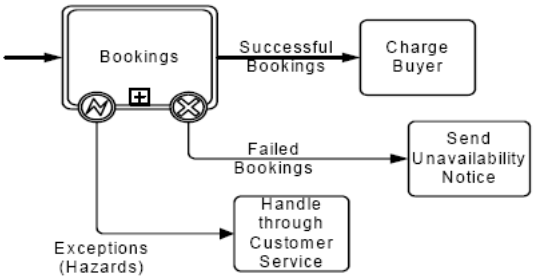
Транзакция (Transaction)

Транзакцией называется специфический тип **Подпроцесса**, который демонстрирует определенное поведение, контролируемое посредством протокола транзакции (например, WS-Transaction). Граница графического элемента **Транзакция** выполнена двойной линией (см. фигуру 10.33).

- **Транзакция** представляет собой прямоугольник с закругленным углами, который ДОЛЖЕН БЫТЬ выполнен двойной тонкой линией.
    - Текст, цвет, размер, а также линии, используемые для изображения данного **Подпроцесса**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».



Фигура 10.33 –Подпроцесс Транзакция



Фигура 10.34 – Свернутый Подпроцесс Транзакция

Элемент **Transaction Sub-Process** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3) за счет взаимосвязи с **Подпроцессом**. Таблица 10.21 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Transaction Sub-Process**.

Таблица 10.21 – Атрибуты и ассоциации элемента Transaction Sub-Process

Название атрибута	Описание/использование
method: Transaction- Method	Данный атрибут определяет `метод`, используемый для совершения **Транзакции** или её отмены. Для выполняемых процессов данный атрибут ДОЛЖЕН содержать особый URL, например, http://schemas.xmlsoap.org/ws/2004/10/wsat for WS-AtomicTransaction, или http://docs.oasis-open.org/ws-tx/wsba/2006/ 06/AtomicOutcome для WS-BusinessActivity. Для сохранения совместимости с **BPMN 1.1** данный атрибут также может иметь значения "`##compensate`", "`##store`" и "`##image`".

Использование **Транзакции** может привести к следующим результатам:

1. **Удачное завершение**. Отображается в виде Стандартного **Потока Операций**, отходящего от **Транзакции**.
2. **Неудачное завершение** (с использованием **События Отмена**). При отказе от выполнения **Транзакции Действия**, находящиеся внутри данной **Транзакции**, подвергнуться действиям отмены: например, будет выполнена *компенсация* определенных **Действий**, или произойдет возврат к **Процессу** (для получения более подробной информации о *компенсации* смотрите соответствующий раздел документа). Обратите внимание, что никакой другой механизм остановки **Транзакции** (например, использование Событий Ошибка и Таймер или каких-либо **Действий**, не предусмотренных **Транзакцией**) не повлечет за собой *компенсацию*. **Промежуточное Событие Отмена**, соединенное с границей **Действия**, оказывает влияние на направление хода **Процесса** после того, как произошел возврат и была выполнена *компенсация*. Это **Событие** может быть использовано только в том случае, если оно соединено с границей **Подпроцесса Транзакции**. Оно не может быть использовано в рамках *Стандартного Потока Операций* или не прикрепляется к **Подпроцессу**, не являющемуся **Транзакцией**. Существует два механизма, способных сигнализировать об отмене **Транзакции**:
    - *Поток операций* **Транзакции** достигает **Промежуточного события Отмена**. Это **Событие** используется только для такого типа **Подпроцессов**.
    - **Сообщение** об отмене может быть получено посредством протокола транзакции, поддерживающего выполнение данной **Транзакции**.
3. **Риск (Опасность)**. Появление *Риска* означает, что какое-то действие в **Транзакции** выполняется крайне неверно, и это ведет к тому, что стали невозможны ни удачное завершение, ни отмена. *Риски* отображаются посредством **Промежуточного события Ошибка**. При появлении *Риска* выполнение текущего **Действия** прекращается без возможности *компенсации,* а *Поток операций* возобновляется от **Промежуточного события Ошибка**.

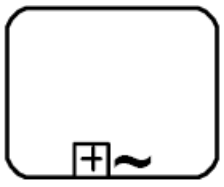
Поведение, которое при удачном завершении демонстрирует **Транзакция** на конечном этапе, немного отличается от того, как завершается стандартный **Подпроцесс**. Когда все маршруты, содержащиеся в **Транзакции**, достигают **Конечного события** (не имеющего тип Отмена), моментальный возврат *Потока операций* к родительскому **Процессу** верхнего уровня, как это происходит в стандартном **Подпроцессе,** не осуществляется. Прежде всего, протоколом транзакции утверждаются все *Участники*, удачно завершившие **Транзакцию**. В большинстве случаев так и происходит, после чего *Поток операций* возвращается в **Процесс** верхнего уровня. Однако случается, что у одного из *Участников* возникают проблемы при завершении, что влечет за собой появление *Отмены* или *Риска* (*Опасности*). При таком развитии событий *Поток операций* направляется к соответствующему **Промежуточному событию**, даже если оно, скорее всего, было успешно завершено.

Спонтанный Подпроцесс (Ad-Hoc Sub-Process)

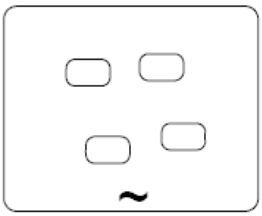
Спонтанным Подпроцессом называется особый тип **Подпроцесса**, представляющий собой группу **Действий**, взаимоотношения между которыми не поддаются строго регламентированным правилам. Для **Процесса** определяется набор **Действий**, однако, их последовательность и количество выполнений определяются исполнителями этих **Действий**.

Графический элемент **Спонтанный Подпроцесс** содержит маркер, выполненный в виде знака тильды и располагающийся в центре нижней части фигуры **Подпроцесса** (см фигуру 10.35 и 10.36).

- Маркер **Спонтанного Подпроцесса** ДОЛЖЕН БЫТЬ выполнен в виде тильды.
    - Маркер **Ad Hoc** МОЖЕТ сочетаться с любым другим Маркером **Подпроцесса**.



Фигура 10.35 – Графический элемент Свернутый Спонтанный Подпроцесс



Фигура 10.36 – Графический элемент Развернутый Спонтанный Подпроцесс

Элемент **Ad-HocSub-Process** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3) за счет взаимосвязи с **Подпроцессом**.

Таблица 10.22 содержит информацию о дополнительных ассоциациях элемента **Ad-HocSub-Process**.

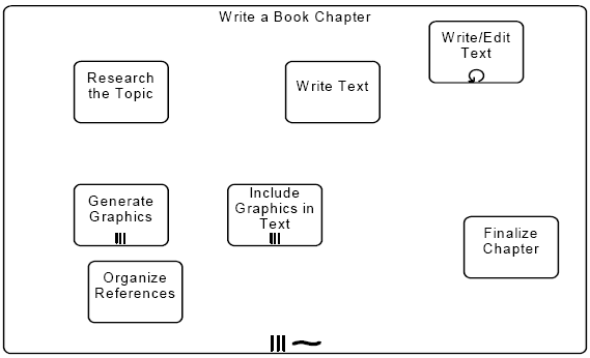
Таблица 10.22 – Ассоциации элемента Ad-HocSub-Process

Название атрибута	Описание/использование
completionCondition: Expression	Данный атрибут определяет условия, при которых завершается **Процесс**. Значение «*true*» указывает на то, что **Процесс** будет завершен.
ordering: AdHocOrdering = Parallel { Parallel | Sequential }	Данный атрибут указывает на то, будут ли **Действия**, включенные в **Процесс**, выполняться параллельно или НЕОБХОДИМО последовательное их выполнение. Значением по умолчанию является «`parallel`». Значение «`sequential`» ограничивает одновременное выполнение нескольких **Действий**. В данном случае в определенный период времени может быть выполнено лишь одно **Действие**. Если выбрано значение «`parallel`», то одновременно может выполняться любое количество **Действий**, от нуля и более.
cancelRemaining- Instances: boolean = true	Данный атрибут используется только в том случае, если для вышеописанного атрибута ordering установлено значение «parallel». Указывает на то, будут ли отменены запущенные *экземпляры* Действий, если значение атрибута `completionCondition` становится «*true*».

Действия, включенные в состав **Спонтанного Подпроцесса**, друг с другом, как правило, не соединяются. В ходе выполнения **Подпроцесса** запущенными МОГУТ быть одно или несколько **Действий**. **Действия** МОГУТ выполняться многократно. То, когда будет запущено **Действие**, каким будет следующее **Действие**, а также другие детали выполнения **Действий** определяются *Исполнителем*.

Примерами **Процессов**, входящих в состав **Спонтанного Подпроцесса**, являются разработка кода (на низком уровне), поддержка клиентов, а также написание главы книги. Рассмотрим, к примеру, написание главы для книги. Мы видим, что данный **Процесс** включает следующие **Действия**: поиск темы, составление текста, редактирование текста, создание дизайна, графическое оформление текста, оформление ссылок и т.д. (см. фигуру 10.37). В таком **Процессе** МОЖЕТ наблюдаться определенная зависимость **Задач** друг от друга, например, редактирование текста не может происходить раньше его написания. Однако такая корреляция между *экземплярами* **Задачам** по написанию и редактированию текста не обязательна. Редактирование может возникать

нерегулярно и зависит от текста, полученного в результате выполнения нескольких *экземпляров* **Задачи** по написанию текста.

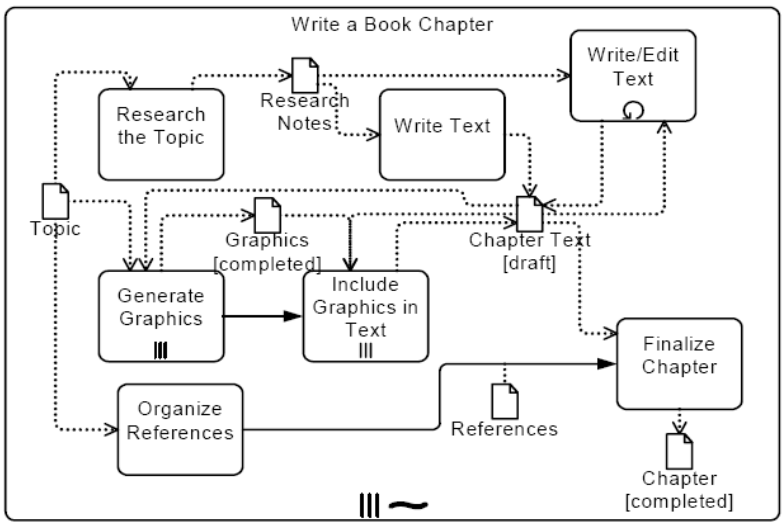


Фигура 10.37 – Спонтанный Подпроцесс написания главы для книги

Хотя в **Спонтанном Подпроцессе** структура не определена, в его детали все же может быть добавлена информация о последовательности **Действий** или корреляции данных. К примеру, можно расширить вышеописанный **Спонтанный Подпроцесс** написания главы для книги путем добавления в него **Объектов данных, Ассоциаций** или **Потоков операций** (см. фигуру 10.38).

Однако использование в **Спонтанном Подпроцессе** элементов потока должно осуществляться со следующими ограничениями, не имеющими отношения к использованию элементов в **Подпроцессах** других типов:

- В список элементов **BPMN**, которые ДОЛЖНЫ использоваться в **Спонтанном Подпроцессе**, входит **Действие**.
- В список элементов **BPMN**, которые МОГУТ использоваться в **Спонтанном Подпроцессе**, входят: **Объект данных, Поток операций, Ассоциация, Ассоциация данных, Группа, Поток сообщений** (может являться как *целью*, так и *результатом*), **Шлюз**, **Промежуточное событие**.
- В список элементов **BPMN**, которые НЕ ДОЛЖНЫ использоваться в **Спонтанном Подпроцессе**, входят: **Стартовое событие, Конечное событие, Переговоры** (графически), **Соединители переговоров** (графически), **Действия Хореографии**.



Фигура 10.38 – Спонтанный Подпроцесс написания главы для книги с отображением последовательности Действий и зависимых данных

Объект данных, предоставленный для **Задач** на входе, является дополнительным ограничением для выполнения этих **Задач**. В данном случае *Исполнители*, хотя и решают, когда будут выполнены **Задачи**, уже

ограничены в действиях тем, что не могут начать выполнение **Задачи** без соответствующих данных. Наличие **Потока операций** между **Задачами** (например, между разработкой дизайна и оформлением текста в соответствии с дизайном) устанавливает зависимость, согласно которой вторая **Задача** ДОЛЖНА быть выполнена после выполнения первой. Это не означает, что вторая **Задача** должна выполняться незамедлительно, но лишь то, что она ДОЛЖНА выполняться после того, как будет завершена первая **Задача**.

Проблемой для BPM стало отслеживание статуса **Спонтанного Подпроцесса**. Как правило, **Процессы** такого типа контролируются при помощи приложений для групповой работы (например, e-mail). **BPMN** позволяет моделировать **Процессы**, необязательные для выполнения, хотя определенные механизмы для выполнения **процессов**, способные отслеживать **Спонтанные Подпроцессы**, существуют. Исходя из этого, **Спонтанный Подпроцесс** будет завершен при определенных условиях, которые указываются посредством установки значения для атрибута `completionCondition`, определяющего, в свою очередь, атрибуты **Процесса**, корректируемые **Действием** данного **Процесса**.

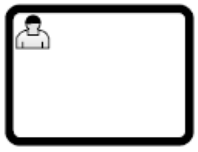
10.2.6. Действие Вызов

Действие Вызов определяет отрезок **Процесса**, где используется глобальный **Процесс** (Global **Process**) или `Глобальная Задача` (`Global Task`). Оно оформляет их вызов в ходе выполнения **Процесса**. Результатом запуска этого **Действия** является передача управления глобальным **Процессом** или `Глобальной Задачей`.

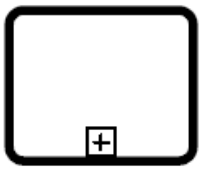
Действие, названное в **BPMN 2.0** «**Вызов**», в **BPMN 1.2** называется *повторно используемым* **Подпроцессом**. **Подпроцесс** в **BPMN 2.0** и *Встроенный* **Подпроцесс** в **BPMN 1.2** являются одним и тем же (см. предыдущий подраздел).

Действие Вызов, как **Задачи** и **Подпроцессы**, отображается на диаграмме **Процесса** в виде прямоугольника с закругленными углами, однако, особенности его отображения зависят от того, что является целью вызова:

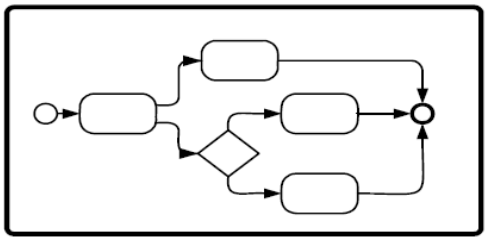
- Если целью **Вызова** является `Глобальная Задача`, то границы прямоугольника ДОЛЖНЫ БЫТЬ выполнены жирной линией (см. фигуру 10.39).
    - Действие **Вызов** в данном случае ДОЛЖНО содержать маркер `глобальной Задачи` (например, при вызове `глобальной Пользовательской задачи` в графическом элементе данного **Действия** будет отображаться маркер **Пользовательской задачи**).
- Если целью **Вызова** является **Процесс**, то **Действие Вызов** будет отображаться следующим образом:
    - Если детали вызываемого **Процесса** скрыты, то графический элемент **Действия** будет выглядеть, как *Свернутый* **Подпроцесс**, границы которого ДОЛЖНЫ БЫТЬ выполнены жирной линией (см. фигуру 10.40).
    - Если детали вызываемого **Процесса** прозрачны, то графический элемент **Действия** будет выглядеть, как *Развернутый* **Подпроцесс**, границы которого ДОЛЖНЫ БЫТЬ выполнены жирной линией (см. фигуру 10.41).



Фигура 10.39 – Графический элемент Действие «Вызов», используемый для вызова глобальной Задачи



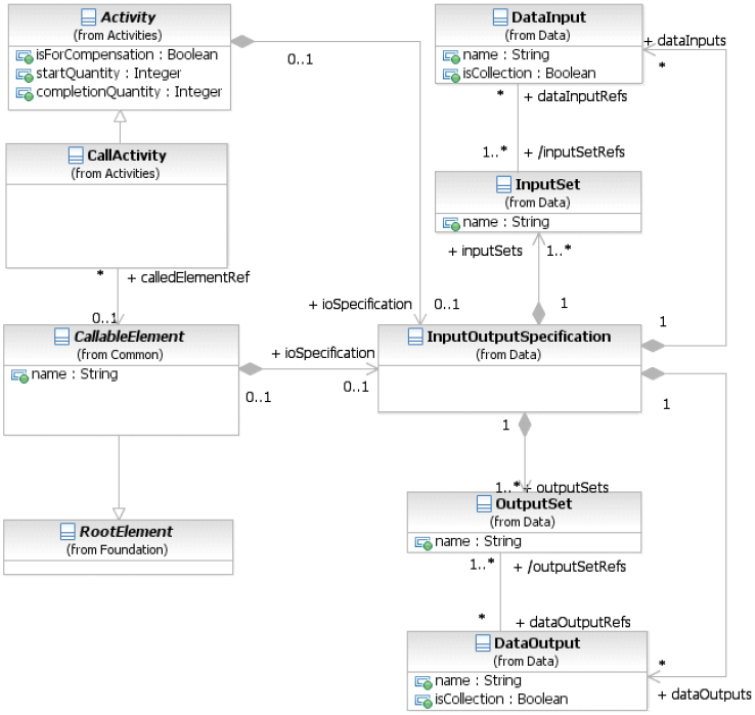
Фигура 10.40 – Графический элемент Действие «Вызов», используемый для вызова Процесса (свернутый)



Фигура 10.41 – Графический элемент Действие «Вызов», используемый для вызова Процесса (развернутый)

Когда **Процесс** со **Взаимоотношением** вызывает другой **Процесс** со **Взаимоотношением**, соответствие между *Участниками* этих **Взаимоотношений** производится посредством элемента `ParticipantAssociations`, принадлежащего **Взаимоотношению** вызывающего **Процесса**.

Действие Вызов ДОЛЖНО удовлетворять требованиям, предъявляемым к данным, а также возвращать данные, предоставляемые вызванным элементом `CallableElement` (см. фигуру 10.42). Это означает, что элементы, входящие в состав атрибута `InputOutputSpecification` для **Действия Вызов**, ДОЛЖНЫ точно соответствовать элементам, содержащимся в атрибуте `CallableElement`, на который он ссылается. Сюда входят такие элементы, как `DataInputs, DataOutputs, InputSets и OutputSets`.



Фигура 10.42 – Диаграмма классов элемента CallActivity

Свойства и атрибуты **Действия Вызов** могут замещать свойства и атрибуты вызываемого элемента. Таким образом, **Вызов** потенциально изменяет поведение этого элемента в соответствии с контекстом вызова. К примеру, когда для **Действия** данного типа определены один или более элементов `ResourceRole`, то значения, указанные посредством `CallableElement`, не учитываются, - вместо них используются те, что определены для **Действия Вызов**. Также и размещенные вдоль иерархии **События** (ошибки и эскалации) находятся на протяжении отрезка от вызываемого элемента до **Действия Вызов** (контроль над их выполнением может осуществляться на границе **Действия**).

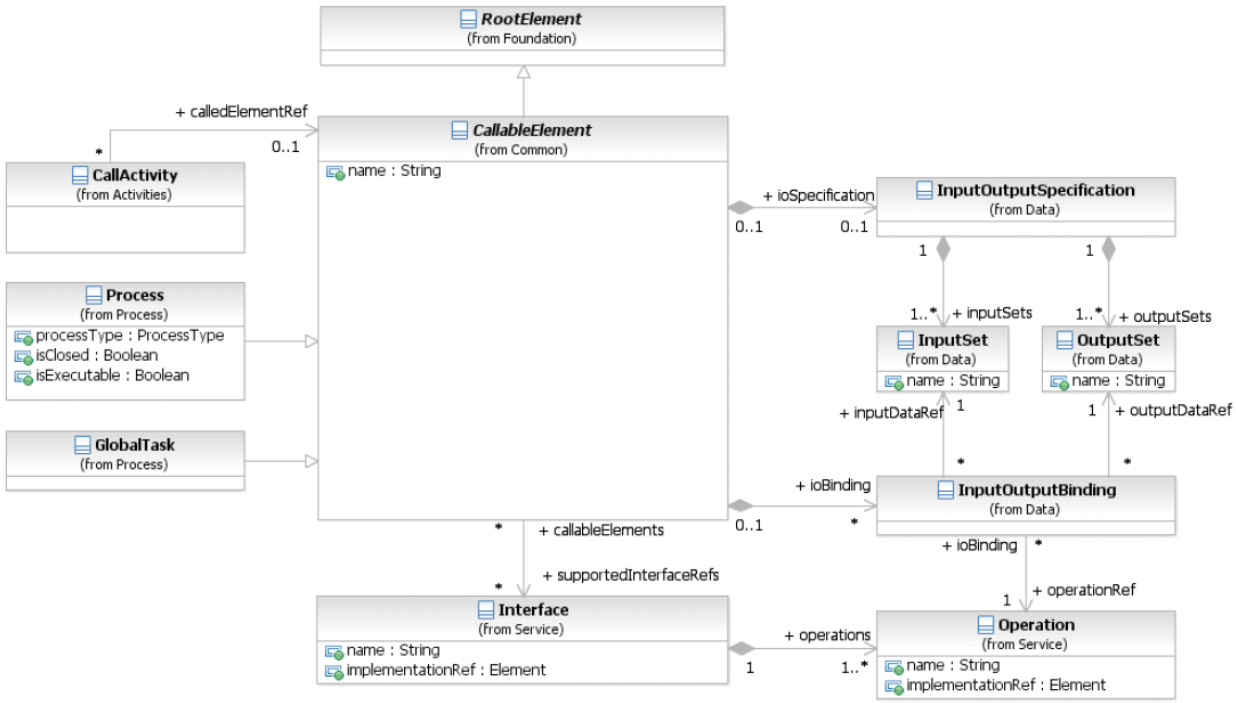
Элемент **CallActivity** наследует атрибуты и ассоциации элемента **Activity** (см. таблицу 10.3). Таблица 10.23 содержит информацию о дополнительных ассоциациях элемента **CallActivity**.

Таблица 10.23 – Ассоциации элемента CallActivity

Название атрибута	Описание/использование
calledElement: CallableElement [0..1]	Данный атрибут определяет вызываемый элемент: **Процесс** или `глобальную Задачу`. Другие вызываемые элементы (**Хореография** (**Choreography**), `глобальная Задача Хореографии`, **Переговоры**, `глобальная Связь`) НЕ ДОЛЖНЫ использоваться **Вызовом**.

Элемент CallableElement

Элемент `CallableElement` является абстрактным суперклассом всех **Действий**. Определяется за рамками **Процесса** или **Хореографии**, но **Действием Вызов** может быть вызван (повторно использован) в данном **Процессе** или **Хореографии**. Вызываемый элемент МОЖЕТ ссылаться на элементы `Interfaces`, определяющий предоставляемые им операции обслуживания. Согласно **BPMN**, в качестве вызываемых **Действиями Вызов** элементов могут выступать **Процесс** и `глобальная Задача` (см. фигуру 10.43). Элементы `CallableElements` относятся к элементам `RootElements`, которые могут быть импортированы и использованы в других элементах (`Definitions`). В случае, если элемент `CallableElement` (например, **Процесс**) указан, он хранится в этих элементах (`Definitions`).



Фигура 10.43 – Диаграмма классов вызываемого элемента CallableElement

Элемент `CallableElement` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.24 содержит информацию о дополнительных атрибутах и ассоциациях элемента `CallableElement`.

Таблица 10.24 – Атрибуты и ассоциации вызываемого элемента CallableElement.

Название атрибута	Описание/использование
name: string [0..1]	Описательное имя данного атрибута.
supportedInterfaceRefs: Interface [0..*]	Значение данного атрибута используется для описания внешнего поведения, демонстрируемого этим элементом.
ioSpecification: Input OutputSpecification [0..1]	Данный атрибут определяет *входы* и *выходы*, а также набор `Входных` и `Выходных Данных` для **Действия**.
ioBinding: InputOutput Binding [0..*]	Данный атрибут определяет комбинацию одного набора `Входных Данных` и одного набора `Выходных Данных` для связывания их с `Операцией`, указанной в `Интерфейсе`.

Если элемента `CallableElement` представлен в виде сервиса, для него необходимо определить значения одного или более элементов `InputOutputBinding`. Элемент `InputOutputBinding` связывает один `Вход` и один `Выход` с `Операцией Service Interface`. Таблица 10.25 содержит информацию о дополнительных ассоциациях элемента `InputOutputBinding`.

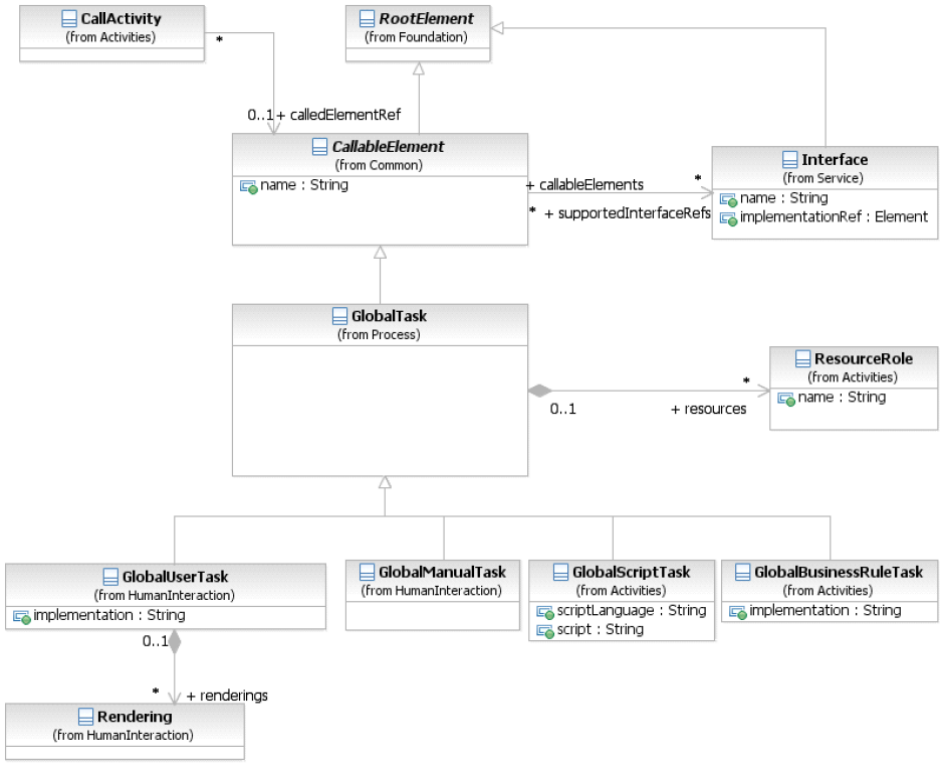
Таблица 10.25 – Ассоциации элемента InputOutputBinding

Название атрибута	Описание/использование
inputDataRef: DataInput	Данный атрибут представляет собой ссылку на определенный `Ввод Данных`, указанный как часть элемента `InputOutputSpecification`,

	относящегося к **Действию**.
outputDataRef: DataOutput	Данный атрибут представляет собой ссылку на определенный `Вывод Данных`, указанный как часть элемента `InputOutputSpecification`, относящегося к **Действию**.
operationRef: Operation	Данный атрибут представляет собой ссылку на определенную `Операцию`, указанную как часть `Интерфейса`, относящегося к **Действию**.

10.2.7. Глобальная Задача (Global Task)

`Глобальная Задача` представляет собой повторно используемую *элементарную* **Задачу**, которая может быть вызвана **Действием Вызов** из любого **Процесса**.



Фигура 10.44 – Диаграмма классов элемента GlobalTask

Элемент `GlobalTask` наследует атрибуты и ассоциации элемента `CallableElement` (см. таблицу 10.24). Таблица 10.26 содержит информацию о дополнительных ассоциациях элемента `GlobalTask`.

Таблица 10.26 – Ассоциации элемента GlobalTask

Название атрибута	Описание/использование
resources: ResourceRole [0..*]	Посредством данного атрибута указывается исполнитель `глобальной Задачи` или лицо, ответственное за её выполнение. Если **Действие Вызов**, относящееся к данной `глобальной Задаче`, использует свои ресурсы, то ресурсы, указанные в данном атрибуте, игнорируются.

Типы глобальной Задачи

В **BPMN** существует несколько различных типов **Задач**, предназначенных для разграничения типов наследуемого ими поведения. Типы глобальных Задач представляют собой лишь подгруппу в рамках классификации типов **Задач**. Согласно **BPMN**, глобальными могут быть Пользовательская задача, Ручное выполнение, Сценарий и Бизнес-правило. Для облегчения понимания данной спецификации все типы **Задач** описаны в одном месте – подразделе **10.2.3.1**. Поведение, демонстрируемое разными типами **Задач**, атрибуты и ассоциации, описанные в вышеуказанном подразделе, - все это также применимо и к соответствующим типам **Глобальных Задач**.

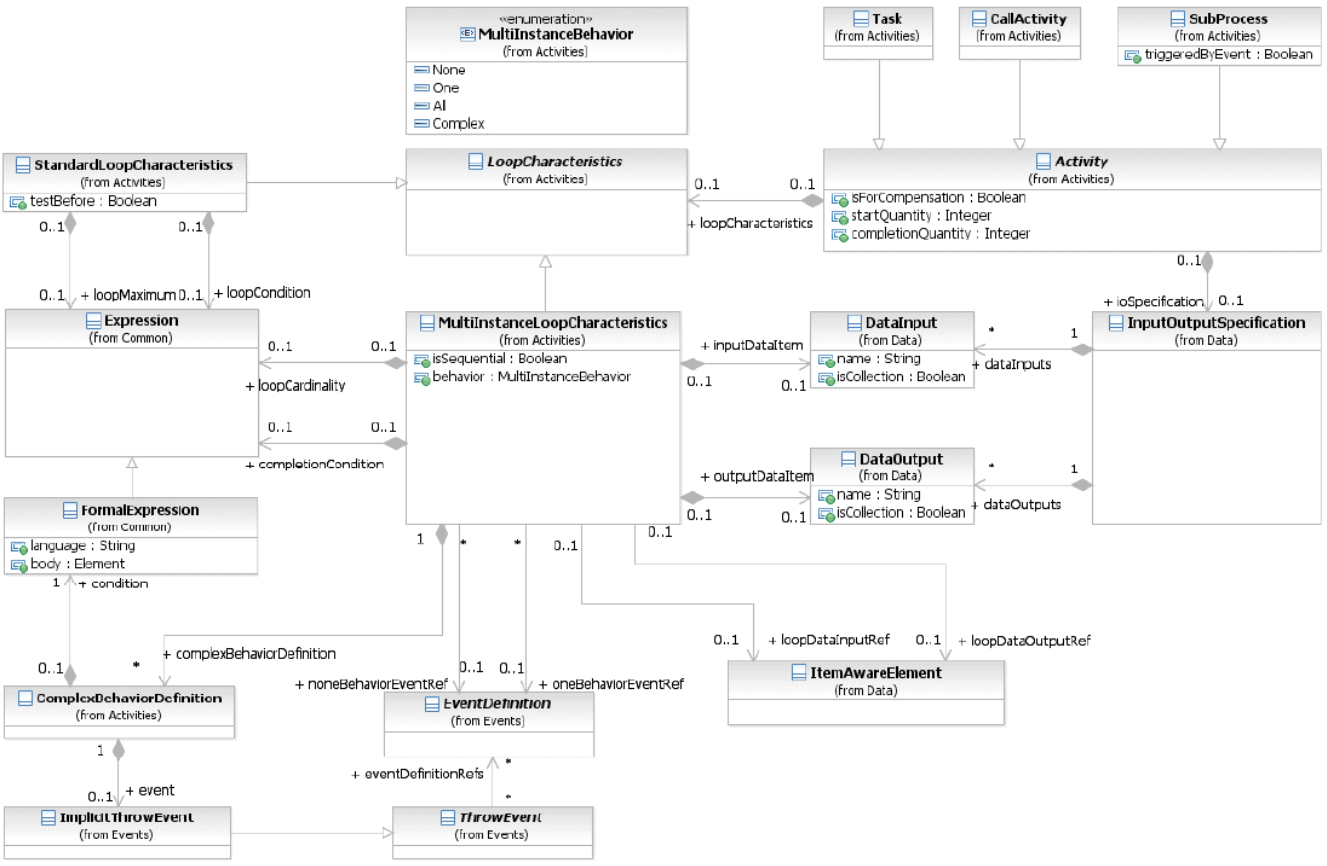
10.2.8. Характеристики цикличности

Действия МОГУТ повторно выполняться в определенной последовательности, т.е. выполняться *циклами*. Наличие элемента LoopCharacteristics свидетельствует о циклическом поведении **Действия**. Элемент LoopCharacteristics представляет собой абстрактный класс. Более конкретные подклассы определяют особенности типов циклического поведения.

Элемент LoopCharacteristics наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Однако он не может иметь каких-либо других собственных атрибутов или ассоциаций.

При этом каждый *экземпляр* **Циклически Выполняемого Действия** имеет атрибуты, на значения которых МОГУТ ссылаться выражения. Эти значения доступны только в момент выполнения такого **Действия**.

Фигура 10.45 отображает диаграмму классов характеристик циклического **Действия**, включающую описание, как стандартного *цикла*, так и *многоэкземплярности*.



Фигура 10.45 – Диаграмма классов элемента LoopCharacteristics

Элемент `LoopCharacteristics` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5), однако не может иметь каких-либо других атрибутов или ассоциаций. Содержимое таблицы 10.27 показывает, что **Действие Цикл** не имеет дополнительных атрибутов *экземпляров*.

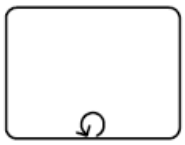
Таблица 10.28 – Атрибуты экземпляров цикличного Действия

Название атрибута	Описание/использование
loopCounter: integer	Данный атрибут используется во время выполнения **Процесса** для подсчета числа *циклов*. Автоматически обновляется механизмом выполнения **Процесса**.

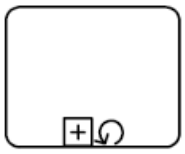
Standard Loop Characteristics Характеристики стандартного цикла

Класс `StandardLoopCharacteristics` определяет поведение *цикла*, основанное на проверке истинности `условного` выражения. **Действие** будем *циклическим* до тех пор, пока проверяемое условие вычисляется как «*true*». Условие вычисляется для каждой из итераций *цикла*. Оно МОЖЕТ проверяться как в начале, так и в конце итерации. Также, по желанию, может указываться максимальное количество итераций, которое НЕ МОЖЕТ быть превышено.

- Маркер стандартного *цикла*, помещенный в фигуру **Задачи** или **Подпроцесса**, ДОЛЖЕН БЫТЬ выполнен в виде небольшой стрелки, острие которой загнуто в направлении, противоположном направлению самой стрелки (см. фигуры 10.46 и 10.47).
    - Маркер цикла МОЖЕТ сочетаться с Маркером **Компенсации**.



Фигура 10.46 – Графический элемент Задача с заключенным в нем маркером стандартного цикла



Фигура 10.46 – Графический элемент Подпроцесс с заключенным в нем маркером стандартного цикла

Элемент `StandardLoopCharacteristics` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством связи с элементом `LoopCharacteristics`. Таблица 10.28 содержит информацию о дополнительных атрибутах и ассоциациях элемента `StandardLoopCharacteristics`.

Таблица 10.28 – Атрибуты и ассоциации элемента StandardLoopCharacteristics

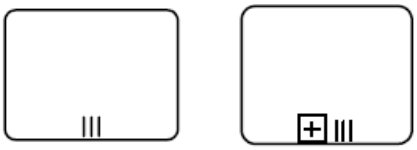
Название атрибута	Описание/использование
testBefore: boolean = false	Данный атрибут используется для осуществления контроля над выполнением условия в начале (`testBefore` = *true*) или конце (`testBefore` = *false*) итерации цикла.
loopMaximum: integer [0..1]	Данный атрибут используется для указания максимального количества итераций цикла.

loopCondition: Expression [0..1]	Условное `выражение`, контролирующее цикл. Действие будет циклическим до тех пор, проверяемое условие вычисляется как «*true*». Ход выполнения цикла МОЖЕТ БЫТЬ не определен полностью. Это означает, что разработчик модели может просто задокументировать условие, и цикл не сможет быть выполнен соответствующим образом.

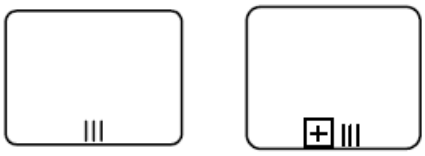
Характеристики многоэкземплярного цикла

Класс `MultiInstanceLoopCharacteristics` позволяет задавать необходимое количество *экземпляров* **Действия**. Экземпляры МОГУТ выполняться как параллельно, так и последовательно. Для указания или вычисления необходимого количества *экземпляров* используется выражение или выставление значения, основанного не имеющихся данных. В данном случае может быть указан способ ввода данных, посредством которого можно управлять набором данных. Количество элементов набора данных обусловливает количество *экземпляров* **Действия**. Такой ввод данных может быть использован благодаря задействованию **Ассоциации данных**. Разработчик модели может также настраивать *цикл* для осуществления контроля над полученными *токенами*.

- *Многоэкземплярный* Маркер, помещенный в фигуру **Задачи** или **Подпроцесса**, ДОЛЖЕН БЫТЬ выполнен в виде трех параллельных вертикальных линий.
  - Если *Многоэкземплярный* Маркер используется для параллельного (атрибут `isSequential` имеет значение «*false*»), а не последовательного, выполнения *экземпляров* **Действия**, линии должны отображаться вертикально (см. фигуру 10.48).
  - Если *Многоэкземплярный* Маркер используется для последовательного (атрибут `isSequential` имеет значение «*true*»), а не параллельного, выполнения *экземпляров* **Действия**, линии должны отображаться горизонтально (см. фигуру 10.49).
  - **Многоэкземплярный** Маркер МОЖЕТ сочетаться с Маркером **Компенсации**.



Фигура 10.48 –Задача и Подпроцесс с Многоэкземплярным маркером для параллельного выполнения



Фигура 10.49 – Задача и Подпроцесс с Многоэкземплярным маркером для последовательного выполнения

Элемент `MultiInstanceLoopCharacteristics` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) посредством связи с элементом `LoopCharacteristics`. Таблица 10.29 содержит информацию о дополнительных атрибутах и ассоциациях элемента `MultiInstanceLoopCharacteristics`.

Таблица 10.29 – Атрибуты и ассоциации элемента MultiInstanceLoopCharacteristics

Название атрибута	Описание/использование
isSequential: boolean = false	Данный атрибут является переключателем для

	осуществления контроля над тем, каким образом будут выполняться *экземпляры* **Действия**: последовательно или параллельно.
loopCardinality: Expression [0..1]	Данный атрибут представляет собой числовое `выражение` для осуществления контроля над количеством *экземпляров* **Действий**, которые будут сформированы. Данное выражение ДОЛЖНО вычислять **целое число**. Оно также МОЖЕТ БЫТЬ не полностью определенным, т.к. разработчик модели МОЖЕТ просто задокументировать условие, и цикл не сможет быть выполнен соответствующим образом. Для установки корректного значения *многоэкземплярности* ДОЛЖНО БЫТЬ указано выражение `loopCardinality` или значение `loopDataInput`.
loopDataInputRef: ItemAwareElement [0..1]	Данный атрибут используется для указания количества *экземпляров* **Действия:** один экземпляр приходится на один элемент набора данных, хранимого в элементе `ItemAwareElement`. Для **Задач** используется ссылка на ввод данных, являющийся частью атрибута `InputOutputSpecification` для **Действия**. Для **Подпроцессов** используется ссылка на **Объект данных**, содержащий информацию о наборе данных, в контексте, прозрачном для **Подпроцессов**. Для установки корректного значения *многоэкземплярности* ДОЛЖНО БЫТЬ указано `выражение loopCardinality` или значение `loopDataInput`.
loopDataOutputRef: ItemAwareElement [0..1]	Данный атрибут определяет набор данных, которые создаются посредством включения *многоэкземплярности* **Действий**. Для **Задач** используется ссылка на **Ввод Данных**, являющийся частью атрибута `InputOutputSpecification` для **Действия**. Для **Подпроцессов** используется ссылка на **Объект Данных**, содержащий информацию о наборе данных, в контексте, прозрачном для **Подпроцессов**.
inputDataItem: DataInput [0..1]	Данный атрибут используется для отображения индивидуального элемента **Набора Данных**, хранимого в `loopDataInput`, для каждого отдельно взятого *экземпляра* **Действия**. Данный атрибут может являться источником `DataInputAssociation (Ассоциации)` для **Ввода Данных** `InputOutputSpecification Действия`. Тип значения данного атрибута МОЖЕТ БЫТЬ скалярной величиной типа, определенного для атрибута `loopDataInput`.

outputDataItem: DataOutput [0..1]	Данный атрибут используется для отображения индивидуального элемента **Набора Данных**, хранимого в `loopDataOutput`, для каждого отдельно взятого экземпляра **Действия**. Данный атрибут может являться целью `DataOutputAssociation (Ассоциации)` для **Вывода Данных** `InputOutputSpecification` **Действия**. Тип данного атрибута МОЖЕТ БЫТЬ скалярной величиной типа, определенного для атрибута `loopDataOutput`.
behavior: MultiInstanceBehavior = all { None | One | All | Complex }	Данный атрибут используется в качестве ярлыка для определения того, когда от практически завершенного *экземпляра* **Действия** ДОЛЖНО БЫТЬ запущено **Событие**. Допускаются значения: «`None`», «`One`», «`All`» и «`Complex`». Им соответствует следующее поведение: • **«None»**. Элемент `EventDefinition`, связанный посредством ассоциации `noneEvent`, будет запущен для каждого завершающегося *экземпляра*. • **«One».** Элемент `EventDefinition`, на который ссылаются посредством ассоциации `oneEvent`, будет запущен для первого завершающегося *экземпляра*. • **«All».** Ни одно **Событие** запущено не будет. `Токен` образуется после завершения всех *экземпляров*. • **«Complex».** Элемент `complexBehaviorDefinitions` определяет, необходимо ли запускать **События**, и если да, то какие. Значения «`None`» и «`One`»указывают, что по умолчанию будет запущено `Событие Сигнал` (`SignalEventDefinition`), которое автоматически наделяется текущими атрибутами **Действия MI** (**Multi- Instance**). На любые **События**, выполнение которых было активировано, могут отреагировать **События**, присоединенные к границам **Многоэкземплярного Действия**.
complexBehaviorDefinition: ComplexBehaviorDefinition [0..*]	Данный атрибут контролирует то, когда и какое **Событие** будет запущено, если значение behavior равно «complex».
completionCondition: Expression [0..1]	Данный атрибут определяет условное `выражение`. В случае, если значение условного выражение равно «*true*», происходит отмена повторения *экземпляра* **Действия** и создается *токен*.
oneBehaviorEventRef: EventDefinition [0..1]	Если behavior имеет значение «`one`», а первый

	экземпляр внутреннего **Действия** завершается, то запускается `EventDefinition`.
noneBehaviorEventRef: EventDefinition [0..1]	Если behavior имеет значение «`none`», а первый *экземпляр* внутреннего **Действия** завершается, то запускается `EventDefinition`.

Таблица 10.30 содержит список атрибутов *экземпляра* **Действия**, доступных во время его выполнения. Во время выполнения каждому *экземпляру* **Многоэкземплярного Действия** (внешнему) полагается несколько сформированных *экземпляров* (внутренних) этого **Действия**.

Таблица 10.30 – Атрибуты экземпляра Многоэкземплярного Действия

Название атрибута	Описание/использование
loopCounter: integer	Данный атрибут используется для каждого сформированного (внутреннего) *экземпляра* **Действия**. Содержит данные о порядковых номерах этих *экземпляров*, т.е. если значение данного атрибута для *экземпляра* – это некое число (количественное числительное), то данному *экземпляру* присваивается порядковый номер, образованный от указанного числа.
numberOfInstances: integer	Данный атрибут используется только для внешних *экземпляров* **Многоэкземплярного Действия**. Содержит данные об общем количестве внутренних *экземпляров*, сформированных для **Многоэкземплярного Действия**.
numberOfActiveInstances: integer	Данный атрибут используется только для внешних экземпляров **Многоэкземплярного Действия**. Содержит данные о количестве активных в текущий момент внутренних *экземплярах* этого **Действия**. В случае, если *экземпляры* **Действия** выполняются последовательно, значение данного атрибута будет превышать «1». Если же *экземпляры* **Действия** выполняются параллельно, значение данного атрибута не может превышать значение, хранимое в `numberOfInstances`.
numberOfCompletedInstances: integer	Данный атрибут используется только для внешних *экземпляров* **Многоэкземплярного Действия**. Содержит данные о количестве уже выполненных внутренних *экземпляров* данного **Действия**.
numberOfTerminatedInstances: integer	Данный атрибут используется только для внешних *экземпляров* **Многоэкземплярного Действия**. Содержит данные о количестве прерванных внутренних *экземпляров* данного **Действия**. `numberOfInstances (общее число экземпляров)` является результатом сложения значений `numberOfTerminatedInstances (число прерванных экземпляров)`, `numberOfCompletedInstances (число`

	выполненных экземпляров) и numberOfActiveInstances (число активных экземпляров).

Элемент Complex Behavior Definition

Элемент ComplexBehaviorDefinition используется для осуществления контроля над временем запуска **Событий** и выбором **Событий** для запуска в случае, если для **Многоэкземплярного Действия** установленное значение behavior равно complex.

Данный элемент ComplexBehaviorDefinition наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5). Таблица 10.31 содержит информацию о дополнительных атрибутах и ассоциациях элемента ComplexBehaviorDefinition.

Таблица 10.31 – Атрибуты и ассоциации элемента StandardLoopCharacteristics

Название атрибута	Описание/использование
condition: Formal Expression	Данный атрибут представляет собой булевское выражение. Если значение определено как «*true*», то происходит отмена выполнения оставшихся *экземпляров* **Действия** и формируется *токен*.
event: ImplicitThrowEvent	Если условие верно (*true*), то определяется предназначенное для запуска **Событие**. На него отреагирует **Событие**, *находящееся на границе* **Многоэкземплярного Действия**.

10.2.9. Представление XML-схемы для Действий

Таблица 10.32 – XML-схема для элемента Activity

```
<xsd:element name="activity" type="tActivity"/>
<xsd:complexType name="tActivity" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tFlowNode">
            <xsd:sequence>
                <xsd:element ref="ioSpecification" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="dataInputAssociation" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="dataOutputAssociation" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="resourceRole" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="loopCharacteristics" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="isForCompensation" type="xsd:boolean" default="false"/>
            <xsd:attribute name="startQuantity" type="xsd:integer" default="1"/>
            <xsd:attribute name="completetionQuantity" type="xsd:integer" default="1"/>
            <xsd:attribute name="default" type="xsd:IDREF" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.33 – XML-схема для элемента AdHocSubProcess

```
<xsd:element name="adHocSubProcess" type="tAdHocSubProcess" substitutionGroup="flowElement"/>
<xsd:complexType name="tAdHocSubProcess">
    <xsd:complexContent>
        <xsd:extension base="tSubProcess">
            <xsd:sequence>
                <xsd:element name="completionCondition" type="tExpression" minOccurs="0"
                        maxOccurs="1"/>
```

```
                </xsd:sequence>
                <xsd:attribute name="cancelRemainingInstances" type="xsd:boolean" default="true"/>
                <xsd:attribute name="ordering" type="tAdHocOrdering" default="Parallel"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>


    <xsd:simpleType name="tAdHocOrdering">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Parallel"/>
            <xsd:enumeration value="Sequential"/>
        </xsd:restriction>
    </xsd:simpleType>
```

Таблица 10.34 – XML-схема для элемента BusinessRuleTask

```
    <xsd:element name="businessRuleTask" type="tBusinessRuleTask" substitutionGroup="flowElement"/>
    <xsd:complexType name="tBusinessRuleTask">
        <xsd:complexContent>
            <xsd:extension base="tTask">
                <xsd:attribute name="implementation" type="tImplementation" default="##unspecified"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.35 – XML-схема для элемента CallableElement

```
    <xsd:element name="callableElement" type="tCallableElement"/>
    <xsd:complexType name="tCallableElement">
        <xsd:complexContent>
            <xsd:extension base="tRootElement">
                <xsd:sequence>
                    <xsd:element name="supportedInterfaceRef" type="xsd:QName" minOccurs="0"
                            maxOccurs="unbounded"/>
                    <xsd:element ref="ioSpecification" minOccurs="0" maxOccurs="1"/>
                    <xsd:element ref="ioBinding" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.36 – XML-схема для элемента CallActivity

```
    <xsd:element name="callActivity" type="tCallActivity" substitutionGroup="flowElement"/>
    <xsd:complexType name="tCallActivity">
        <xsd:complexContent>
            <xsd:extension base="tActivity">
                <xsd:attribute name="calledElement" type="xsd:QName" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

Таблица 10.37 – XML-схема для элемента GlobalBusinessRuleTask

```
    <xsd:element name="globalBusinessRuleTask" type="tGlobalBusinessRuleTask" substitution-
            Group="rootElement"/>
    <xsd:complexType name="tGlobalBusinessRuleTask">
    <xsd:complexContent>
            <xsd:extension base="tGlobalTask">
                <xsd:attribute name="implementation" type="tImplementation" default="##unspecified"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

Таблица 10.38 – XML-схема для элемента GlobalScriptTask

```
<xsd:element name="globalScriptTask" type="tGlobalScriptTask" substitutionGroup="rootElement"/>
<xsd:complexType name="tGlobalScriptTask">
    <xsd:complexContent>
        <xsd:extension base="tGlobalTask">
            <xsd:sequence>
                <xsd:element ref="script" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="scriptLanguage" type="xsd:anyURI"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.39 – XML-схема для элемента GlobalTask

```
<xsd:element name="globalTask" type="tGlobalTask" substitutionGroup="rootElement"/>
<xsd:complexType name="tGlobalScriptTask">
    <xsd:complexContent>
        <xsd:extension base="tCallableElement">
            <xsd:sequence>
                <xsd:element ref="resourceRole" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.40 – XML-схема для элемента LoopCharacteristics

```
<xsd:element name="loopCharacteristics" type="tLoopCharacteristics"/>
<xsd:complexType name="tLoopCharacteristics" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.41 – XML-схема для элемента MultiInstanceLoopCharacteristics

```
<xsd:element name="multiInstanceLoopCharacteristics" type="tMultiInstanceLoopCharacteristics"
        substitutionGroup="loopCharacteristics"/>
<xsd:complexType name="tMultiInstanceLoopCharacteristics">
    <xsd:complexContent>
    <xsd:extension base="tLoopCharacteristics">
        <xsd:sequence>
            <xsd:element name="loopCardinality" type="tExpression" minOccurs="0"
                maxOccurs="1"/>
            <xsd:element name="loopDataInputRef" type="xsd:QName" minOccurs="0"
                maxOccurs="1"/>
            <xsd:element name="loopDataOutputRef" type="xsd:QName" minOccurs="0"
                maxOccurs="1"/>
            <xsd:element name="inputDataItem" type="tDataInput" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="outputDataItem" type="tDataOutput" minOccurs="0"
                maxOccurs="1"/>
            <xsd:element ref="complexBehaviorDefinition" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="completionCondition" type="tExpression" minOccurs="0"
                maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="isSequential" type="xsd:boolean" default="false"/>
        <xsd:attribute name="behavior" type="tMultiInstanceFlowCondition" default="All"/> <xsd:attribute
        name="oneBehaviorEventRef" type="xsd:QName" use="optional"/>
        <xsd:attribute name="noneBehaviorEventRef" type="xsd:QName" use="optional"/>
    </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tMultiInstanceFlowCondition">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="None"/>
        <xsd:enumeration value="One"/>
        <xsd:enumeration value="All"/>
        <xsd:enumeration value="Complex"/>
    </xsd:restriction>
</xsd:simpleType>
```

Таблица 10.42 – XML-схема для элемента ReceiveTask

```
<xsd:element name="receiveTask" type="tReceiveTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tReceiveTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:attribute name="implementation" type="tImplementation" default="##WebService"/>
            <xsd:attribute name="instantiate" type="xsd:boolean" default="false"/>
            <xsd:attribute name="messageRef" type="xsd:QName" use="optional"/>
            <xsd:attribute name="operationRef" type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.43 – XML-схема для элемента ResourceRole

```
<xsd:element name="resourceRole" type="tResourceRole"/>
<xsd:complexType name="tResourceRole">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:choice>
                <xsd:sequence>
                    <xsd:element name="resourceRef" type="xsd:QName" minOccurs="0"
                        maxOccurs="1"/>
                    <xsd:element ref="resourceParameterBinding" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:element ref="resourceAssignmentExpression" minOccurs="0" maxOccurs="1"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.44 – XML-схема для элемента ScriptTask

```
<xsd:element name="scriptTask" type="tScriptTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tScriptTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:sequence>
                <xsd:element ref="script" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="scriptFormat" type="xsd:anyURI"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="script" type="tScript"/>
<xsd:complexType name="tScript" mixed="true">
    <xsd:sequence>
        <xsd:any namespace="##any" processContents="lax" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

Таблица 10.45 – XML-схема для элемента SendTask

```
<xsd:element name="sendTask" type="tSendTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tSendTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:attribute name="implementation" type="tImplementation" default="##WebService"/>
            <xsd:attribute name="messageRef" type="xsd:QName" use="optional"/>
            <xsd:attribute name="operationRef" type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.46 – XML-схема для элемента ServiceTask

```
<xsd:element name="serviceTask" type="tServiceTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tServiceTask">
    <xsd:complexContent>
        <xsd:extension base="tTask">
            <xsd:attribute name="implementation" type="tImplementation" default="##WebService"/>
            <xsd:attribute name="operationRef" type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.47 – XML-схема для элемента StandardLoopCharacteristics

```
<xsd:element name="standardLoopCharacteristics" type="tStandardLoopCharacteristics"/>
<xsd:complexType name="tStandardLoopCharacteristics">
    <xsd:complexContent>
        <xsd:extension base="tLoopCharacteristics">
            <xsd:sequence>
                <xsd:element name="loopCondition" type="tExpression" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="testBefore" type="xsd:boolean" default="false"/>
            <xsd:attribute name="loopMaximum" type="xsd:integer" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.48 – XML-схема для элемента SubProcess

```
<xsd:element name="subProcess" type="tSubProcess" substitutionGroup="flowElement"/>
<xsd:complexType name="tSubProcess">
    <xsd:complexContent>
        <xsd:extension base="tActivity">
            <xsd:sequence>
                <xsd:element ref="laneSet" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="flowElement" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="artifact" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="triggeredByEvent" type="xsd:boolean" default="false"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.49 – XML-схема для элемента Task

```
<xsd:element name="task" type="tTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tTask">
    <xsd:complexContent>
        <xsd:extension base="tActivity"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.50– XML-схема для элемента Transaction

```
<xsd:element name="transaction" type="tTransaction" substitutionGroup="flowElement"/>
<xsd:complexType name="tTransaction">
    <xsd:complexContent>
        <xsd:extension base="tSubProcess">
            <xsd:attribute name="method" type="tTransactionMethod" default="Compensate"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tTransactionMethod">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Compensate"/>
        <xsd:enumeration value="Image"/>
        <xsd:enumeration value="Store"/>
    </xsd:restriction>
</xsd:simpleType>
```

10.3. Компоненты и Данные

Традиционным требованием к моделированию **Процессов** является возможность моделирования компонентов (физических или информационных), которые создаются, управляются и используются в ходе выполнения **Процесса**. Важным аспектом этого требования является возможность сбора введенных данных, а также запроса этих данных и управления ими.

В **BPMN** не представлено встроенных образцов описания структуры данных или языка `выражений` для запроса данных. Вместо этого, нотация позволяет использовать любые другие (внешние) структуры данных и языки `выражений`**. BPMN** также допускает совместное использование нескольких языков описания физической структуры баз данных и языков `выражений` в одной модели **Процесса**. Описание их совместимости и проверки не входят в данную спецификацию, - за это ответственны производители инструментов моделирования.

XML Schema и XPath определены в **BPMN** как язык описания физической структуры базы данных и язык `выражений` (соответственно) по умолчанию. Однако производители инструментов моделирования в праве заменять их на собственные языки.

10.3.1. Моделирование данных

Традиционным требованием к моделированию **Процессов** является возможность моделирования элементов (физических или информационных), которые создаются, управляются и используются в ходе выполнения **Процесса**.

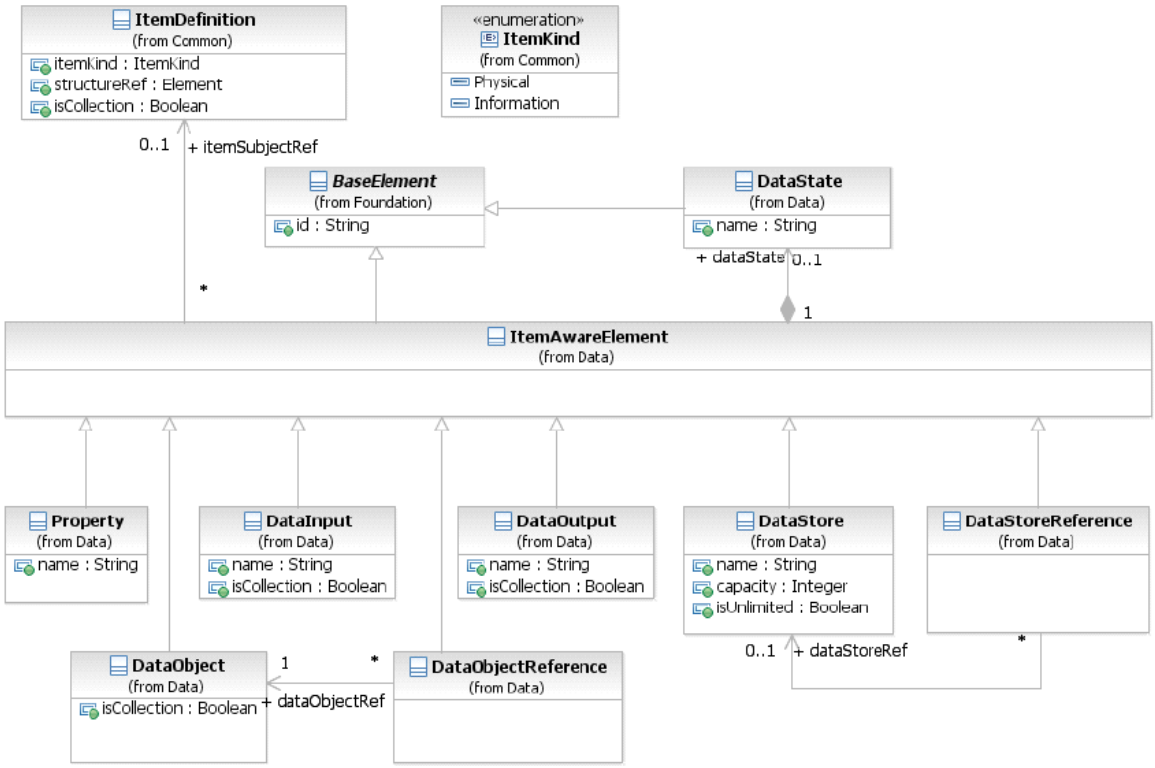
BPMN предлагает реализацию этого требования посредством использования различных структурных компонентов: **Объектов данных**, атрибутов *ItemDefinition*, *Properties*, *Data Inputs*, *Data Outputs*, **Сообщений**, *Input Sets*, *Output Sets*, а также **Ассоциаций данных**.

Элементы, связанные с компонентами

BPMN выделяет несколько элементов, предназначенных для хранения и передачи компонентов в ходе выполнения **Процесса**. Обычно такие элементы относят к «связанным с компонентами» (item-aware). Они схожи с переменными, являющимися общими для многих языков. Как и переменные, данные элементы имеют атрибуты класса `ItemDefinition`.

Структура данных, которую содержат эти элементы, определяется посредством `ItemDefinition`. Элемент `ItemAwareElement` МОЖЕТ БЫТЬ не указан. Это означает, что атрибут `структуры ItemDefinition` является опциональным в том случае, если разработчик модели не желает указывать структуру связанных данных.

Таким образом, элементами, связанными с компонентами, являются: **Объекты данных**, **Ссылки на Объекты Данных**, **Хранилища данных**, `Properties, DataInputs и DataOutputs.`



Фигура 10.50 – Диаграмма классов элемента ItemAwareElement

Элемент `ItemAwareElement` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.51 содержит информацию о дополнительных ассоциациях элемента `ItemAwareElement`.

Таблица 10.51 – Ассоциации элемента ItemAwareElement

Название атрибута	Описание/использование
itemSubjectRef: ItemDefinition [0..1]	Характеристика компонентов, хранящихся в элементе `ItemAwareElement` или передающихся им.
dataState: DataState [0..1]	Ссылка на атрибут `DataState`, используемый для определения статусов данных, хранящихся в компоненте.

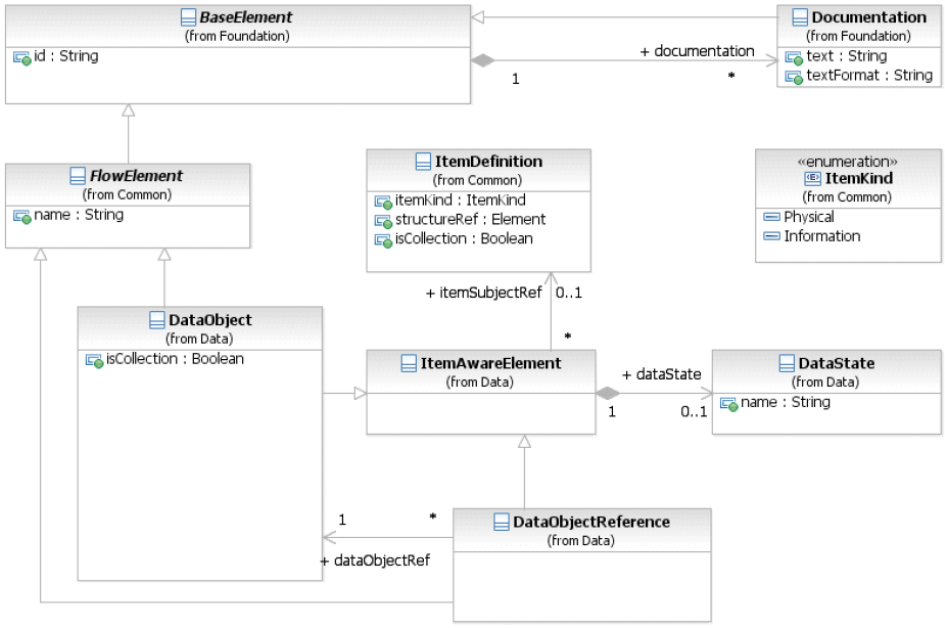
Объекты данных

Данные могут быть внедрены в модель **Процесса,** прежде всего, посредством элемента `DataObject`. Данный элемент имеет четкий жизненный цикл, результатом которого является ограничение на доступ.

Элемент DataObject

Элементы класса **DataObject** содержат информацию о наборе данных и ДОЛЖНЫ находиться внутри графического элемента **Процесс**/**Подпроцесс**. Графически они отображаются на диаграмме **Процесса**. **Ссылки на Объекты** `данных` предназначены для многократного использования **Объектов данных** в рамках одной и той же диаграммы. Они могут указывать на различные `состояния (states)` одних и тех же **Объектов данных** на разных отрезках **Процесса. Ссылка на Объект данных** не может указывать на характеристики компонентов, а **Объект данных** – на статусы. Название **Ссылки на Объект данных** формируется посредством объединения названия соответствующего **Объекта данных** и статуса, хранящегося в

Ссылке на Объект данных (последнее заключено в квадратные скобки): <Data Object Name> [<Data Object Reference State>].



Фигура 10.51 – Диаграмма элементов класса DataObject

Элемент **DataObject** наследует атрибуты и ассоциации элементов `FlowElement` (см. таблицу 8.44) и `ItemAwareElement` (см. таблицу 10.52). Таблица 10.52 содержит информацию о дополнительных атрибутах элемента **DataObject**.

Таблица 10.52 – Атрибуты элемента DataObject

Название атрибута	Описание/использование
isCollection: boolean = false	Необходим при отсутствии ссылки на элемент `itemDefinition` и указывает на то, будет ли **Объект данных** состоять из ряда элементов. В случае, если ссылка на `itemDefinition` все же отсутствует, то атрибут `isCollection` ДОЛЖЕН иметь такое же значение, какое установлено для атрибута `isCollection` элемента `itemDefinition`. Значением по умолчанию является «*false*».

Элемент **DataObjectReference** наследует атрибуты и ассоциации элементов `ItemAwareElement` (см. таблицу 10.52) и `FlowElement` (см. таблицу 8.44). Таблица 10.53 содержит информацию о дополнительных атрибутах элемента **DataObjectReference**.

Таблица 10.53 – Атрибуты и ассоциации элемента DataObjectReference

Название атрибута	Описание/использование
dataObjectRef: DataObject	Используется для определения **Объекта данных**, на который ссылается данная **Ссылка на Объект данных.**

Состояние данных

Элементы **DataObject** по желанию могут ссылаться на элемент `DataState`, который указывает на состояние набора данных, хранимого в **Объекте данных** (см. пример использования элемента `DataState` для **Объекта данных** на фигуре 7.8). Данная спецификация не включает информацию по определению таких состояний (например, возможных значений) и специфической семантики конструкций. Тем не менее, пользователи **BPMN** могут задействовать такие состояния наборов данных и способность нотации к расширению для их определения.

Элемент `DataState` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.54 содержит информацию о дополнительных атрибутах элемента **DataObject.**

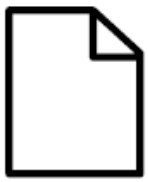
Таблица 10.54 – Атрибуты и ассоциации элемента DataState

Название атрибута	Описание/использование
name: string	Используется для определения названия элемента `DataState`.

Объект данных как коллекция данных

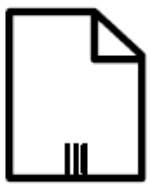
Объекты данных, которые ссылаются на элемент `ItemDefinition`, выступающий в качестве *коллекции* (*collection*), должны отображаться по-разному в зависимости от количества *экземпляров* структур данных. Графически они могут отображаться следующим образом:

Один *экземпляр* (фигура 10.52)



Фигура 10.52 – Объект данных

Коллекция (фигура 10.53)



Фигура 10.53 – Объект данных, выступающий в качестве коллекции данных

Графическое отображение Объекта данных

На диаграмме **Процесса Объект данных** может возникать многократно. Каждый такой элемент ссылается на такой же *экземпляр* **Объекта данных**. Многократное использование **Объекта данных** предусмотрено для упрощения отображения связей на диаграмме.

Жизненный цикл и доступность

Жизненный цикл **Объекта данных** связан с жизненным циклом родительского **Процесса**/**Подпроцесса**. При запуске *экземпляра* **Процесса**/**Подпроцесса Объекты данных**, заключенные в нем, становятся активны. При отмене выполнения данного *экземпляра* **Процесса**/**Подпроцесса** все находящиеся в нем *экземпляры*

Объектов данных становятся неактивны. В данном случае данные, которые содержатся в этих *экземплярах*, перестают быть доступными.

Доступность **Объекта данных** обусловливается его жизненным циклом. Данные, содержащиеся в **Объекте данных**, могут быть доступны лишь тогда, когда его *экземпляр* гарантированно активен в текущий момент времени. Таким образом, содержимое **Объекта данных** доступно лишь непосредственно для элементов потока родительского **Процесса/Подпроцесса** или родственных ему **Процессов/Подпроцессов** и их дочерних включений (включая **Ссылку на Объект данных**, ссылающуюся на этот **Объект данных**).

Рассмотрим следующую схему:

Process A
- Data object 1
- Task A
- Sub-process A
  - Data object 2
  - Task B
- Sub-process B
  - Data object 3
  - Sub-process C
    - Data object 4
    - Task C
  - Task D

Элемент **Data object 1** может быть доступен для **Process A**, **Task A**, **Sub-Process A**, **Task B**, **Sub-Process B**, **Sub-Process C**, **Task C** и **Task D**.
Элемент **Data object 2** может быть доступен для **Sub-Process A** и **Task B**.
Элемент **Data object 3** может быть доступен для **Sub-Process B**, **Sub-Process C**, **Task C** и **Task D**.
Элемент **Data object 4** может быть доступен для **Sub-Process C** и **Task C**.

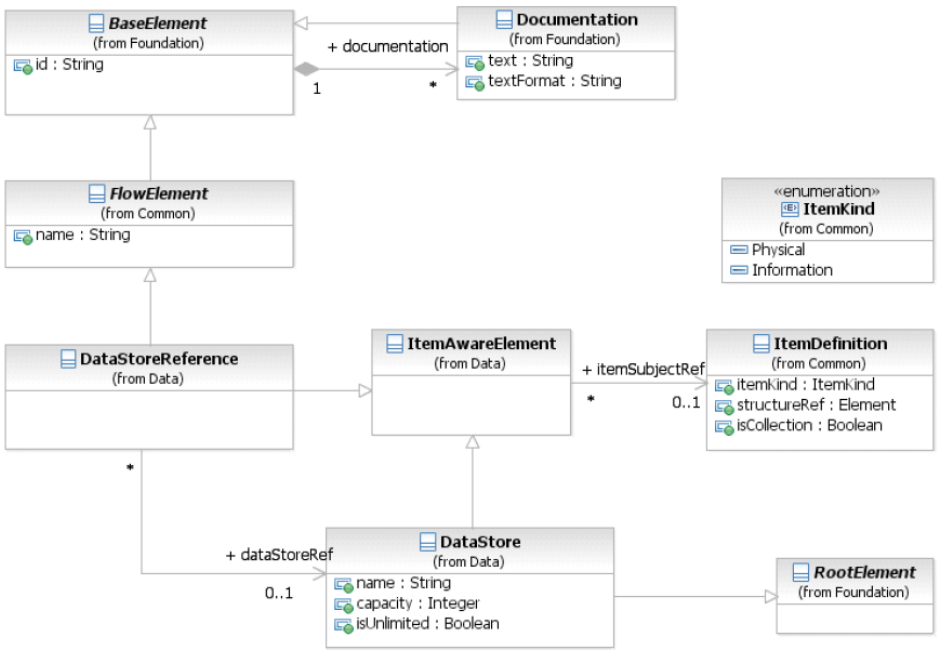
Хранилища данных

Использование элемента `Хранилище данных` позволяет **Действиям** отыскивать и обновлять хранимую в **Процессе** информацию, попадающую за его пределы. Один и тот же этот элемент может многократно появляться на диаграмме **Процесса** в виде **Ссылок на Хранилище данных**.

Ссылка на Хранилище данных является элементом `ItemAwareElement` и поэтому может служить и источником, и целью **Ассоциации данных**. При поступлении информации в **Ссылку на хранилище данных** или выходе её из этого графического элемента данная информация также поступает в соответствующее `Хранилище данных` и выходит из него.

Графически Хранилище данных отображается следующим образом (см. фигуру 10.54):

Фигура 10.54 – Графический элемент Хранилище данных



Фигура 10.55 – Диаграмма классов элемента DataStore

Элемент `DataStore` наследует атрибуты и ассоциации элемента `FlowElement` (см. таблицу 8.44) посредством его связи с элементами `RootElement` и `ItemAwareElement` (см. таблицу 10.51). Таблица 10.55 содержит информацию о дополнительных атрибутах элемента `DataState`.

Таблица 10.55 – Атрибуты элемента DataState

Название атрибута	Описание/использование
name: string	Представляет собой описательное имя элемента.
capacity: integer [0..1]	Данный атрибут определяет `вместимость` **Хранилища данных**. Не используется, если значение атрибута `isUnlimited` установлено как «*true*».
isUnlimited: boolean = false	Если значение атрибута `isUnlimited` установлено как «*true*», то `вместимость` **Хранилища данных** является неограниченной и отменяет любое из значений, установленных для атрибута `capacity`.

Элемент **DataStoreReference** наследует атрибуты и ассоциации элементов `FlowElement` (см. таблицу 8.44) и `ItemAwareElement` (см. таблицу 10.51). Таблица 10.56 содержит информацию о дополнительных ассоциациях элемента `DataStoreReference`.

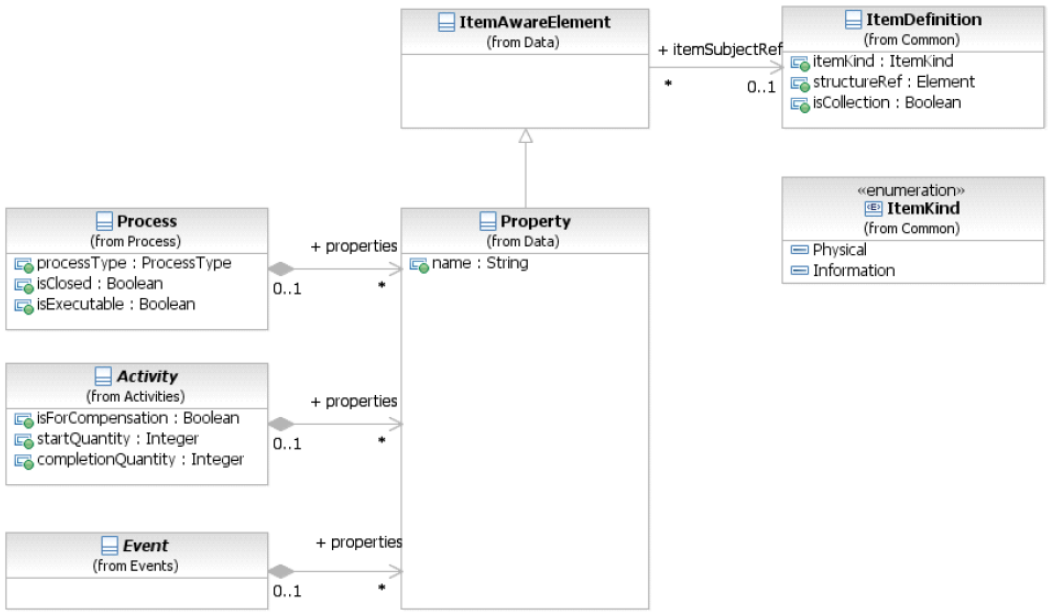
Таблица 10.56 – Атрибуты элемента DataStore

Название атрибута	Описание/использование
dataStoreRef: DataStore	Данный атрибут предоставляет ссылку на глобальный элемент `DataStore`.

Свойства

Свойства, как и **Объекты данных**, содержат информацию о наборе данных. Однако, в отличие от **Объекта данных**, на диаграмме **Процесса они** не отображаются. Лишь некоторые *Элементы Потока* МОГУТ содержать `Свойства`, а именно: **Процессы**, **Действия** и **События**.

Элементы класса `Property` являются элементом `Dataelement`, который выступает в роли контейнера для данных, относящихся к `Элементам Потока`. Они ДОЛЖНЫ храниться в элементах `FlowElement`. На диаграмме **Процесса** не отображаются.



Фигура 10.56 – Диаграмма классов элемента Property

Элемент `Property` наследует атрибуты и ассоциации элемента `ItemAwareElement` (см. таблицу 10.51). Таблица 10.57 содержит информацию о дополнительных атрибутах элемента `Property`.

Таблица 10.57 – Атрибуты элемента Property

Название атрибута	Описание/использование
name: string	Указывает имя элемента.

Жизненный цикл и доступность

Жизненный цикл элемента `Property` связан с жизненным циклом родительского элемента `Flow Element`. При запуске родительского элемента `Свойства`, хранящиеся в нем, становятся активны. При отмене выполнения *экземпляра* родительского элемента все находящиеся в нем *экземпляры* элемента `Property` становятся неактивны. В данном случае данные, которые содержатся в Свойствах, перестают быть доступными.

Доступность Свойства обусловливается его жизненным циклом. Данные, содержащиеся в элементе Property, могут быть доступны лишь тогда, когда его *Экземпляр* гарантированно активен в текущий момент времени. Таким образом, содержимое Свойства доступно лишь непосредственно для родительских **Процесса, Подпроцесса**, *Элементов Потока*. В случае, если родителем для данного элемента является **Процесс** или **Подпроцесс**, то Свойство становится доступно и для прямых дочерних включений этого **Процесса/Подпроцесса** (включая дочерние элементы).

Рассмотрим следующую схему:

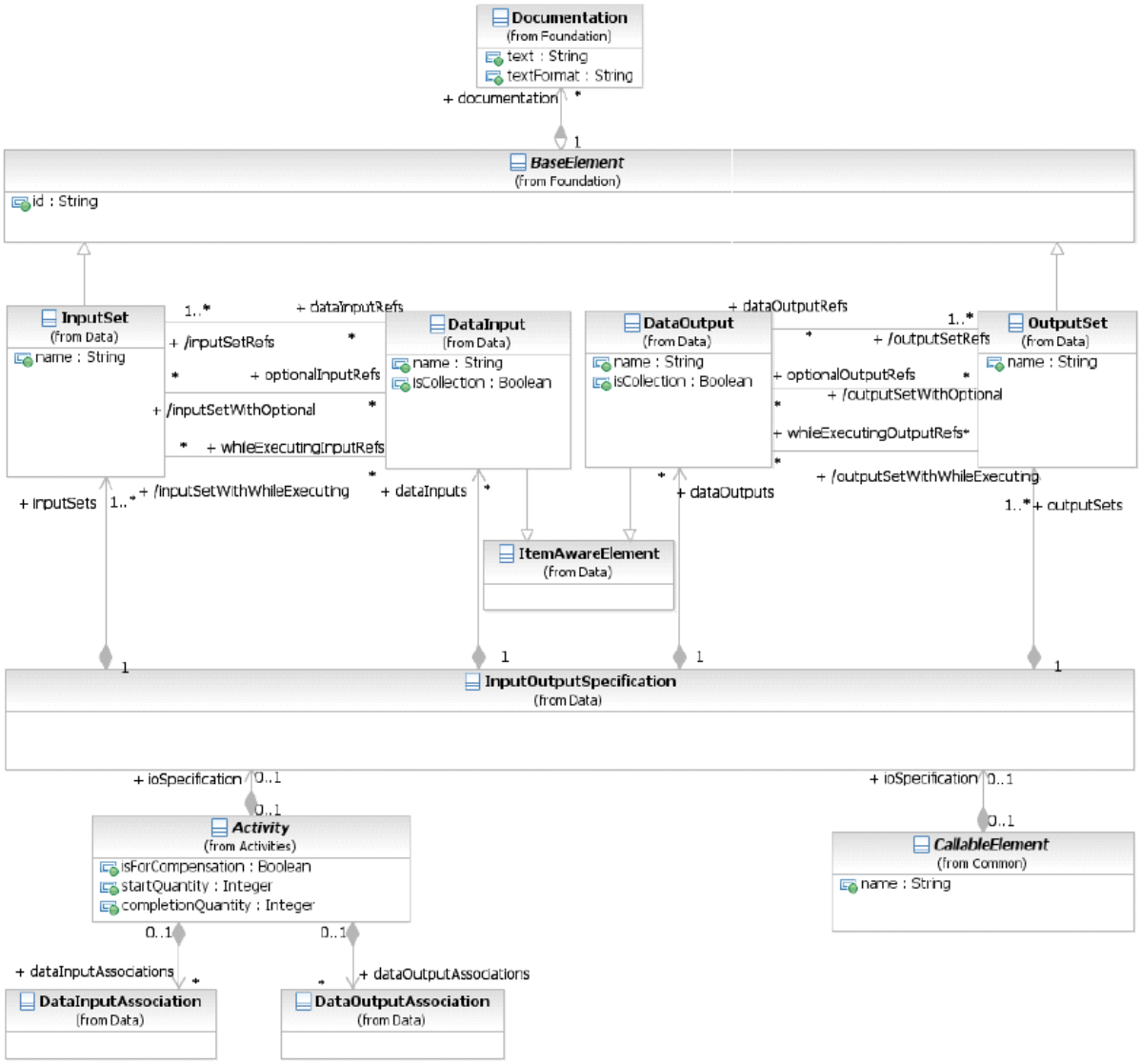
Process A
    Task A
    Sub-Process A
        Task B
    Sub-Process B
        Sub-Process C
            Task C
        Task D

`Свойства` **Process A** могут быть доступны для **Process A**, **Task A**, **Sub-Process A**, **Task B**, **Sub-Process B**, **Sub-Process C**, **Task C** и **Task D**.
`Свойства` **Sub-Process A** могут быть доступны для **Sub-Process A** и **Task B**.
`Свойства` **Task C** могут быть доступны для **Task C**.

Входные и Выходные данные

Для выполнения **Действия** или **Процесса** обычно необходимые определенные данные. **Действия** и **Процессы** могут сами формировать наборы данных в ходе их выполнения или в конце как результат выполнения. Требования к данным отображаются в виде **Входных данных** и элемента `InputSets`. Те данные, которые формируются в ходе или по окончании выполнения **Действия** или **Процесса**, отображаются посредством **Выходных данных** и элемента `OutputSets`. Вышеуказанные элементы входят в состав класса `InputOutputSpecification`.

Некоторые **Действия** и элементы `CallableElements` содержат элемент `InputOutputSpecification`, используемый для описания требований к содержащимся в них данным. Для элемента `InputOutputSpecification` определена семантика исполнения, которая полностью применима ко всем расширяющим его элементам. Однако входы и выходы определяются не для любого типа **Действия.** Они МОГУТ быть указаны лишь для **Задач** и элементов `CallableElements` (**глобальная Задача, Процесс**). Встроенный **Подпроцесс** НЕ ДОЛЖЕН напрямую определять **Входные** и **Выходные данные**, однако, он может косвенно указывать их посредством элемента `MultiInstanceLoopCharacteristics`.



Фигура 10.57 – Диаграмма классов элемента InputOutputSpecification

Элемент `InputOutputSpecification` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.58 содержит информацию о дополнительных атрибутах и ассоциациях элемента `InputOutputSpecification`.

Таблица 10.58 – Атрибуты и ассоциации элемента InputOutputSpecification

Название атрибута	Описание/использование
inputSets: InputSet [1..*]	Используется для ссылки на элемент `InputSets`, определенный элементом `InputOutputSpecification`. Любой элемент `InputOutputSpecification` ДОЛЖЕН определять по-меньшей мере один `InputSet`.
outputSets: OutputSet [1..*]	Используется для ссылки на элемент `OutputSets`, определенный элементом `InputOutputSpecification`. Любой элемент `Data Interface` ДОЛЖЕН определять по-меньшей мере один `OutputSet`.

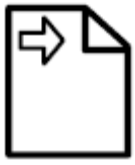
dataInputs: DataInput [0..*]	Используется для ссылки (по желанию) на **Входные данные**, определенные элементом `InputOutputSpecification`. В случае, если элемент `InputOutputSpecification` не определяет никакие **Входные данные**, это означает, то для запуска **Действия** такие данные НЕ ОБЯЗАТЕЛЬНЫ. Является упорядоченным набором данных.
dataOutputs: DataOutput [0..*]	Используется для ссылки (по желанию) на **Выходные данные**, определенные элементом `InputOutputSpecification`. В случае, если элемент `InputOutputSpecification` не определяет никакие **Выходные данные**, это означает, то для завершения выполнения **Действия** такие данные НЕ ОБЯЗАТЕЛЬНЫ. Является упорядоченным набором данных.

Входные данные

Посредством элемента **Входные Данные** указывается то, что в качестве входа элемента `InputOutputSpecification` будут использованы какие-то данные. С элементом `InputOutputSpecification` может быть ассоциировано любое количество **Входных Данных**.

Элемент **Входные данные** содержит информацию о наборе данных. На диаграмме **Процесса** он появляется для того, чтобы отобразить входы высокоуровневого или вызываемого **Процесса** (например, такого, на которое ссылается **Действие Вызов**, используемое для отображения вызываемого **Процесса** из вызывающего **Процесса**).

- Графически **Входные данные** отображается так же, как и **Объект данных**. Исключением является то, что графический элемент **Входные данные** ДОЛЖЕН иметь маркер, выполненный в виде фигурной стрелки без заливки (см. фигуру 10.58).
- Элемент **Входные данные** МОЖЕТ иметь *исходящую* **Ассоциацию данных**, но такое использование данного графического элемента подразумевает следующее:
  - Если **Входные данные** находятся непосредственно в высокоуровневом **Процессе,** то они НЕ ДОЛЖНЫ являться целью **Ассоциации данных** данной модели. Лишь **Входные данные**, содержащиеся в **Действиях** или **Событиях**, МОГУТ являться целью **Ассоциации данных**.
  - Если **Процесс** вызывается **Действием Вызов**, то **Ассоциации данных**, целью которых являются **Входные данные Действия Вызов** данной модели, МОГУТ БЫТЬ отображены соединенными с соответствующими **Входными данными** вызываемого **Процесса** и пересекать границы **Действия Вызов**. Однако помните, что вышеприведенный вариант совместного использования **Входных данных** и **Ассоциации данных** - всего лишь визуальное отображение. В модели **Процесса** целью **Ассоциаций данных** являются **Входные данные Действия Вызов**, а не вызываемого **Процесса**.



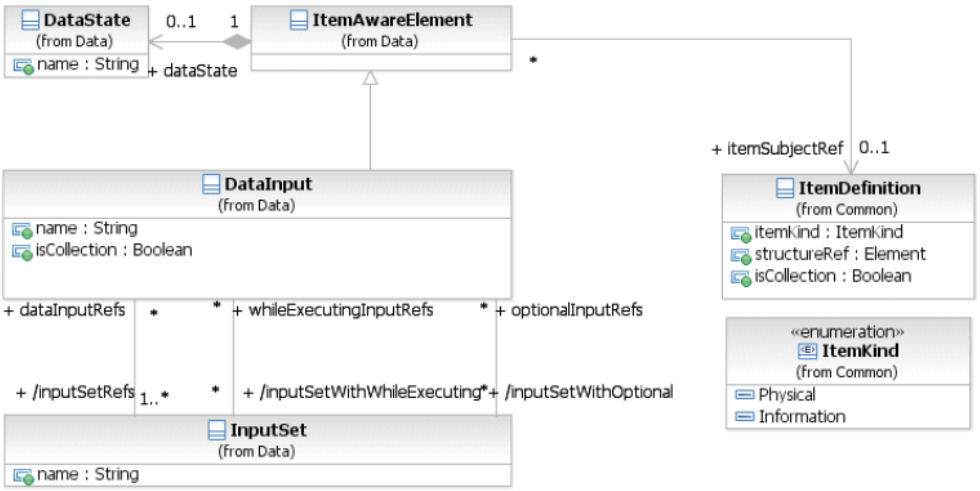
Фигура 10.58 – Графический элемент Входные данные

Атрибут **optional** указывает, является ли значение **DataInput** верным, даже при состоянии данных, равном «unavailable». Значением по умолчанию является «*false*». В случае, если значение атрибута равно «*true*», то

выполнение **Действия** не будет производиться до тех пор, пока посредством соответствующей **Ассоциации данных** не будет определено значение для элемента **DataInput**.

Состояние данных

Элементы **DataInput** по желанию могут ссылаться на элемент `DataState`, который используется для указания состояния данных, содержащихся в элементах **DataInput**. Данная спецификация не включает информацию по определению таких состояний (например, возможных значений) и специфической семантики конструкций. Тем не менее, для определения состояний пользователи **BPMN** могут задействовать элемент `DataState` и способность нотации к расширению.



Фигура 10.59 – Диаграмма классов элемента DataInput

Элемент **DataInput** наследует атрибуты и ассоциации элементов `BaseElement` (см. таблицу 8.5) и `ItemAwareElement` (см. таблицу 10.52). Таблица 10.59 содержит информацию о дополнительных атрибутах и ассоциациях элемента **DataInput**.

Таблица 10.59 – Атрибуты и ассоциации элемента DataInput

Название атрибута	Описание/использование
name: string [0..1]	Представляет собой описательное имя элемента.
inputSetRefs: InputSet [1..*]	Указывает, для какого количества элементов (одного или более) `InputSets` используется элемент `DataInput`. Данный атрибут порождается элементом `InputSets`.
inputSetwithOptional: InputSet [0..*]	Для каждого элемента `InputSet`, использующего элемент `DataInput`, можно указать то, может ли выполнение **Действия** начаться, если состояние `DataInput` равно «**unavailable**» (т.е. входные данные недоступны). Данный атрибут предоставляет список таких состояний.
inputSetWithWhileExecuting: Inputset [0..*]	Для каждого элемента `InputSet`, использующего элемент `DataInput`, можно указать то, может ли **Действие** во время выполнения обработать входные данные.

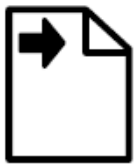
	Данный атрибут предоставляет список `InputSets`.
isCollection: boolean = false	Данный атрибут определяет, является ли **DataInput** набором элементом. Необходим тогда, когда нет ни одной ссылки на элемент `itemDefinition`. В случае, если ссылка на `itemDefinition` все же существует, то данный атрибут ДОЛЖЕН иметь то же значение, что и атрибут `isCollection` элемента `itemDefinition`, на который он ссылается. Значением по умолчанию является «*false*».

Выходные данные

Посредством элемента **Выходные Данные** указывается то, что в качестве выхода элемента `InputOutputSpecification` будут использованы какие-то данные. С элементом `InputOutputSpecification` МОЖЕТ БЫТЬ ассоциировано любое количество **Выходных Данных**.

Элемент **Выходные данные** содержит информацию о наборе данных. На диаграмме высокоуровневого **Процесса** он появляется для того, чтобы отобразить выходы данного **Процесса** (например, такого, на которое ссылается **Действие Вызов**, используемое для отображения вызываемого **Процесса** из вызывающего **Процесса**).

- Графически **Выходные данные** отображается так же, как и **Объект данных**. Исключением является то, что графический элемент **Выходные данные** ДОЛЖЕН иметь маркер, выполненный в виде небольшой фигурной стрелки с заливкой (см. фигуру 10.60).
- Элемент **Выходные данные** МОЖЕТ иметь *исходящую* **Ассоциацию данных**, но такое использование данного графического элемента подразумевает следующее:
  - Если **Выходные данные** находятся непосредственно в высокоуровневом **Процессе**, то они НЕ ДОЛЖНЫ являться источником **Ассоциации данных** данной модели. Лишь **Выходные данные**, содержащиеся в **Действиях** или **Событиях**, МОГУТ являться источником **Ассоциации данных**.
  - Если **Процесс** вызывается **Действием Вызов**, то **Ассоциации данных**, источником которых являются **Выходные данные Действия Вызов** данной модели, МОГУТ БЫТЬ отображены соединенными с соответствующими **Выходными данными** вызываемого **Процесса** и пересекать границы **Действия Вызов**. Однако помните, что вышеприведенный вариант совместного использования **Выходных данных** и **Ассоциации данных** - всего лишь визуальное отображение. В модели **Процесса** источником **Ассоциаций данных** являются **Выходные данные Действия Вызов**, а не вызываемого **Процесса**.

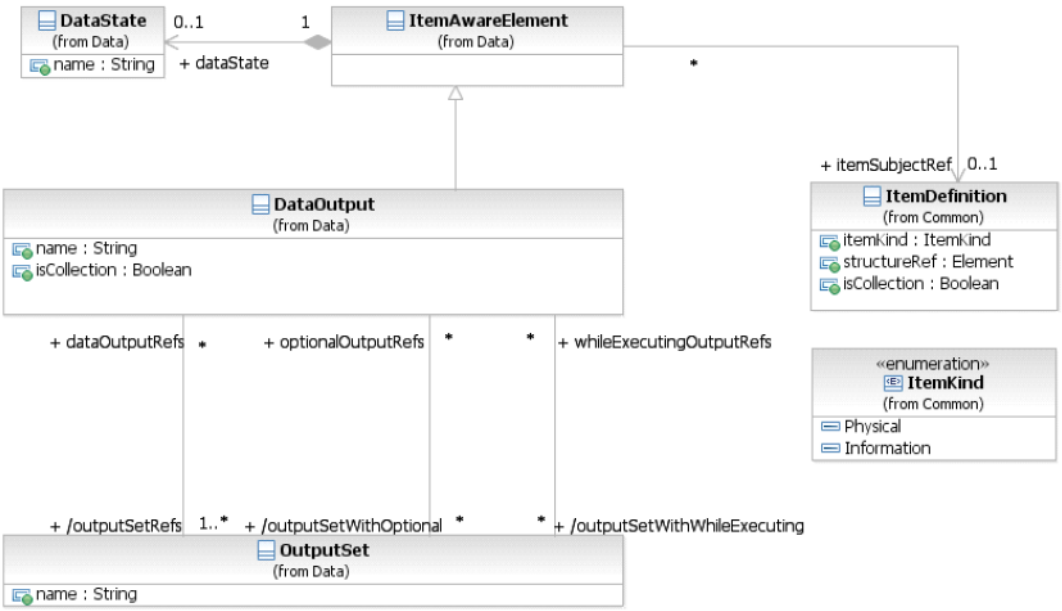


Фигура 10.60 – Графический элемент Выходные данные

Состояние данных

Элементы **DataOutput** по желанию могут ссылаться на элемент `DataState`, который используется для указания состояния данных, содержащихся в элементах **DataOutput**. Данная спецификация не включает информацию по определению таких состояний (например, возможных значений) и специфической семантики конструкций. Тем не

менее, для определения состояний пользователи **BPMN** могут задействовать элемент `DataState` и способность нотации к расширению.



Фигура 10.61 – Диаграмма классов элемента DataOutput

Элемент `DataOutput` наследует атрибуты и ассоциации элементов `BaseElement` (см. таблицу 8.5) и `ItemAwareElement` (см. таблицу 10.52). Таблица 10.60 содержит информацию о дополнительных атрибутах и ассоциациях элемента `DataOutput`.

Таблица 10.60 – Атрибуты и ассоциации элемента DataOutput

Название атрибута	Описание/использование
name: string [0..1]	Представляет собой описательное имя элемента.
outputSetRefs: OutputSet [1..*]	Указывает, для какого количества элементов (одного или более) `OutputSets` используется элемент `DataOutput`. Данный атрибут порождается элементом `OutputSets`.
outputSetwithOptional: Output- Set [0..*]	Для каждого элемента `Outputset`, использующего элемент `Dataoutput`, можно указать то, может ли **Действие** быть завершено без формирования на `Выходе Данных`. Данный атрибут предоставляет список таких значений.
outputSetWithWhileExecuting: OutputSet [0..*]	Для каждого элемента `OutputSet`, использующего элемент **DataOutput**, можно указать то, может ли **Действие** во время выполнения сформировать `выходные данные`. Данный атрибут предоставляет список таких значений.
isCollection: boolean = false	Данный атрибут определяет, является ли **DataOutput** набором элементом. Необходим тогда, когда нет ни одной ссылки на элемент `itemDefinition`. В случае, если ссылка на

	`itemDefinition` все же существует, то данный атрибут ДОЛЖЕН иметь то же значение, что и атрибут `isCollection` элемента `itemDefinition`, на который он ссылается. Значением по умолчанию является «*false*».

Ниже приведено описание соответствия входных и выходных данных **Действиям** и **Событиям**.

Соответствие Задаче Сервис

В случае связи **Задачи Сервис** с `Операцией`, **Задача** ДОЛЖНА иметь **Сообщение о входных данных**, а также элемент `itemDefinition`, значение которого равно значению атрибута `inMessageRef` данной операции, указанному в этом **Сообщении**. Если Операция предполагает использование на выходе **Сообщения**, у **Задачи** ДОЛЖНЫ БЫТЬ **Выход для Данных**, а также элемент `itemDefinition`, значение которого равно значению атрибута `outMessageRef` данной `операции`, указанному в этом **Сообщении**.

Соответствие Задаче Отправка сообщений

В случае связи **Задачи Отправка сообщений** с **Сообщением**, **Задача** ДОЛЖНА иметь максимум один набор **Входных Данных**, а также не более одного входа для данных. Если **Входные Данные** имеются**, Задача** ДОЛЖНА иметь элемент `itemDefinition`, значение которого равно значению, указанному в этом **Сообщении**. Если **Входные Данные** отсутствуют, во время выполнения **Задачи** данные в **Сообщении** не появятся.

Соответствие Задаче Получение сообщений

В случае связи **Задачи Получение сообщений** с **Сообщением**, **Задача** ДОЛЖНА иметь максимум один набор **Выходных Данных**, а также не более одного выхода для данных. Если **Выходные Данные** имеются, **Задача** ДОЛЖНА иметь элемент `itemDefinition`, значение которого равно значению, указанному в этом **Сообщении**. Если **Выходные Данные** отсутствуют, то данные из **Сообщения** не попадут из **Задачи Получение сообщений** в **Процесс** (т.е. в принципе не покинут **Задачу**).

Соответствие Пользовательской Задаче

Пользовательская задача имеет доступ к **Входным** и **Выходным Данным**, а также к элементам, содержащим информацию о наборе данных, которые доступны в рамках данной **Задачи**.

Соответствие Действию Вызов

`Входные` и `Выходные Данные` **Действия Вызов** привязаны к соответствующим элементам объекта `CallableElement`, однако, без использования `Ассоциации Данных`.

Соответствие Задаче Сценарий

Задача Сценарий имеет доступ к **Входным** и **Выходным Данным**, а также к элементам, содержащим информацию о наборе данных, которые доступны в рамках данной **Задачи**.

Events События

Если любой из элементов `EventDefinitions` какого-либо **События** ассоциирован с элементом, имеющим значение `ItemDefinition` (`Message, Escalation, Error` или `Signal`), должны быть приняты во внимание следующие ограничения:

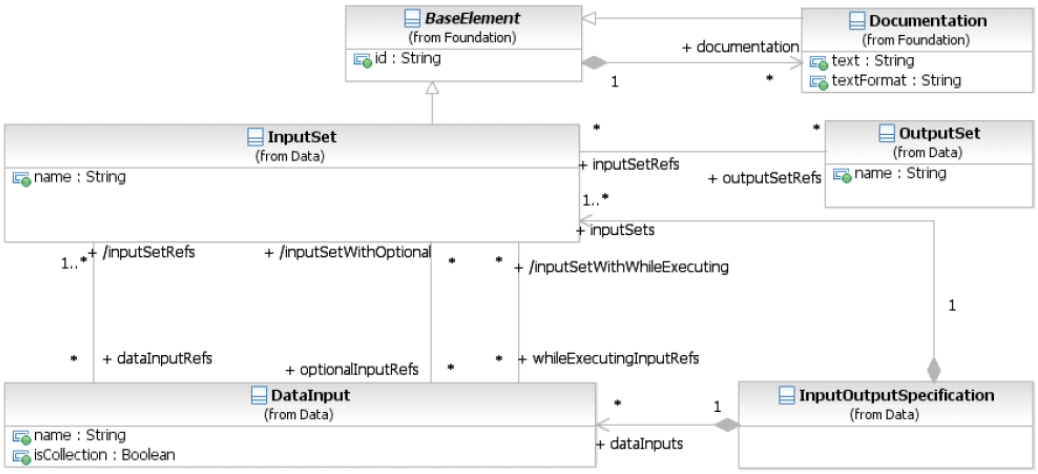
- В случае, если **Событие** ассоциировано с множеством элементов `EventDefinitions`, то для каждого из этих элементов ДОЛЖНЫ использоваться один **Вход Данных** (для *влияющих на ход потока* **Событий**) или один **Выход Данных** (для *реагирующих на триггер* **Событий**). Последовательность элементов `EventDefinitions` и **Входов/Выходов Данных** определяет, какому из элементов какой вход/выход соответствует.
- Для каждой пары `EventDefinition`+**Входные/Выходные Данные**, при наличии таких данных, ДОЛЖНО БЫТЬ указано значение элемента `ItemDefinition`, равное значению, указанному в `Сообщении, Эскалации, Ошибке` или `Сигнале` соответствующего элемента `EventDefinition`. В случае, если **Событие** оказывает *влияние на ход потока* и для него не указан вход данных, то **События** типа `Сообщение, Эскалация, Ошибка` или `Сигнал` не будут заключать в себе эти данные. В случае, если **Событие** *реагирует на триггер* и для него не указан **Выход Данных**, данные из **Событий** типа `Сообщение, Эскалация, Ошибка` или `Сигнал` в **Процесс** не попадут (т.е. в принципе не покинут **Событие**).

Элемент InputSet

Элемент `Inputset` представляет собой коллекцию элементов `Datainput`, определяющих вместе значимый набор `Входных Данных` для элемента `InputOutputSpecification`. Элемент `InputOutputSpecification` ДОЛЖЕН содержать по-меньшей мере один элемент `InputSet`. Элемент `InputSet` МОЖЕТ ссылаться на любое количество (от нуля и более) элементов `DataInput`. Любой отдельно взятый элемент `DataInput` МОЖЕТ БЫТЬ ассоциирован с множеством элементов `InputSet`, однако, на него всегда ДОЛЖЕН ссылаться по-меньшей мере один элемент `InputSet`.

Если элемент `InputSet` - «пустой» (не ссылающийся ни на один элемент `DataInput`), это означает то, что для начала выполнения **Действия** не требуется наличие каких-либо данных. Использование такого пустого элемента подразумевает отсутствие входов данных или то, что на них ссылается другой набор данных.

Элементы `InputSet` входят в состав элементов `InputOutputSpecification`. То, в какой последовательности они содержатся в элементах `InputOutputSpecification`, определяет порядок, в котором они будут обработаны.



Фигура 10.62 – Диаграмма классов элемента InputSet

Элемент `InputSet` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.61 содержит информацию о дополнительных атрибутах и ассоциациях элемента `InputSet`.

Таблица 10.61 – Атрибуты и ассоциации элемента InputSet

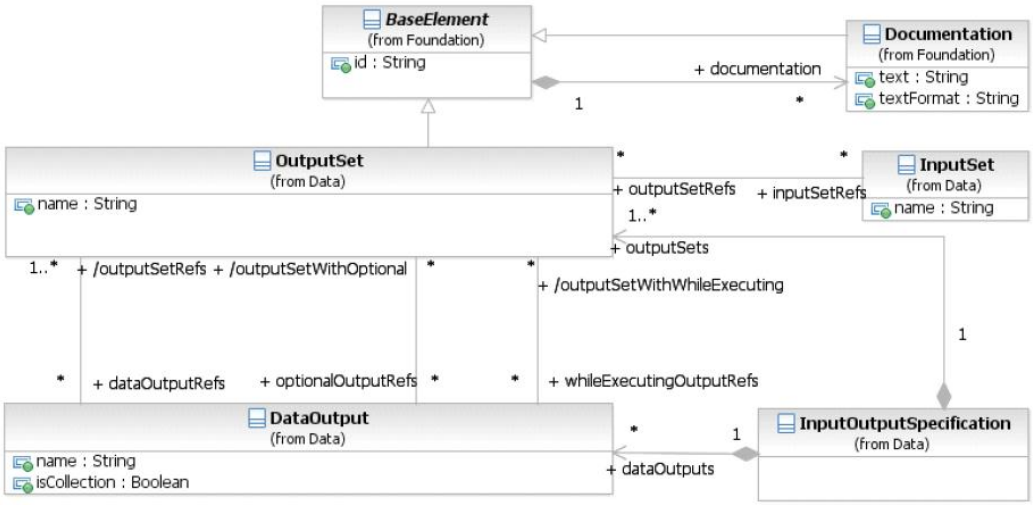
Название атрибута	Описание/использование
name: string [0..1]	Представляет собой описательное имя элемента.
outputSetRefs: OutputSet [1..*]	Данный атрибут определяет элементы Datainput, которые вместе удовлетворяют требованиям к данным.
dataInputRefs: DataInput [0..*]	Данная ассоциация определяет элементы Datainput, являющиеся частью элемента Inputset, которые могут быть недоступны в начале выполнения **Действия**. Такая ассоциация НЕ ДОЛЖНА ссылаться на элементы Datainput, не вошедшие в список значений для datainputrefs.
whileExecutingInputRefs: DataInput [0..*]	Данная ассоциация определяет элементы Datainput, являющиеся частью элемента Inputset, значения которых могут быть обработаны в ходе выполнения **Действия**. Такая ассоциация НЕ ДОЛЖНА ссылаться на элементы Datainput, не вошедшие в список значений для datainputrefs.
outputSetRefs: OutputSet [0..*]	Данный атрибут указывает на правило входа\выхода данных, определяющее, какой элемент OutputSet будет с большой вероятностью создан **Действием**, когда данный элемент InputSet будет иметь верное значение. Данный атрибут используется в паре с атрибутом inputSetRefs элемента OutputSets. Такая комбинация заменяет описанный в **BPMN 1.2** атрибут **Действий** IORules.

Элемент OutputSet

Элемент OutputSet представляет собой коллекцию входных элементов DataOutput, формируемых вместе в качестве выходных данных из **Действия** или События. Элемент InputOutputSpecification ДОЛЖЕН определять по-меньшей мере один элемент OutputSet. Элемент OutputSet МОЖЕТ ссылаться на любое количество (от нуля и более) элементов DataOutput. Любой отдельно взятый элемент DataOutput МОЖЕТ БЫТЬ ассоциирован с множеством элементов OutputSet, однако, на него всегда ДОЛЖЕН ссылаться по-меньшей мере один элемент OutputSet.

Если элемент OutputSet - «пустой» (не ассоциированный ни с одним элементом DataOutput), это означает то, что в ходе выполнения **Действия** не формируется никаких данных.

Объект нотации, для которого указано значение OutputSet, определяет то, какой набор данных будет сформирован на выходе. Таким образом, именно **Действие** или **Событие** определяет какой набор данных будет на выходе.



Фигура 10.63 – Диаграмма классов элемента OutputSet

Элемент `OutputSet` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.62 содержит информацию о дополнительных атрибутах и ассоциациях элемента `OutputSet`.

Таблица 10.62 – Атрибуты и ассоциации элемента OutputSet

Название атрибута	Описание/использование
name: string [0..1]	Представляет собой описательное имя элемента.
dataOutputRefs: DataOutput [0..*]	Данный атрибут определяет элементы DataOutput, которые МОГУТ БЫТЬ произведены на выходе.
optionalOutputRefs: DataOutput [0..*]	Данная ассоциация определяет элементы `DataOutput`, являющиеся частью элемента `OutputSet`, которые не обязательно должны быть сформированы при завершении выполнения **Действия**. Такая ассоциация НЕ ДОЛЖНА ссылаться на элементы `DataOutput`, не вошедшие в список значений для `dataOutputRefs`.
whileExecutingOutputRefs: DataOutput [0..*]	Данная ассоциация определяет элементы `DataOutput`, являющиеся частью элемента `OutputSet`, которые могут быть сформированы в ходе выполнения **Действия**. Такая ассоциация НЕ ДОЛЖНА ссылаться на элементы `DataOutput`, не вошедшие в список значений для `dataOutputRefs`.
inputSetRefs: InputSet [0..*]	Данный атрибут указывает на правило входа\выхода данных, определяющее, какой элемент `InputSet` должен получить значение, верное для возможного создания набора выходных данных. Данный атрибут используется в паре с атрибутом `outputSetRefs` элемента `InputSets`. Такая комбинация заменяет описанный в **BPMN 1.2**

	атрибут **Действий** `IORules`.

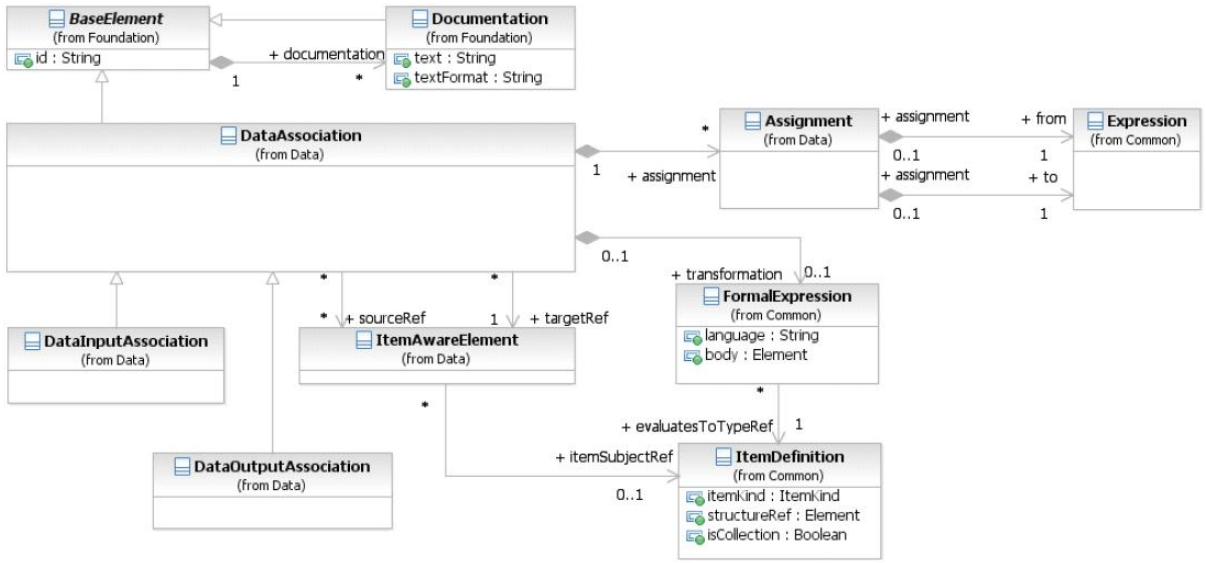
Ассоциация данных

Ассоциации данных используются для перемещения данных между **Объектами данных**, `Свойствами`, а также *входами* и *выходами* **Действий**, **Процессов** и `глобальных Задач`. **Ассоциации данных** не являются объектами, через которые проходят *Токены*, поэтому они не способны оказывать существенное влияние на ход выполнения **Процесса**.
Целью извлечения информации из **Объекта данных** или **Входных Данных Процесса** является помещение этих данных в точку *входа* **Действия** и дальнейшее продвижение *выходных* значений от выполненного **Действия** обратно в **Объект Данных** или **Выходные Данные Процесса**.

Класс DataAssociation

Класс `DataAssociation` представляет собой элемент `BaseElement`, содержащийся в **Действии** или **Событии**. Элементы `DataAssociation` используются для отображения перемещения данных в элементы, содержащие информацию о наборе данных, или из них. Они имеют один или более источников, а также одну или более целей. Цель Ассоциации является копией её источника.

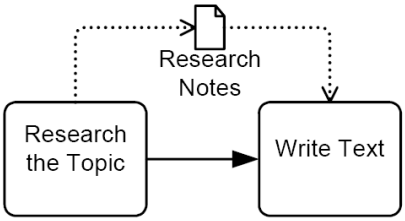
Определенные для `souceRef` и `targetRef` значения атрибута `ItemDefinition` ДОЛЖНЫ иметь одинаковые значения `ItemDefinition`, либо элемент **DataAssociation** ДОЛЖЕН содержать выражение, которое трансформирует (преобразовывает) исходный элемент `ItemDefinition` в целевой элемент `ItemDefinition`.



Фигура 10.64 – Диаграмма класса DataAssociation

По желанию, на диаграмме **Процесса Ассоциация данных** может быть представлена посредством Ассоциации (см. фигуры 10.65 и 10.66).

Фигура 10.65 – Ассоциация данных



Фигура 10.66 – Ассоциация данных, используемая для отображения перемещения входных и выходных данных Действий

Базовая концепция элементов DataAssociation заключается в наличии у них источников, целей, а также возможности трансформации (преобразования).

При «выполнении» Ассоциации данных информация копируется в цель. То, является ли трансформация заданной или нет, определяет содержание копируемых данных.

Если трансформация не была задана или отсутствуют ссылки на нее, то ДОЛЖЕН БЫТЬ указан один единственный источник, а содержимое этого источника будет копироваться в цель.

Если трансформация не была задана или отсутствуют ссылки на нее, то будет вычисляться значение `выражение` трансформации, а результат вычисления будет копироваться в цель. В данном случае может быть указано любое количество источников (от нуля и более), однако, эти источники не обязательно используются в `выражении`.

В любом случае, источники предназначены для определения возможности «выполнения» Ассоциации данных. Ассоциация данных не может быть «выполнена», если эти источники недоступны. В этом случае **Действие** или **Событие**, для которых используется Ассоциация данных, ДОЛЖНЫ ждать выполнения этого условия.

Ассоциации данных всегда располагаются внутри элементов, определяющих то, когда будут «выполнены» эти Ассоциации данных. **Действия** могут определять момент «выполнения» двух Ассоциаций данных, а **События** – лишь одной.

Для **Событий** используется одна Ассоциация данных, но для разных типов **Событий** она используется по-разному. Для *реагирующих на триггер* **Событий** Ассоциация данных используется в целях помещения данных из полученного **Сообщения** в Объект данных и свойства. Для *определяющих ход потока* **Событий** Ассоциация данных используется в целях помещения данных в запущенное **Сообщение**.

Поскольку Ассоциация данных используется на разных участках диаграммы **Процесса** или жизненного цикла **Действия**, ее возможные цели и источники могут меняться в зависимости от того, на каком участке расположен этот графический элемент. Исходя из этого, можно предположить, какие элементы могут являться источниками и целями Ассоциации данных. Например, если запущено выполнение какого-либо **Действия**, то допустимыми целями могут служить входы для данных этого **Действия**. Подобно этому, при завершении выполнения какого-либо **Действия** допустимыми источниками могут служить выходы для данных этого **Действия**.

Элемент `DataAssociation` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.63 содержит информацию о дополнительных ассоциациях элемента `DataAssociation`.

Таблица 10.63 – Ассоциации элемента DataAssociation

Название атрибута	Описание/использование
transformation: Expression [0..1]	Служит для указания опционального `выражения` трансформации. Фактическая область доступных данных для данного `выражения`

	определяется источником и целью типа **Ассоциации данных**.
assignment: Assignment [0..*]	Служит для указания одного или более элементов данных `Assignments`. Посредством элемента `Assignment` в структуру цели могут быть добавлены отдельные элементы структуры данных из структуры источника.
sourceRef: ItemAwareEle-ment [0..*]	Используется для указания источника **Ассоциации данных**. Им ДОЛЖЕН БЫТЬ элемент `ItemAwareElement`.
targetRef: ItemAwareElement	Используется для указания цели **Ассоциации данных**. Ею ДОЛЖЕН БЫТЬ элемент `ItemAwareElement`.

Класс Assignment

Класс `Assignment` используется для установки простого соответствия элементов данных посредством языка `выражений (Expression)`.

Язык `выражений`, применяемый для всех выражений по умолчанию, указывается в атрибуте `expressionLanguage` элемента `Definitions`. Значения атрибута `expressionLanguage` могут отменять друг друга в разных случаях использования `Assignment`.

Элемент `Assignment` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.64 содержит информацию о дополнительных атрибутах элемента `Assignment`.

Таблица 10.64 – Атрибуты элемента Assignment

Название атрибута	Описание/использование
from: Expression	Атрибут, вычисляющий источник элемента `Assignment`.
to: Expression	Атрибут, определяющий активную в текущий момент операцию `Assignment`, а также элемент данных, являющийся целью.

Элемент DataInputAssociation

Элемент `DataInputAssociation` может использоваться для установки ассоциации между элементом `ItemAwareElement` и `Входными Данными`, хранящимися в **Действии**. Источником такой **Ассоциации данных** может стать любой элемент `ItemAwareElement`, доступный в данных условиях, например, **Объект данных**, `Свойство` или `Выражение`.

Элемент `DataInputAssociation` наследует атрибуты и ассоциации элемента `DataAssociation` (см. таблицу 10.64), но не может иметь каких-либо других атрибутов или ассоциаций.

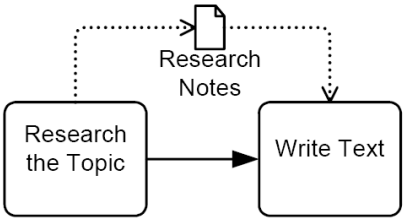
Элемент DataOutputAssociation

Элемент `Dataoutputassociation` может использоваться для установки ассоциации между выходными данными, хранящимися в **Действии,** и элементом `Itemawareelement`, доступным в условиях, при которых ассоциация будет «выполнена». Целью такой **Ассоциации данных** может стать любой элемент `Itemawareelement`, доступный в данных условиях, например, **Объект данных**, `Свойство` или `Выражение`.

Элемент `DataOutputAssociation` наследует атрибуты и ассоциации элемента `DataAssociation` (см. таблицу 10.64), но не может иметь каких-либо других атрибутов или ассоциаций.

Объект данных, ассоциированный с Потоком операций

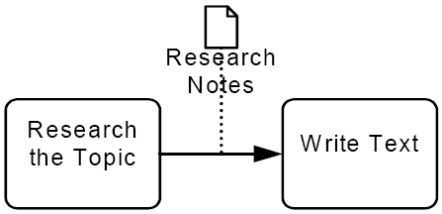
Фигура 10.67 является копией фигуры 10.66 и показывает то, каким образом для отображения входов и выходов для данных в **Действиях** может быть использована **Ассоциация данных**.



Фигура 10.67 – Объект данных как вход и выход

По желанию, **Объект данных** МОЖЕТ БЫТЬ ассоциирован с **Потоком операций** напрямую, т.е. с соединителем (см. фигуру 10.68). Такой тип отображения передает ту же самую взаимосвязь между входами и выходами. Фигура 10.67 представляет собой пример упорядочивания двух **Ассоциаций данных**:

- Одна **Ассоциация данных** направлена от элемента, содержащего набор данных (например, **Действия**) и входящего в состав источника **Потока операций**, и соединена с **Объектом данных**.
- Вторая **Ассоциация данных** направлена от **Объекта данных** и соединена с содержащим набор данных элементом, входящим в состав цели **Потока операций**.



Фигура 10.68 – Объект данных, ассоциированный с Потоком операций

10.3.2. Семантика исполнения для данных

Когда элемент, определяющий значение `InputOutputSpecification`, готов к запуску выполнения посредством элементов **Потока операций** или **События**, на которое должна последовать реакция, входы заполняются данными, поступающими от соответствующих элементов, например, **Объектов данных** или свойств. Для отображения таких назначений служит **Ассоциация данных**.

Значение каждого их указанных элементов `InputSet` вычисляется согласно порядку, в котором эти элементы расположены в `InputOutputSpecification`.

Входные данные, на которые ссылается элемент `InputSet`, будут вычислены в случае, если они верны.

Все ассоциации данных будут вычислены в том случае, если они указывают вход данных в качестве цели. Если один из источников какой-либо ассоциации данных является **недоступным**, то значение `InputSet` также становится **недоступным**; при этом вычисляется значение следующего элемента `InputSet`.

Первый элемент `InputSet`, в котором все входные данные доступны для ассоциаций данных, используется для запуска выполнения Действия. Если же значения всех этих элементов недоступны, выполнение **Действия** не начнется до тех пор, пока не будет выполнено данное условие.

Данная спецификация не содержит информацию о том, когда и как часто должно проверяться это условие. Выполнение не будет осуществляться до тех пор, пока источники ассоциаций данных не станут доступны, после чего произойдет повторное вычисление значений элементов `InputSets`.

Если в состав Потока операций включено **Событие**, *реагирующее на триггер* или *определяющее результат*, семантика исполнения для данных будет отличаться.

Если запущено определяющее результат **Событие**, то будут задействованы все элементы `DataInputAssociation` данного **События**; при этом произойдет наполнение входов данного **События** данными. Затем элементы `DataInput` копируются в объекты, запущенные вышеуказанным **Событием** (**Сообщением, Сигналом** и т.д.). Поскольку для **События** не определены элементы `InputSets`, то и выполнение никогда не будет ждать удовлетворения каких-либо условий.

Если запущено *реагирующее на триггер* **Событие,** то выходы данного **События** заполняются объектом, запустившим данное **Событие**; при этом задействованы все элементы `DataOutputAssociations` **События**. Элементы `OutputSets` для этого **События** не определяются.

Для того, чтобы сделать возможным запуск **Процесса** посредством **Действия Вызов** и **Потока Сообщений**, **Стартовое** и **Конечное События** используются одинаково.

При использовании **Стартового События** входные данные содержащего его **Процесса** доступны элементам `DataOutputAssociations` **События** как цели. Таким образом, входы **Процесса** могут быть заполнены элементами, запустившими **Стартовое Событие**.

При использовании **Конечного События** выходные данные содержащего его **Процесса** доступны элементам `DataInputAssociations` **События** как источники. Таким образом, результирующие элементы **Конечного События** в качестве источников могут использовать **Входные Данные Процесс**а.

Если хотя бы один раз элемент `InputSet` стал доступным, то происходит выполнение всех **Ассоциаций данных**, целью которых является любые входные данные элемента `InputSet`. Благодаря этому, заполняются входы для данных **Действия**, при этом само **Действие** начинает выполняться. Когда выполнение **Действия** завершается, то происходит выполнение всех **Ассоциаций данных**, источниками которых являются любые **Выходные Данные** элемента `OutputSet`. Благодаря этому, значения **Выходных Данных** копируются обратно в элемент, содержащий данные (**Объект данных**, свойства и т.д.)

Семантика исполнения для DataAssociation

Выполнение любого объекта **Ассоциации данных** ДОЛЖНО БЫТЬ произведено в соответствии со следующей семантикой исполнения:

- Если **Ассоциация данных** определяет выражение трансформации, то данное выражение вычисляется, а результат копируется в элемент targetRef. Благодаря этим действия, предыдущее значение этого элемента полностью заменяется.
- Для каждого элемента назначения необходимо выполнять следующие действия:
    - Вычислить выражение «from» и получать *source value*.
    - Вычислить выражение «to» и получать * target element *. В качестве элемента * target element * может использоваться любой соответствующий элемент или подчиненный ему элемент (например, Объект данных или подчиненный ему элемент).
    - Скопировать *source value* в *target element*.
- Если в **Ассоциации данных** отсутствуют выражение трансформации либо элементы назначения, то:
    - Скопировать значение элемента «sourceRef» **Ассоциации данных** в элемент «targetRef». В данном случае доступен лишь элемент sourceRef.

10.3.3. Использование данных в выражениях XPath

Механизм расширения **BPMN** позволяет использовать различные языки `выражений` и запросов. Данный раздел посвящен описанию использования XPath в **BPMN**: в нем представлен механизм доступа к **Объектам данных**, *свойствам* и другим атрибутам *экземпляров* через использование `выражений` XPath.

В языке `выражений` доступность подразумевает основанную на информации возможность доступа **Действий**, содержащих `выражение`. Все элементы, доступные из содержащего их элемента выражения XPath, ДОЛЖНЫ БЫТЬ доступны для XPath-процессора.

В **BPMN Объекты данных** и *Свойства* указываются посредством элемента `ItemDefinition`. Связь с XPath предполагает, что элемент `ItemDefinition` является либо сложным XSD типом, либо XSD элементом. Если используется XSD элемент, то он ДОЛЖЕН БЫТЬ заявлен как одноблочная (with a single member node) узловая переменная (node-set XPath variable). Если используется сложный XSD тип, то он ДОЛЖЕН БЫТЬ заявлен как одноблочная переменная node-set, содержащая безымянный элемент документа, в котором хранится текущее значение **Объекта данных** или *Свойства* **BPMN**.

Доступ к Объектам данных BPMN

Таблица 10.65 содержит информацию о функциях XPath, используемых для получения доступа к **Объектам данных BPMN**. `Параметр processName` указывает Процесс и имеет тип `string` (ОБЯЗАТЕЛЬНО literal string).

Таблица 10.65 – Функция расширения XPath для Объектов данных

Функция расширения XPath	Описание/использование
Element getDataObject ('processName', 'dataObject- Name')	Данная функция возвращает значение представленного **Объекта данных**. Параметр `processName` является опциональным. В случае, если он опущен, то **Процесс**, содержащий **Действие** с `выражением`, допускается. Для получения доступа к **Объектам данных**, указанным в родительском процессе, ДОЛЖЕН БЫТЬ использован параметр `processName`. В других ситуациях он ДОЛЖЕН БЫТЬ опущен.

Поскольку возврат отказа функциями XPath 1.0 не поддерживается, в случае ошибки возвращается пустой набор узлов.

Доступ к входам и выходам данных BPMN

Таблица 10.66 содержит информацию о функциях XPath, используемых для получения доступа к **Входам** и **Выходам Данных BPMN**. Параметр `dataInputName` указывает вход данных и имеет тип `string`.

Таблица 10.66 – Функция расширения XPath для входа и выхода данных

Функция расширения XPath	Описание/использование
Element getDataInput ('dataInputName')	Данная функция возвращает значение представленного входа для данных.
Element getDataOutput ('dataOutput- Name')	Данная функция возвращает значение выхода для данных.

Доступ к Свойствам BPMN

Таблица 10.67 содержит информацию о функциях XPath, используемых для получения доступа к *свойствам* **BPMN**.

Параметр `processName` указывает **Процесс** и имеет тип string. Параметр `propertyName` указывает свойство и имеет тип string. Параметр `activityName` указывает **Действие** и имеет тип string. Параметр `eventName` указывает **Событие** и имеет тип string (ОБЯЗАТЕЛЬНО literal string). Функция расширения XPath возвращает значение представленного свойства.

Поскольку возврат отказа функциями XPath 1.0 не поддерживается, в случае ошибки возвращается пустой набор узлов.

Таблица 10.67 – Функция расширения XPath для входа и выхода данных

Функция расширения XPath	Описание/использование
Element getProcessProperty (‘processName’, ‘propertyName’)	Данная функция возвращает значение представленного свойства **Процесса**. Параметр `processName` является опциональным. В случае, если он опущен, то **Процесс**, содержащий **Действие** с выражением, допускается. Для получения доступа к свойствам, указанным в родительском процессе, ДОЛЖЕН БЫТЬ использован параметр `processName`. В других ситуациях он ДОЛЖЕН БЫТЬ опущен.
Element getActivityProperty (‘activityName’, ‘propertyName’)	Данная функция возвращает значение представленное свойство **Действия**.
Element getEventProperty ‘eventName’, ‘propertyName’)	Данная функция возвращает значение представленное свойство **События**.

Доступ к атрибутам экземпляров BPMN

Таблица 10.68 содержит информацию о функциях XPath, используемых для получения доступа к *атрибутам экземпляров* **BPMN.**

Параметр `processName` указывает **Процесс** и имеет тип `string`. Параметр `attributeName` указывает атрибут экземпляра и имеет тип `string`. Параметр `activityName` указывает **Действие** и имеет тип `string` (ОБЯЗАТЕЛЬНО literal string).

Эти функции возвращают значение представленного экземпляра **Действия**. Поскольку возврат отказа функциями XPath 1.0 не поддерживается, в случае ошибки возвращается пустой набор узлов.

Таблица 10.68 – Функции расширения XPath для атрибутов экземпляров

Функция расширения XPath	Описание/использование
Element getProcessInstanceAttribute (‘processName’,‘attributeName’)	Данная функция возвращает значение представленного атрибута *экземпляра* **Процесса**. Параметр `processName` является опциональным. В случае, если он опущен, то **Процесс**, содержащий **Действие** с `выражением`, допускается. Для получения доступа к атрибутам родительского процесса, указанным

	в родительском процессе, ДОЛЖЕН БЫТЬ использован параметр `processName`. В других ситуациях он ДОЛЖЕН БЫТЬ опущен.
Element getChoreographyInstance- Attribute (‘attributeName’)	Данная функция возвращает значение представленного атрибута *экземпляра* **Хореографии**.
Element getActivityInstanceAttribute (‘activityName’, ‘attributeName’)	Данная функция возвращает значение атрибута *экземпляра* **Действия**. Примерами такого **Действия** могут служить **Пользовательская задача** и цикл.

10.3.4. Представление XML-схемы для Данных

Таблица 10.69 – XML–схема для элемента Assignment

```
<xsd:element name="assignment" type="tAssignment" />
<xsd:complexType name="tAssignment">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="from" type="tExpression" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="to" type="tExpression" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

Таблица 10.70 – XML–схема для элемента DataAssociation

```
<xsd:element name="dataAssociation" type="tDataAssociation" />
<xsd:complexType name="tDataAssociation" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="sourceRef" type="xsd:IDREF" minOccurs="0" maxOccurs="unbounded"/
                    >
                <xsd:element name="targetRef" type="xsd:IDREF" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="transformation" type="tFormalExpression" minOccurs="0" maxOc-
                        curs="1"/>
                <xsd:element ref="assignment" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.71 – XML–схема для элемента DataInput

```
<xsd:element name="dataInput" type="tDataInput" />
<xsd:complexType name="tDataInput">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="name" type="xsd:string" use="optional" />
            <xsd:attribute name="itemSubjectRef" type="xsd:QName" />
            <xsd:attribute name="isCollection" type="xsd:boolean" default="false"/>
            <xsd:attribute name="dataState" type="xsd:IDREF"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.72 – XML–схема для элемента DataInputAssociation

```
<xsd:element name="dataInputAssociation" type="tDataInputAssociation" />
<xsd:complexType name="tDataInputAssociation">
    <xsd:complexContent>
        <xsd:extension base="tDataAssociation"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.73 – XML–схема для элемента DataObject

```
<xsd:element name="dataObject" type="tDataObject" />
<xsd:complexType name="tDataObject">
    <xsd:complexContent>
        <xsd:extension base="tFlowElement">
            <xsd:sequence>
                <xsd:element ref="dataState" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="itemSubjectRef" type="xsd:QName"/>
            <xsd:attribute name="isCollection" type="xsd:boolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.74 – XML–схема для элемента DataState

```
<xsd:element name="dataState" type="tDataState" />
<xsd:complexType name="tDataState">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="name" type="xsd:string"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.75 – XML–схема для элемента DataOutput

```
<xsd:element name="dataOutput" type="tDataOutput" />
<xsd:complexType name="tDataOutput">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="name" type="xsd:string" use="optional"/>
            <xsd:attribute name="itemSubjectRef" type="xsd:QName"/>
            <xsd:attribute name="isCollection" type="xsd:boolean" default="false"/>
            <xsd:attribute name="dataState" type="xsd:IDREF"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.76 – XML–схема для элемента DataOutputAssociation

```
<xsd:element name="dataOutputAssociation" type="tDataOutputAssociation" />
<xsd:complexType name="tDataOutputAssociation">
    <xsd:complexContent>
        <xsd:extension base="tDataAssociation"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.77 – XML–схема для элемента InputOutputSpecification

```
<xsd:element name="ioSpecification" type="tInputOutputSpecification" />
<xsd:complexType name="tInputOutputSpecification">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element ref="dataInput" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="dataOutput" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="inputSet" minOccurs="1" maxOccurs="unbounded"/>
                <xsd:element ref="outputSet" minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
```

```
</xsd:complexType>
```

Таблица 10.78 – XML–схема для элемента InputSet

```
<xsd:element name="inputSet" type="tInputSet" />
<xsd:complexType name="tInputSet">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="dataInputRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
                <xsd:element name="optionalInputRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
                <xsd:element name="whileExecutingInputRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
                <xsd:element name="outputSetRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" />
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.79 – XML–схема для элемента OutputSet

```
<xsd:element name="outputSet" type="tOutputSet" />
<xsd:complexType name="tOutputSet">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="dataOutputRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
                <xsd:element name="optionalOutputRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
                <xsd:element name="whileExecutingOutputRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
                <xsd:element name="inputSetRefs" type="xsd:IDREF" minOccurs="0" maxOc-
                        curs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.80 – XML–схема для элемента Property

```
<xsd:element name="property" type="tProperty" />
<xsd:complexType name="tProperty">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:attribute name="name" type="xsd:string"/>
            <xsd:attribute name="itemSubjectRef" type="xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

10.4. Событие

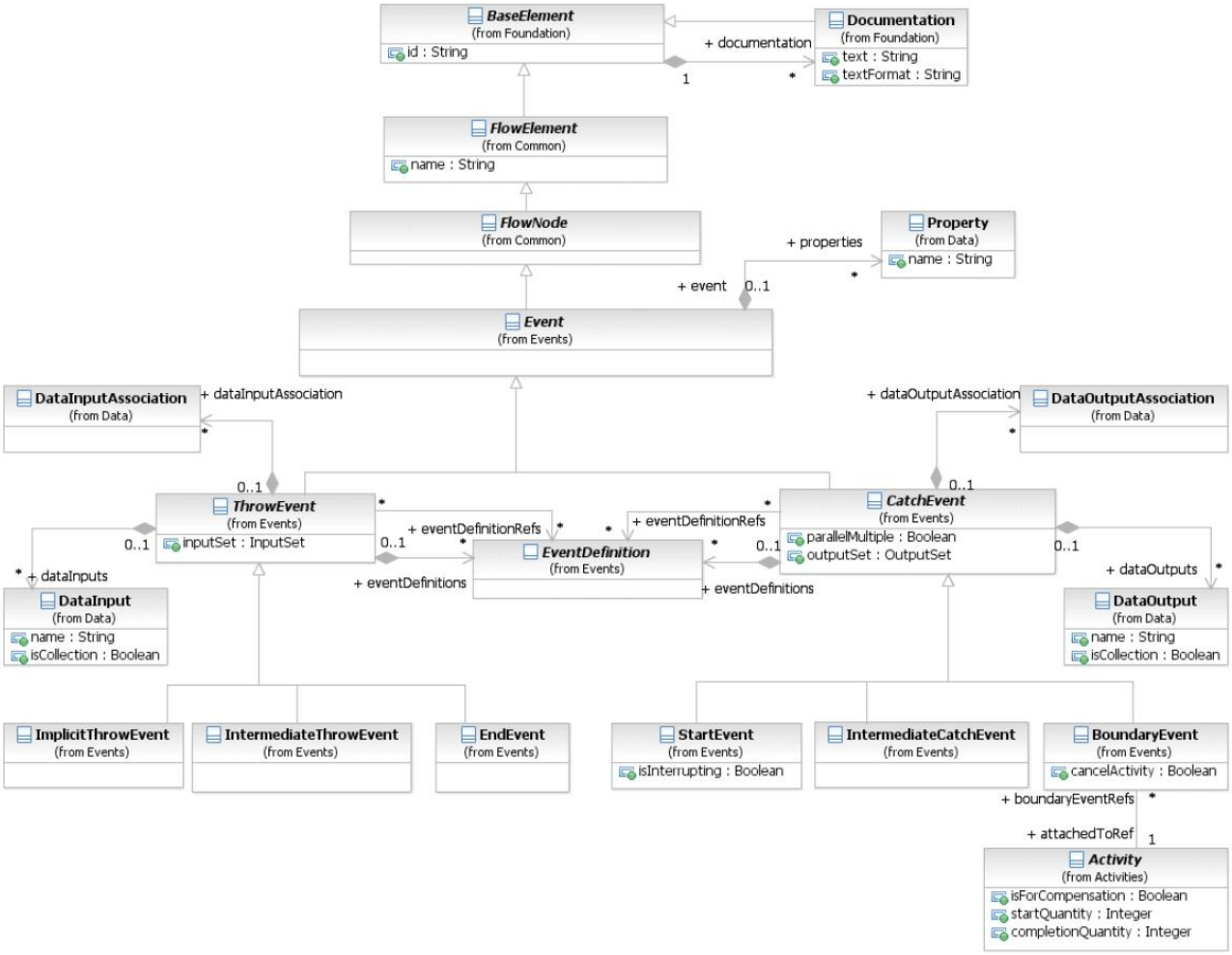
Событие – это то, что происходит в течение **Бизнес-процесса** или его **Хореографии**. **Событие** оказывает влияние на ход **Бизнес-процесса** и чаще всего имеет причину (триггер) или воздействие (результат). Изображается в виде круга со свободным центром, предназначенным для дифференцировки внутренними маркерами различных триггеров или их результатов. Объемное понятие «**Событие**» включает в себя такие попадающие под эту категорию явления **Процесса**, как: старт **Процесса**, его завершение, смена статуса задействуемого документа, получение **Сообщения** и мн.др.

Согласно влиянию **Событий** на ход **Бизнес-процесса**, выделяют три типа:

1. **Стартовое событие** (указывающее на точку запуска **Процесса**)
2. **Конечное событие** (указывающее на точку завершения **Процесса**)
3. **Промежуточное событие** (указывающее на что-то, происходящее на ограниченном **Стартовым** и **Конечным Событиями** отрезке **Процесса**).

Вышеперечисленные типы **Событий**, в свою очередь, могут:

- *Обрабатывать триггер*. Такими являются все **Стартовые** и некоторые **Промежуточные События**.
- *Определять результат (возбуждать триггер)*. Такими являются все **Конечные** и некоторые **Промежуточные События**. Они МОГУТ *влиять* на другие **События**. Как правило, *триггеры* передают информацию из места, где произошло такое **Событие**, туда, где находится *реагирующее на триггер* **Событие**. Согласно стандартам **BPMN**, *возбуждение триггера* МОЖЕТ БЫТЬ скрытым (или каким-либо другим, являющимся разновидностью скрытого) или явным (при содействии *определяющего результат* **События**).



Фигура 10.69 – Диаграмма классов элемента Event

10.4.1. Общее представление о Событии

В зависимости от типа **События**, применяются различные способы запуска триггера в направлении реагирующего на него **События**: публикация, вынесение непосредственного решения, распространение, отмена, а также компенсация.

В рамках публикации *триггер* МОЖЕТ быть получен **Событием**, находящимся в любой части системы, где производится публикация данного *триггера*. **События**, для которых используется данный способ *возбуждения триггера*, образуют группу **Обмена сообщениями**. В качестве *триггеров* здесь выступают **События**, сформированные за рамками **Пула**, в котором производится их публикация. Обычно они используются для описания B2B модели сообщения между разными **Процессами** разных **Пулов**. Для указания определенного *экземпляра* **Процесса** в момент, когда такое **Сообщение** должно достичь данного *экземпляра* **Процесса**, используется *корреляция*. *Триггеры*, называемые **Сигналами**, формируются в **Пуле**, где производится их публикация. Как правило, они используются для распространения **Сообщения** в **Процессах** и между ними, а также между **Пулами** и диаграммами **Процессов**.

Триггеры, называемые **Таймер** и **Условие**, запускаются скрыто. После запуска они ожидают условия, при котором могут повлиять на **Событие**, *обрабатывающее триггер*. Эти условия могут быть основаны на временном или статусном критериях.

Триггер, используемый в качестве распространения, передается от места, где было запущено **Событие**, к наиболее удаленному месту *экземпляра* (например, на другой уровень **Процесса**), к которому привязано **Событие**, *реагирующее на триггер*. *Триггер*, используемый в качестве **Ошибки**, является критичным и прекращает выполнение в месте его запуска. **Эскалация**, в свою очередь, критичной не является, а выполнение продолжается в месте запуска этого *триггера*. В случае, если для двух последних типов *триггеров* не определено ни одного реагирующего на них **События**, они считаются невыполненными (неразрешенными). В рамках **Процесса** или определенного *экземпляра* **Действия** рекомендуется использовать завершение, *компенсацию*, а также *отмену*. Завершение используется для указания того, что выполнение всех **Действий**, содержащихся в **Процессе** или другом **Действии**, должно быть немедленно завершено (без *компенсации* или использования **События**). Это касается все *экземпляров многоэкземплярного* **Действия** или **Процесса**.

Компенсация удачно выполненного **Действия** запускает *обработчика* этой *компенсации*. *Обработчик компенсации* может быть двух типов: определяемый пользователем и скрытый. Скрытый *обработчик компенсации*, используемый для **Подпроцесса**, вызывает *обработчики компенсаций* всех содержащихся в нем **Действий** в последовательности, обратной **Потоку операций**. В случае, если вызываемая *компенсация* предназначается для *ещё не выполненного* или *неудачно выполненного* **Действия**, она никак себя не проявляет (в частности, не возникает никаких ошибок).

Отмена прерывает выполнение всех активных **Действий** и *компенсирует* все успешно выполненные **Действия** **Подпроцесса**, для которого она используется. Если этот **Подпроцесс** является *Транзакцией*, то он возвращается в начало.

Моделирование данных в Событиях

Некоторые **События** (например, **Сообщение, Эскалация, Ошибка, Сигнал, Множественное**) могут содержать в себе данные. Для продвижения данных от *реагирующего на триггер* **События** к элементу данных используется **Ассоциация данных**. Однако при использовании **Событий**, содержащих данные, должны быть приняты во внимание следующие ограничения:

- Если **Событие** связанно с множеством элементов `EventDefinitions`, то каждому из этих элементов ДОЛЖЕН соответствовать один **Вход для данных** (если **Событие** *определяет результат*) или один **Выход для данных** (если **Событие** *реагирует на триггер*) **События**. Порядок расположения элементов `EventDefinition` и порядок размещения **Входов/Выходов для данных** определяют, какой **Вход/Выход** какому элементу соответствует.
- Если **Событие** имеет **вход/выход для данных**, то для каждой пары «элемент `EventDefinition` - **вход/выход для данных**» ДОЛЖЕН БЫТЬ определен элемент `ItemDefinition`, значение которого равно значению, установленному для элементов `ItemDefinition` **Сообщения**, **Эскалации**, **Ошибки** или **Сигнала**. В случае, если **Событие** *определяет результат* и не имеет **входа для данных**, то данные в **Сообщение, Эскалацию**, **Ошибку** или **Сигнал** переданы не будут. Если

же **Событие** реагирует на триггер и не имеет **выхода для данных**, то данные из **Сообщения, Эскалации, Ошибки** или **Сигнала** не покинул пределы этого **События** и не попадут в **Процесс**.

Во время выполнения **События** ведут себя следующим образом:

- *Определяющие результат* **События**. После запуска **События**, соответствующий элемент `EventDefinition` ссылается на **входные данные**, автоматически переданные в **Сообщение**, **Эскалацию, Ошибку** и **Сигнал**.
- Имеющие *триггер* **События**. Когда появляется предназначенный для данного **События** *триггер* (например, при получении **Сообщения**), данные автоматически передаются на **выход для данных**, соответствующий элементу `EventDefinition` с описанием *триггера*.

Общие атрибуты События

Элемент **Event** наследует атрибуты и ассоциации элемента `FlowElement` (см. таблицу 8.44). Таблица 10.81 содержит информацию о дополнительных ассоциациях элемента **Event**.

Таблица 10.81 – Ассоциации элемента Event

Название атрибута	Описание/использование
properties: Property [0..*]	Разработчик модели МОЖЕТ добавить в **Событие** какие-либо `свойства`. Эти `свойства` хранятся в **Событии**.

Общие атрибуты обрабатывающего триггер События

Элемент `CatchEvent` наследует атрибуты и ассоциации элемента **Event** (см. таблицу 10.81). Таблица 10.82 содержит информацию о дополнительных атрибутах и ассоциациях элемента `CatchEvent`.

Таблица 10.82 – Атрибуты и ассоциации элемента CatchEvent

Название атрибута	Описание/использование
eventDefinitionRefs: EventDefinition [0..*]	Ссылается на повторно используемые элементы `EventDefinition`, являющиеся ожидаемыми для данного вида **Событий** триггерами. Повторно используемые элементы `EventDefinition` определены как высокоуровневые элементы. Эти элементы могут быть использованы как для **Событий**, имеющих триггер, так и для **Событий**, *определяющих результат*. • В случае, если не указано значение ни одного элемента `EventDefinition`, это означает, что тип имеющего *триггер* **События** не определен, а графический элемент **Событие** не содержит маркера для дифференцировки (см. фигуру 10.91). • В случае, если для элемента `EventDefinition` указано более одного значения, это означает, что данное имеющее *триггер* **События** относится к **Множественному типу**, а графический элемент **Событие** содержит соответствующий маркер (пятиугольник)

	для дифференцировки (см. фигуру 10.90). Является упорядоченным множеством.
eventDefinitions: EventDefinition [0..*]	Определяет элементы `EventDefinition` **События**, являющиеся ожидаемыми для данного вида **Событий** *триггерами*. Значения этих элементов верны лишь в пределах текущего **События**. • В случае, если не указано значение ни одного элемента `EventDefinition`, это означает, что тип имеющего *триггер* **События** не определен, а графический элемент **Событие** не содержит маркера для дифференцировки (см. фигуру 10.91). • В случае, если для элемента `EventDefinition` указано более одного значения, это означает, что данное имеющее *триггер* **События** относится к **Множественному типу**, а графический элемент **Событие** содержит соответствующий маркер (пятиугольник) для дифференцировки (см. фигуру 10.90). Является упорядоченным множеством.
dataOutputAssociations: Data OutputAssociation [0..*]	Указывает **Ассоциацию данных** имеющего *триггер* **События.** Элемент `dataOutputAssociation`, относящийся к данному **Событию**, используется для передачи данных из **События** в элемент данных, находящийся в рамках этого **События**. В зависимости от того, какие индивидуальные *триггеры* имеет **Многоэкземплярное событие** этого вида, могут быть НЕОБХОДИМЫ множественные **Ассоциации данных**.
dataOutputs: DataOutput [0..*]	Определяет **выходы для данных** имеющего *триггер* **События**. Является упорядоченным множеством.
outputSet: OutputSet [0..1]	Определяет набор **выходных данных** для имеющего триггер **События**.
parallelMultiple: boolean = false	Данный атрибут является важным только тогда, когда указаны значения нескольких элементов `EventDefinition` имеющего *триггер* **События** (**Множественного события**). Если установленное значение равно «*true*», то все типы *триггеров*, используемых для данного **События**, ДОЛЖНЫ запуститься до начала **Процесса**.

Общие атрибуты возбуждающего триггер События

Элемент `ThrowEvent` наследует атрибуты и ассоциации элемента **Event** (см. таблицу 10.81). Таблица 10.83 содержит информацию о дополнительных атрибутах и ассоциациях элемента `ThrowEvent`.

Таблица 10.83 – Атрибуты и ассоциации элемента ThrowEvent

Название атрибута	Описание/использование
eventDefinitionRefs: EventDefinition [0..*]	Ссылается на повторно используемые элементы `EventDefinition`, являющиеся ожидаемым для данного вида **Событий** *результатом*. Повторно используемые элементы `EventDefinition` определены как высокоуровневые элементы. Эти элементы могут быть использованы как для **Событий**, *имеющих триггер*, так и для **Событий**, *определяющих результат*. • В случае, если не указано значение ни одного элемента `EventDefinition`, это означает, что тип *определяющего результат* **События** не определен, а графический элемент **Событие** не содержит маркера для дифференцировки (см. фигуру 10.91). • В случае, если для элемента `EventDefinition` указано более одного значения, это означает, что данное *определяющее результат* **Событие** относится к **Множественному типу**, а графический элемент **Событие** содержит соответствующий маркер (пятиугольник) для дифференцировки (см. фигуру 10.90). Является упорядоченным множеством.
eventDefinitions: EventDefinition [0..*]	Определяет элементы `EventDefinition` **События**, являющиеся ожидаемым для данного вида **Событий** *результатом*. Значения этих элементов верны лишь в пределах текущего **События**. • В случае, если не указано значение ни одного элемента `EventDefinition`, это означает, что тип *определяющего результат* **События** не определен, а графический элемент **Событие** не содержит маркера для дифференцировки (см. фигуру 10.91). • В случае, если для элемента `EventDefinition` указано более одного значения, это означает, что данное *определяющее результат* **Событие** относится к **Множественному типу**, а графический элемент **Событие** содержит соответствующий маркер (пятиугольник) для дифференцировки (см. фигуру 10.90). Является упорядоченным множеством.
dataInputAssociations: DataInput Association [0..*]	Указывает **Ассоциацию данных** *определяющего*

	результат **События**. Элемент `dataInputAssociation` используется для передачи элемента данных, находящегося в пределах **События**, в данные этого **События**. В зависимости от того, к какому *результату* приводит **Многоэкземплярное событие** этого вида, могут быть НЕОБХОДИМЫ множественные **Ассоциации данных**.
dataInputs: DataInput [0..*]	Определяет **Входы Для Данных** *определяющего результат* **События**. Является упорядоченным множеством.
inputSet: InputSet [0..1]	Определяет набор входных данных для *определяющего результат* **События**.

Скрытое возбуждающее триггер Событие

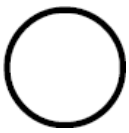
Подтипом *определяющего результат* **События** является элемент `ImplicitThrowEvent` (скрытое *возбуждающие триггер* **Событие**). На диаграмме **Процесса** это **Событие** не отображается и используется для **Многоэкземплярных Действий**. Элемент `ImplicitThrowEvent` атрибуты и ассоциации элемента `ThrowEvent` (см. таблицу 10.84), однако, не может иметь каких-либо других дополнительных атрибутов или ассоциаций.

10.4.2. Стартовое событие

Как видно из названия, **Стартовое Событие** указывает на то, в какой точке берет начало тот или иной **Процесс**. Говоря языком **BPMN**, от **Стартового События** берет начало **Поток Операций Процесса**, а значит, оно не может иметь *Входящих* **Потоков Операций**.

Изображается в виде круга со свободным центром (установленное в **BPMN** отображение Графического Элемента **Событие**), предназначенным для дифференцировки внутренними маркерами различных триггеров или их результатов.

- **Событие** представляет собой круг со свободным центром, который ДОЛЖЕН БЫТЬ выполнен одинарной тонкой линией (см. фигуру 10.70).
    - Текст, цвет, размер, а также линии, используемые для изображения **События**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».
        - Толщина линии ДОЛЖНА БЫТЬ тонкой настолько, чтобы без труда можно было бы отличить **Стартовое событие** от **Промежуточного** или **Конечного**.



Фигура 10.70 – Графический элемент Стартовое событие

На протяжении изложения материала данного документа авторами будет обсуждаться то, в каком направлении движется **Поток операций** внутри **Процесса**. Для облегчения усвоения материала обратимся к понятию «*токен*». *Токен* пересекает **Поток операций** и проходит через **Элементы потока**, содержащиеся в **Процессе**. *Токен* – это теоретическое понятие, используемое для определения характера поведения выполняемого **Процесса**. Поведение элементов **Процесса** определяется посредством описания их взаимоотношений с *токеном*, когда он пересекает **Процесс**.

Примечание: *Токен* не пересекает **Поток сообщений**, поскольку именно **Сообщение** передается посредством **Потока сообщений**.

Стартовое событие имеет следующие особенности:

- **Стартовое событие** является НЕОБЯЗАТЕЛЬНЫМ. Высокоуровневый **Процесс**, Встроенный **Подпроцесс** (уровни **Процесса**), а также Глобальный **Процесс** (**Процесс**) МОГУТ содержать **Стартовое событие**, однако, это НЕ является ОБЯЗАТЕЛЬНЫМ.

Примечание: Диаграмма **Бизнес-Процесса** может включать **Процессы** разных уровней (например, может содержать Развернутый **Подпроцесс** или **Действие Вызов** для вызова других **Процессов**). **Стартовое** и **Конечное События** используются индивидуально для каждого уровня диаграммы **Процесса**.

- Если **Процесс** является сложным и/или условия его запуска не очевидны, то РЕКОМЕНДУЕТСЯ использовать **Стартовое Событие**.
- Если **Стартовое Событие** не включено в Состав **Процесса**, то скрытое **Стартовое Событие** НЕ ДОЛЖНО реагировать на т*риггер*.
- Если в состав **Процесса** включено **Конечное Событие**, то в его состав должно также входить по-меньшей мере одно **Стартовое Событие**.
- Все Элементы потока, не имеющие *Входящих* **Потоков Операций** (например, те, что не являются целью **Потока операций**), ДОЛЖНЫ запускаться при старте экземпляра **Процесса**.
    - Исключения составляют **Действия**, используемые в качестве **Компенсаций** (имеют соответствующий маркер). **Действие Компенсация** не является частью **Стандартного потока** операций и НЕ ДОЛЖНО запускаться одновременно с **Процессом**.
    - Исключением является и реагирующее на *триггер* **Промежуточное Событие Связь**, которое не может иметь *Входящих* **Потоков Операций**.
    - Исключением также является **Событийный Подпроцесс**, который не может иметь *Входящих* **Потоков Операций** и запускается только тогда, когда запускается **Стартовое Событие**, включенное в его состав.
- Процесс МОЖЕТ содержать множество **Стартовых событий** на любом уровне.
    - Каждое **Стартовое событие** является независимым, т.е. новый *экземпляр* **Процесса** ДОЛЖЕН БЫТЬ сформирован при запуске **Стартового события**.

Если **Процесс** является глобальным (**Процесс**, вызываемый **Действиями Вызов** других **Процессов**) и содержит множество **Стартовых событий** неопределенного типа, то при перемещении **Потока операций** из родительского **Процесса** в глобальный будет запущено лишь одно из **Стартовых событий** глобального **Процесса**. Атрибут `targetRef` **Потока операций**, направленного к **Действию Вызов**, может быть расширен с целью указать соответствующее **Стартовое событие**.

Примечание: Если **Процесс** содержит несколько **Стартовых событий**, его поведение может стать более сложным для понимания. РЕКОМЕНДУЕТСЯ как можно более редкое использование такой схемы во избежание неверного понимания пользователями диаграммы **Процесса** цели, для которой она создавалась.

При появлении *триггера*, предназначенного для **Стартового События**, запускается новый экземпляр **Процесса**, и для каждого *Исходящего* **Потока Операций**, направленного от этого **События**, создается *токен*.

Триггеры Стартового события

Стартовые события могут быть включены в состав следующих типов **Процессов**:

- *Высокоуровневые* **Процессы**
- **Подпроцессы** (*встроенные*)
- Глобальные **Процессы** (*вызываемые*)

- **Событийные Подпроцессы**

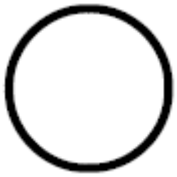
Следующие три подраздела содержат описание типов **Стартовых событий**, используемых для разных типов **Процессов**.

Стартовые события в Высокоуровневых Процессах

Существует множество способов запуска *высокоуровневых* **Процессов** и их экземпляров. Графическое изображение т*риггеров* **Стартовых событий** помогает отобразить общий механизм запуска экземпляров отдельно взятого **Процесса**. В состав *высокоуровневого* **Процесса** могут входить как **Стартовые события** с неопределенным типом *триггера* (тип *Неопределенный*), так и **События**, имеющие *триггеры*: «**Сообщение**», «**Таймер**», «**Условие**», «**Сигнал**», «**Множественный**», «**Параллельный**».

Высокоуровневый **Процесс,** содержащий хотя бы одно **Стартовое событие** с неопределенным типом *триггера*, МОЖЕТ вызываться посредством **Действия Вызов** другого **Процесса**. Такое **Стартовое событие** используется для запуска **Процесса Действием Вызов**. Другие типы **Стартового события** могут использоваться только в том случае, если текущий **Процесс** – *высокоуровневый*.

Таблица 10.84 – Типы Стартового события высокоуровневого Процесса

Тип триггера	Описание	Маркер
Неопределенный	Отсутствие маркера предполагает отсутствие у **Стартового события** определенного *триггера*. Для **Стартовых событий** без *триггера* не существует конкретного подкласса `EventDefinition`. В случае, если **Стартовое событие** не связано ни с каким элементом `EventDefiniton`, оно ДОЛЖНО отображаться пустым, т.е. без маркера.	
Сообщение	От *Участника* поступает **Сообщение**, которое инициирует запуск **Процесса**. Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `MessageEventDefinition`, то данное **Стартовое событие** будет иметь тип **Сообщение**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде конверта. Текущий *Участник*, от которого было получено **Сообщение**, определяется посредством соединения графического элемента **События** с *Участником* при помощи **Потока сообщений**. В **Процессе** это отображается в рамках **Взаимодействия** (см. таблицу 10.1).	
Таймер	Определенный временной интервал или цикл (например, еженедельно по понедельникам в 9.00 утра), инициирующий возникновение **Процесса**. Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `TimerEventDefinition`, то данное **Стартовое событие** будет иметь тип Таймер. Оно ДОЛЖНО отображаться с маркером, выполненным в виде аналоговых часов.	
Условие	**Стартовое событие** данного типа возникает в случае,	

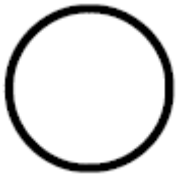
	если условия для правил, типа «Повышение индекса S&P 500 с момента открытия составляет более 10%» или «Температура свыше 300С», становятся верными. Для повторного запуска данного типа **События** условное выражение для него ДОЛЖНО переключиться в режим «*ложь*» и только затем – в режим «*истина*». Условное выражение НЕ ДОЛЖНО ссылаться на контекст данных или атрибут экземпляра **Процесс**а (т.к. новый экземпляр **Процесса** ещё не создан). Однако оно МОЖЕТ ссылаться на статические атрибуты **Процесса** и статусы объектов. Данный документ не содержит описания механизмов получения доступа к таким статусам. Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `ConditionalEventDefinition`, то данное **Стартовое событие** будет иметь тип **Условие**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде фрагмента разлинованной бумаги.	
Сигнал	Переданный из другого **Процесса** сигнал поступает в текущий **Процесс** и инициирует его запуск. Обратите внимание, что **Сигнал** – это не **Сообщение**, которое преследует совершенно другую цель. Многие **Процессы** могут содержать в составе **Стартовые события**, запускаемые с помощью одного и того же переданного **Сигнала**. Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `SignalEventDefinition`, то данное **Стартовое событие** будет иметь тип **Сигнал**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде треугольника.	
Множественный	Подразумевает наличие множества способов инициирования **Процесса**. Однако для запуска **Процесса** НЕОБХОДИМ лишь один из таких способов. Атрибуты **Стартового события** указывают, какой из видов *Триггеров* был использован. Для **Стартовых событий Множественного типа** не существует конкретного подкласса `EventDefinition`. В случае, если такое **Стартовое событие** связано более, чем с одним `EventDefiniton`, оно ДОЛЖНО отображаться с маркером, выполненным в виде пятиугольника.	
Параллельный многоэкземплярный	Подразумевает НЕОБХОДИМОСТЬ задействования множества способов инициирования **Процесса** до момента его запуска. Все типы *триггеров*, включенные в **Стартовое событие**, ДОЛЖНЫ БЫТЬ активированы до того, как начнется выполнение экземпляра **Процесса**. Для **Стартовых событий Параллельного Множественного типа** не существует конкретного подкласса `EventDefinition`. В случае, если такое **Стартовое событие** связано более, чем с одним	

	элементом `EventDefiniton`, а атрибут `parallelMultiple` **События** имеет значение «*true*», оно ДОЛЖНО отображаться с маркером, выполненным в знаке «плюс» без заливки.	

Стартовые события в Подпроцессах

В состав **Подпроцесса** может входить **Стартовое событие** лишь одного типа: Неопределенное (см. фигуру 10.82).

Таблица 10.85 –Триггер Стартового события Подроцесса

Тип триггера	Описание	Маркер
Неопределенный	**Неопределенное Стартовое событие** входит в состав всех типов **Подпроцессов:** встроенных и вызываемых (повторно используемых). Другие типы *триггеров* для **Подпроцессов** не используются, т.к. **Поток операций** (*токен*), направленный от *родительского* **Процесса**, является *триггером* для **Подпроцесса**. В случае, если **Подпроцесс** является вызываемым (повторно используемым) и содержит несколько **Стартовых событий**, некоторые из таких **Событий** МОГУТ иметь *триггеры*, но при этом они не будут использоваться в контексте данного **Подпроцесса**. Вызов этих **Стартовых событий** запускал бы экземпляры высокоуровневых **Процессов**.	

Стартовые события в Событийных Подпроцессах

Стартовое событие может быть использовано в общем потоке для запуска экземпляра **Событийного Подпроцесса**. В качестве пограничных **Событий** могут использоваться одни и те же типы **Стартовых событий** (см. таблицу 10.86), а именно: **Сообщение, Таймер, Эскалация, Ошибка, Компенсация, Условие, Сигнал, Множественное, Параллельное.**

- **Событийный Подпроцесс** ДОЛЖЕН содержать одно-единственное **Стартовое событие**.

Таблица 10.86 – Типы Стартового события Событийного Подпроцесса

Тип триггера	Описание	Маркер
Сообщение	Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `MessageEventDefinition`, то данное **Стартовое событие** будет иметь тип **Сообщение**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде конверта. • Если **Стартовое событие** данного типа прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены жирной линией (верхняя фигура). • Если **Стартовое событие** данного типа не прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы	***Прерывающее***  ***Непрерывающее*** 

	должны быть выполнены пунктиром (нижняя фигура). Текущий *Участник*, от которого было получено **Сообщение**, *определяется* посредством соединения графического элемента **События** с *Участником* при помощи **Потока сообщений**. В **Процессе** это отображается в рамках **Взаимодействия** (см. таблицу 10.1).	
Таймер	Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `TimerEventDefinition`, то данное **Стартовое событие** будет иметь тип **Таймер**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде аналоговых часов. • Если **Стартовое событие** данного типа прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены жирной линией (верхняя фигура). • Если **Стартовое событие** данного типа не прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены пунктиром (нижняя фигура).	***Прерывающее***  ***Непрерывающее*** 
Эскалация	**Событийный Подпроцесс**, в состав которого входит **Эскалация**, используется для принятия мер по ускорению выполнения **Действия** в случае, если **Действие** не выполняется в соответствии с указанными ограничениями (например, временными рамками). **Стартовое событие** данного типа может использоваться исключительно для запуска **Событийного Подпроцесса** общего потока. Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `EscalationEventDefinition`, то данное **Стартовое событие** будет иметь тип **Эскалация**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде стрелки. • Если **Стартовое событие** данного типа прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены жирной линией (верхняя фигура). • Если **Стартовое событие** данного типа не прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены пунктиром (нижняя фигура).	***Прерывающее***  ***Непрерывающее*** 
Ошибка	**Стартовое событие** данного типа может использоваться исключительно для запуска	***Прерывающее***

	Событийного Подпроцесса общего потока. Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `ErrorEventDefinition`, то данное **Стартовое событие** будет иметь тип **Ошибка**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде молнии. В силу того, что данный *триггер* имеет тип «**ошибка**», **Событийный Подпроцесс**, имеющий **Стартовое сообщение** такого типа, всегда прерывает содержащийся в нем **Процесс**.	
Компенсация	**Стартовое событие** данного типа может использоваться исключительно для запуска **Событийного Подпроцесса Компенсация** общего потока (см. подраздел «Обработчик компенсации»). Оно запускается при появлении *компенсации*. Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `CompensationEventDefinition`, то данное **Стартовое событие** будет иметь тип **Компенсация**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде двух повернутых влево треугольников. Данное **Событие** не прерывает выполнение **Процесса**, т.к. до момента вызова **Стартового события Компенсация Процесс** уже должен быть выполнен.	
Условие	Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `ConditionalEventDefinition`, то данное **Стартовое событие** будет иметь тип Условие. Оно ДОЛЖНО отображаться с маркером, выполненным в виде фрагмента разлинованной бумаги. • Если **Стартовое событие** данного типа прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены жирной линией (верхняя фигура). • Если **Стартовое событие** данного типа не прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены пунктиром (нижняя фигура).	***Прерывающее***  ***Непрерывающее*** 
Сигнал	Если **Стартовое событие** связано только с одним элементом `EventDefinition`, а этот элемент, в свою очередь, входит в подкласс `SignalEventDefinition`, то данное **Стартовое событие** будет иметь тип **Сигнал**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде треугольника. • Если **Стартовое событие** данного типа прерывает **Процесс**, содержащийся в	***Прерывающее***  ***Непрерывающее***

	Событийном Подпроцессе, его границы должны быть выполнены жирной линией (верхняя фигура). • Если **Стартовое событие** данного типа не прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены пунктиром (нижняя фигура).	
Множественный	Подразумевает наличие множества способов инициирования **Процесса**. Однако для запуска **Процесса** НЕОБХОДИМ лишь один из таких способов. Для **Стартовых событий Множественного типа** не существует конкретного подкласса EventDefinition. Если **Стартовое событие** связано более чем с одним элементом `EventDefinition`, то данное **Стартовое событие** будет иметь **тип Множественный**. Оно ДОЛЖНО отображаться с маркером, выполненным в виде треугольника. • Если **Стартовое событие** данного типа прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены жирной линией (верхняя фигура). • Если **Стартовое событие** данного типа не прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены пунктиром (нижняя фигура).	***Прерывающее***  ***Непрерывающее*** 
Параллельный Множественный	Подразумевает НЕОБХОДИМОСТЬ задействования множества способов инициирования **Процесса** до момента его запуска. Все эти способы НЕОБХОДИМЫ для запуска **Процесса**. Для **Стартовых событий Параллельного Множественного** типа не существует конкретного подкласса `EventDefinition`. В случае, если такое **Стартовое событие** связано более, чем с одним элементом EventDefiniton, а атрибут `parallelMultiple` **События** имеет значение «*true*», оно ДОЛЖНО отображаться с маркером, выполненным в знаке «плюс» без заливки. • Если **Стартовое событие** данного типа прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены жирной линией (верхняя фигура). • Если **Стартовое событие** данного типа не прерывает **Процесс**, содержащийся в **Событийном Подпроцессе**, его границы должны быть выполнены пунктиром (нижняя фигура).	***Прерывающее***  ***Непрерывающее*** 

Атрибуты Стартового события

Элемент **StartEvent** наследует атрибуты и ассоциации элемента `CatchEvent` (см. таблицу 10.82). Таблица 10.87 содержит информацию о дополнительных атрибутах элемента **StartEvent**.

Таблица 10.87 – Атрибуты элемента StartEvent

Название атрибута	Описание/использование
isInterrupting: boolean = true	Данный атрибут подходит лишь для **Стартовых событий**, работающих с данными **Подпроцессов**, и игнорируется другими **Стартовыми событиями**. Атрибут указывает на то, должен ли быть отменен **Подпроцесс**, окружающий работающий с данными **Подпроцесс**, или нет. Если выполнение окружающего **Подпроцесса** не отменяется, то множественные экземпляры работающего с данными **События** будут двигаться одновременно. Данный атрибут не может быть использован для **Сообщений** типов: **Ошибка** (т.к. значение данного атрибута всегда равно «*true*») и **Компенсация** (где он не может быть применен в принципе).

Соединение с Потоком операций

Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они МОГУТ являться *целями* и *источниками* **Потока операций**, обратитесь к пункту 7.5.1 «Правила Соединения Потоков Операций».

- **Стартовое Событие** НЕ ДОЛЖНО являться целью **Потока Сообщений**. Оно НЕ ДОЛЖНО БЫТЬ соединено с каким-либо *Входящим* **Потоком Операций**.
    - Исключением является ситуация, когда **Стартовое событие** используется в Развернутом **Подпроцессе** и соединяется с границами данного **Подпроцесса**. В этом случае **Поток операций**, относящийся к **Процессу** более верхнего уровня, МОЖЕТ БЫТЬ соединен со **Стартовым событием**, а не с границей **Подпроцесса**.
- **Стартовое событие** ДОЛЖНО являться источником **Потока операций**.
- Множественный **Поток Операций** МОЖЕТ начинаться со **Стартового события**. Необходимо создание нового параллельного маршрута для каждого **Потока операций**, использующего в качестве источника **Стартовое событие**.
    - Атрибут `conditionExpression` для всех *Исходящих* **Потоков Операций** ДОЛЖЕН иметь значение «None».
    - В случае, если **Процесс** не содержит **Стартового События**, то каждый **Э**лемент потока, не имеющий *Входящего* **Потоков Операций**, ДОЛЖЕН стать началом независимого параллельного маршрута.
    - Каждый маршрут обладает уникальным *Токеном*, пересекающим **Поток Операций**.

Соединение с Потоком сообщений

Примечание: Все **Потоки сообщений** ДОЛЖНЫ соединять два разных **Пула**. Они МОГУТ быть присоединены к границам **Пула**, а также к *Элементам потока* внутри этого **Пула**. Они НЕ ДОЛЖНЫ использоваться для соединения двух элементов внутри одного **Пула**.

Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они могут являться целями **Потока сообщений**, обратитесь к пункту 7.5.2 «Правила Соединения Потоков Сообщений».

- **Стартовое Событие** МОЖЕТ БЫТЬ целью **Потока Сообщений** и иметь как несколько *Входящих* **Потоков Сообщений,** так и ни одного. Любой **Поток Сообщений**, являющийся *входящим* для

Стартового События, представляет собой механизм запуска (*триггер*) **Процесса**. Для запуска нового **Процесса** требуется лишь один из триггеров.

- **Стартовое Событие** НЕ ДОЛЖНО являться источником **Потока Сообщений**; оно также НЕ ДОЛЖНО соединяться с какими-либо *Исходящими* **Потоком Сообщений.**

10.4.3. Конечное событие

Как видно из названия, **Конечное Событие** указывает на то, в какой точке завершается тот или иной **Процесс.** В контексте **Потока Операций Конечное Событие** завершает ход **Процесса**; это означает, что никакой **Поток Операций** (в том числе *Исходящий* **Поток Операций**) не может быть соединен с **Конечным Событием**.

Изображается в виде круга со свободным центром (установленное в **BPMN** отображение графического элемента **Событие**), предназначенным для дифференцировки внутренними маркерами различных типов **Событий.**

- **Конечное событие** представляет собой круг, который ДОЛЖЕН БЫТЬ выполнен одинарной жирной линией (см. фигуру 10.71).
- Текст, цвет, размер, а также линии, используемые для изображения **Конечного события**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».
    - Толщина линии ДОЛЖНА БЫТЬ жирной настолько, чтобы без труда можно было бы отличить **Конечное событие** от **Стартового** и **Промежуточного.**

Фигура 10.71 – Графический элемент Конечное событие

Конечное событие является точкой, где заканчивается *Токен,* сформированный на том же уровне П**роцесса** при запуске **Стартового события**. В случае, если в Параллельном **потоке операций** запланировано **Конечное событие**, то *Токены* будут заканчиваться в момент соединения с **Конечным событием**. Все *Токены*, сформированные внутри **Процесса**, ДОЛЖНЫ завершиться в точке расположения **Конечного события** до полного окончания данного **Процесса**. В случаях, когда **Процесс** является **Подпроцессом**, он может быть завершен заранее посредством прерывающего **Промежуточного события**. В подобных случаях *Токены* будут завершены **Промежуточным событием**, соединенным с границами **Подпроцесса.**

Конечное событие имеет следующие особенности:

- **Процессы** МОГУТ содержать несколько **Конечных событий** на каждом уровне.
- **Конечное событие** является ОПЦИОНАЛЬНЫМ. **Процесс** определенного уровня - **Процесс** или Развернутый **Подпроцесс** – МОЖЕТ содержать в своем составе **Конечное событие**, однако, это не является необходимым условием.
    - В случае, если диаграмма не содержит **Конечного события**, то скрытое **Конечное событие** НЕ ДОЛЖНО оказывать влияние на результат **Процесса.**
    - В случае, если диаграмма содержит **Стартовое событие**, то ДОЛЖНО БЫТЬ по-меньшей мере и одно **Конечное событие**.
    - В случае, если диаграмма не содержит **Конечного события**, то *Элементы Потока*, не имеющие *Исходящих* **потоков операций** (не являющиеся источниками **Потоков операций**), указывают на окончание какого-либо маршрута, содержащегося в **Процессе**. Однако **Процесс** НЕ ДОЛЖЕН БЫТЬ завершен до тех пор, пока к концу не придут все параллельные маршруты.

Примечание: Диаграмма **Процесса** может содержать **Процессы** разных уровней (например, *Развернутый* **Подпроцесс** или **Действие Вызов**, используемое для вызова других **Процессов**). Использование **Стартового** и **Конечного событий** является свободным для **Процессов** любого уровня, содержащихся в диаграмме.

В случае, если **Конечное событие** не включено в состав **Процесса**, то *Токен*, содержащийся в данном **Процессе**, соединяется с Элементом потока, чей маршрут ведет к завершению **Процесса**, и завершается в момент окончания обработки данных (окончания маршрута) этим Элементом потока (как если бы *токен* продолжил движение до **Конечного события**). В момент, когда все *токены* данного экземпляра **Процесса** заканчиваются, **Процесс** считается завершенным.

End Event Results Результаты Конечного события

По типу *результата* **Конечные события** делятся на: *Неопределенные*, **Сообщения**, **Эскалации, Ошибки, Отмены**, **Компенсации, Сигналы, Завершения, Множественные**. Каждый из этих девяти типов **Конечного события** определяет то, каким образом **Поток операций** достигнет этого **События**.

Таблица 10.88 – Типы Конечного события

Тип результата	Описание	Маркер
Неопределенное	Данный тип **Конечного события** не подразумевает какой-то определенный *результат*. Для **Конечных событий Неопределенного типа** не существует конкретного подкласса `EventDefinition`. Если **Конечное событие** не связано ни с одним элементом `EventDefinition`, то данное **Событие** отображается на диаграмме пустым (без маркера).	
Сообщение	Данный тип **Конечного события** служит для указания того, что *Участник* отправил **Сообщение** в момент завершения **Процесса**. Текущий *Участник*, от которого было получено **Сообщение**, определяется посредством соединения графического элемента **События** с *Участником* при помощи **Потока сообщений**. В **Процессе** это отображается в рамках **Взаимодействия** (см. таблицу 10.1).	
Ошибка	Данный тип **Конечного события** служит для указания необходимости формирования определенной **Ошибки**. В результате прерываются все действующие в текущий момент маршруты **Подпроцесса**. На **Ошибку** реагирует соответствующее **Промежуточное событие Ошибка**, имеющее такое же значение `errorCode` или не имеющее `errorCode`. Иерархически, оно должно быть соединено с границей ближайшего родительского **Действия**, в котором находится. В случае, если в иерархии не существует ни одного **Действия**, включающего такое **Промежуточное событие**, поведение **Процесса** непредсказуемо. В этом случае для выполнения **Процесса** может быть определена дополнительная обработка **Ошибки**, представляющая собой завершение *экземпляра* **Процесса**.	
Эскалация	Данный тип **Конечного события** служит для указания того, что должна быть запущена *Эскалация*. Это не	

	влияет на ход других действующих маршрутов, - их выполнение продолжается. На *Эскалацию* может отреагировать соответствующее **Промежуточное событие Эскалация**, имеющее такое же значение `escalationCode` или не имеющее `escalationCode`. Иерархически, оно должно быть соединено с границей ближайшего родительского **Действия**, в котором находится. В случае, если в иерархии не существует ни одного **Действия**, включающего такое **Промежуточное событие**, поведение **Процесса** непредсказуемо.	
Отмена	Данный тип **Конечного события** предназначен для использования в **Подпроцессе Транзакция**. Оно служит для указания того, что должна быть отменена **Транзакция**, а также для дальнейшего запуска **Промежуточного события Отмена**, присоединенного к границе **Подпроцесса**. Такое **Конечное событие** также указывает на то, что субъектам, включенным в **Транзакцию**, должно быть отправлено **Сообщение** `TransactionProtocol Cancel`.	
Компенсация	Данный тип **Конечного события** служит для указания необходимости компенсации. Если указанное **Действие** выполнено успешно, оно будет компенсировано. Такое **Действие** ДОЛЖНО БЫТЬ видимо из **Конечного события Компенсация**, т.е. ДОЛЖНО БЫТЬ удовлетворено одно из следующих требований: • **Конечное событие Компенсация** включено в состав *Стандартного потока операций* на том же уровне **Подпроцесса**. • **Конечное событие Компенсация** включено в состав **Событийного Подпроцесса Компенсация**, который содержится в том же **Подпроцессе**, что и **Действие**. • Если не указано ни одного **Действия**, то компенсируются все успешно выполненные **Действия**, видимые из **Конечного события Компенсация**. Компенсация производится в порядке, обратном порядку **Потока операций**. Видимость **Действия** подразумевает: ○ **Конечное событие Компенсация** включено в состав *Стандартного потока операций* на том же уровне **Подпроцесса**, что и **Действия**. ○ **Конечное событие Компенсация** включено в состав **Событийного Подпроцесса Компенсация**, который содержится в том же **Подпроцессе**, что и **Действия**. Для того, чтобы **Действие** было компенсировано, на границе оно ДОЛЖНО иметь **Событие Компенсация** или содержать работающий с событиями **Подпроцесс Компенсация.**	
Сигнал	Данный тип **Конечного события** служит для указания	

	того, что в момент завершения маршрута будет передан `Сигнал`. Обратите внимание, что `Сигнал`, передаваемый в какой-либо принимающий `сигналы` **Процесс**, может пересекать уровни этого **Процесса** или границы **Пулов**, однако, не является при этом **Сообщением**, которое все же имеет конкретные *источники* и *цели*. Информацию об атрибутах **Конечного события Сигнал** смотрите в соответствующем разделе.	
Завершение	Данный тип **Конечного события** служит для указания того, что выполнение всех **Действий Процесса** должно быть немедленно прекращено. Это относится как к экземплярам **Действия**, так и к *Многоэкземплярным* **Действиям. Процесс** завершается без возможности *компенсации* или *обработки* **Событий**.	
Множественный	Данный тип **Конечного события** служит для указания того, что завершение **Процесса** имеет несколько результатов. Все эти результаты имеют быть (к примеру, может быть отправлено несколько сообщений). Для **Конечных событий Множественного типа** не существует конкретного подкласса `EventDefinition`. **Если Конечное** событие связано с более чем одним соответствующим элементом `EventDefinition`, то данное **Событие** содержит маркер, выполненный в виде пятиугольника.	

Соединение с Потоком операций

Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они МОГУТ являться *целями* и *источниками* Потока операций, обратитесь к пункту 7.5.1 «Правила Соединения Потоков Операций».

- Конечное событие ДОЛЖНО являться целью Потока сообщений.
- Конечное событие МОЖЕТ иметь множество Входящих потоков операций.

Маршруты, направленные к **Конечному Событию**, могут быть как параллельными, так и альтернативными. Для большего удобства моделирования каждый маршрут МОЖЕТ соединяться с отдельным **Конечным Событием**. **Конечное Событие** используется в качестве воронки, куда попадают все прибывшие к нему *токены*. Все *токены*, сформированные при запуске **Стартового События Процесса**, в конечном итоге ДОЛЖНЫ прибыть к **Конечному Событию**. **Процесс** считается незавершенным до тех пор, пока не будут завершены все содержащиеся в нем *токены*.

- **Конечное Событие** НЕ ДОЛЖНО являться источником **Потока Операций**. Оно НЕ ДОЛЖНО иметь *Исходящих* **Потоков Операций**.
    - Исключением является ситуация, когда **Конечное событие** включено в состав Развернутого **Подпроцесса** и соединено с границами этого **Подпроцесса**. В данном случае, **Поток операций**, направленный от **Процесса** более высокого уровня, на выходе МОЖЕТ быть присоединен к **Конечному событию**, а не границам **Подпроцесса**.

Соединение с Потоком сообщений

Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они могут являться целями **Потока сообщений**, обратитесь к пункту 7.5.2 «Правила Соединения Потоков Сообщений».

Примечание: Все **Потоки сообщений** ДОЛЖНЫ соединять два разных **Пула**. Они МОГУТ быть присоединены к границам **Пула**, а также к Элементам потока внутри этого **Пула**. Они НЕ ДОЛЖНЫ использоваться для соединения двух элементов внутри одного **Пула**.

- **Конечное событие** НЕ ДОЛЖНО являться целью **Потока сообщений**. Оно не должно иметь *Входящих* **Потоков сообщений**.
- **Конечное событие** МОЖЕТ являться источником **Потока сообщений** и иметь как несколько *Исходящих* **Потоков сообщений**, так и ни одного. Любой *Исходящий* **Поток сообщений**, направленный от **Конечного события**, будет содержать **Сообщение**, отсылаемое при запуске этого **События**.
    - Если от **Конечного события** отходит несколько *Исходящих* **Потоков сообщений**, то атрибут Result **Конечного сообщения** ДОЛЖЕН иметь значение `Message` или `Multiple`.
    - Если от **Конечного события** отходит более одного *Исходящего* **Потока сообщений**, то атрибут Result **Конечного сообщения** ДОЛЖЕН иметь значение `Multiple`.

10.4.4. Промежуточное событие

Промежуточное событие происходит на отрезке **Процесса,** ограниченном Стартовым и Конечным событиями. **Промежуточное событие** влияет на ход **Процесса**, однако, не может являться его началом или непосредственным завершением. **Промежуточное событие** может быть задействовано для того, чтобы:

- Указать отрезок **Процесса**, где ожидается получение **Сообщений** или планируется их отправка.
- Указать отрезок **Процесса**, на котором ожидаются задержки.
- Нарушить ход *Стандартного потока операций* при помощи *исключений*.
- Указать на необходимость дополнительной работы в рамках *компенсации*.

Изображается в виде круга со свободным центром (установленное в **BPMN** отображение графического элемента **Событие**), предназначенным для дифференцировки внутренними маркерами различных типов **Событий**.

- **Промежуточное событие** представляет собой круг, который ДОЛЖЕН БЫТЬ выполнен двойной тонкой линией (см. фигуру 10.72).
- Текст, цвет, размер, а также линии, используемые для изображения **Промежуточного события**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм», с учетом того, что линия ДОЛЖНА быть двойной для того, чтобы без труда можно было отличить **Промежуточное событие** от Стартового или Конечного.

Фигура 10.72 – Графический элемент Промежуточное событие

Промежуточное событие может быть использовано, в частности, для отображения исключений или *компенсации*, *входящих* в **Процесс.** Для этого **Промежуточное событие** изображают на границе необходимой **Задачи** или **Подпроцесса** (как свернутого, так и развернутого); Исходящие **потоки операций** при этом могут двигаться в любом направлении. Однако для большей прозрачности диаграмм разработчикам рекомендуется выбирать определенную точку соединения. Например, в случае, если диаграмма располагается горизонтально, то **Промежуточное событие** может быть соединено с нижней границей **Действия**, а **Поток операций** направлен строго вниз и вправо. Если же диаграмма располагается вертикально, то **Промежуточное событие** может быть соединено как с правой, так и с левой сторонами **Действия**, а **Поток операций** направлен либо вправо, либо влево, и вниз.

Триггеры Промежуточного события

BPMN выделяет двенадцать типов **Промежуточных событий**: `Неопределенное`, **Сообщение, Таймер, Эскалация, Ошибка, Отмена, Компенсация, Условие, Связь, Сигнал, Множественной, Неопределенное Множественное**. Для определения типа **Промежуточного события** используются различные маркеры, позволяющие отличить данный тип **События** от другого.

Существует два способа использования **Промежуточного события**:

- **Промежуточное событие**, включенное в состав *Стандартного потока операций* какого-либо **Процесса**, может быть использовано для одной из двух целей: такое событие может, как реагировать на *триггер*, так и воздействовать на другие **События**.
- **Промежуточное событие**, присоединенное к границе **Действия**, может быть использовано только в качестве реагирующего на *триггер* **События**.

Промежуточное событие в составе Стандартного потока операций

Когда *токен* прибывает к **Промежуточному событию**, входящему в состав *Стандартного потока операций*, наблюдается одно из нижеописанных явлений:

- Если **Промежуточное событие** запускает *триггер* (приводит к какому-то результату), то такой *триггер* срабатывает немедленно (например, отсылается **Сообщение**); при этом дальнейший ход *токена* будет совпадать с ходом основного *Исходящего потока операций*.
- Если **Промежуточное событие** реагирует на *триггер*, то *токен* не покидает данное **Событие** до тех пор, пока не появится *триггер* (например, получение сообщения). После этого дальнейший ход *токена* будет совпадать с ходом основного *Исходящего потока операций*.

Стандартный поток операций предполагает использование десяти или двенадцати типов **Промежуточного события** (см. таблицу 10.89).

Таблица 10.89 – Промежуточное событие в составе Стандартного потока операций

Тип триггера	Описание	Маркер
Неопределенное	Данный тип **Промежуточного события** может быть использован только в *Стандартном потоке операций*. Оно НЕ МОЖЕТ использоваться на границе **Действия**. Несмотря на то, что маркер для данного типа **Промежуточного события** отсутствует, оно относится к *определяющим результат* (*запускающим триггер*). Такая характеристика необходима при создании методологий, использующих **События** для отображения изменения хода **Процесса**. Для **Промежуточных событий** Неопределенного типа не существует конкретного подкласса `EventDefinition`. Если данное **Промежуточное событие** не связано ни с одним элементом `EventDefinition`, то оно отображается на диаграмме пустым (без маркера).	**Возбуждающее триггер** 
Сообщение	Данный тип **Промежуточного события** может быть использован как для отправки, так и для получения сообщений. При отправке **Сообщения маркер Промежуточного События** ДОЛЖЕН иметь заливку (верхняя фигура). При получении **Сообщения** маркер **Промежуточного**	**Возбуждающее триггер** 

	События ДОЛЖЕН быть пустым (нижняя фигура). Использование одного из этих маркеров определяет либо дальнейшее выполнение **Процесса** после получения сообщения, либо изменение хода **Процесса** с целью обработки исключения. Текущий *Участник*, от которого было получено **Сообщение**, определяется посредством соединения графического элемента **События** с *Участником* при помощи **Потока сообщений**. В **Процессе** это отображается в рамках **Взаимодействия** (см. таблицу 10.1).	**Обрабатывающее триггер** 
Таймер	В *Стандартном потоке операций* **Промежуточное событие** данного типа используется для того, чтобы отложить время вызова **События** (изменить дату/время или параметры цикличность, к примеру, каждый понедельник в 9.00 утра). **Промежуточное событие** Таймер ДОЛЖНО отображаться с маркером, выполненным в виде аналоговых часов.	**Обрабатывающее триггер** 
Эскалация	В *Стандартном потоке операций* **Промежуточное событие** данного типа используется для вызова эскалации. Поскольку данное **Событие** запускает триггер, помещенный в нем маркер должен иметь заливку.	**Возбуждающее триггер** 
Компенсация	В *Стандартном потоке операций* **Промежуточное событие** данного типа используется для указания необходимости *компенсации*. Следовательно, оно запускает **Событие Компенсация**, поэтому помещенный в нем маркер ДОЛЖЕН иметь заливку. Если указанное **Действие** выполнено успешно, оно будет *компенсировано*. Такое **Действие** ДОЛЖНО БЫТЬ видимо из **Промежуточного события Компенсация**, т.е. ДОЛЖНО БЫТЬ удовлетворено одно из следующих требований: • **Промежуточное событие Компенсация** включено в состав **Стандартного потока** операций на том же уровне **Подпроцесса**. • **Промежуточное событие Компенсация** включено в состав **Событийного Подпроцесса Компенсация**, который содержится в том же **Подпроцессе**, что и **Действие**. Если не указано ни одного **Действия**, то *компенсируются* все успешно выполненные **Действия**, видимые из **Конечного события Компенсация**. *Компенсация* производится в порядке, обратном порядку **Потока операций**. Видимость **Действия** подразумевает: • **Промежуточное событие Компенсация** включено в состав *Стандартного потока операций* на том же уровне **Подпроцесса**, что и	**Возбуждающее триггер** 

	Действия. • **Промежуточное событие Компенсация** включено в состав **Событийного Подпроцесса Компенсация**, который содержится в том же **Подпроцессе**, что и **Действия**. Для того, чтобы **Действие** было компенсировано, на границе оно ДОЛЖНО иметь **Событие Компенсация** или содержать работающий с событиями **Подпроцесс Компенсация.**	
Условие	**Промежуточное событие** данного типа запускается в том случае, если какое-либо условие становится верным («*true*»). Условие представляет собой `Expression` тип. Атрибуты элемента `Expression` тип смотрите в соответствующем разделе.	**Возбуждающее триггер** 
Связь	Данный тип **Промежуточного события** может быть использован только в *Стандартном потоке операций*. Оно НЕ МОЖЕТ использоваться на границе **Действия.** Посредством **Связи** соединяются два участка **Процесса**. **События типа Связь** могут применяться для отображения цикличности либо для того, чтобы избежать использования слишком длинных цепочек **Потоков операций**. Использование **Событий Связь** недопустимо для отображения любых соединений на одном уровне **Процесса** (например, с их помощью нельзя соединять родительский **Процесс** с **Подпроцессом**). Парные **Промежуточные события** этого типа могут быть использовать в качестве «соединителя страниц». Такое соединение удобно в тех случаях, когда необходимо напечатать **Процесс**, размещенный на нескольких страницах. Они также могут использоваться для перехода к определенным точкам (в качестве общих объектов «GOTO») в рамках какого-либо одного уровня **Процесса**. В качестве источников может быть использовано любое количество таких **Событий**, однако, в качестве цели может использоваться лишь одно **Промежуточное событие типа Связь**. Если **Событие** данного типа используется в качестве запускающего *триггер*, содержащийся в нем маркер должен иметь заливку (верхняя фигура). Если оно используется в качестве реагирующего на *триггер*, маркер отображается без заливки (нижняя фигура).	**Возбуждающее триггер**  **Обрабатывающее триггер** 
Сигнал	Данный тип **Промежуточного события** используется для отправки и получения сигнала. **Событие типа Сигнал** используется для отображения общих взаимоотношений как в рамках одного уровня **Процесса**, так и между его уровнями, а также между **Пулами** и диаграммами **бизнес-процессов**. По принципу действия **Событие Сигнал**, описанное в	**Возбуждающее триггер**  **Обрабатывающее**

	BPMN, схоже с сигнальной ракетой, которую запускают в небо для того, чтобы кто-то, кому это небезразлично, мог заметить её и, конечно, каким-либо образом отреагировать. Данный тип **Промежуточного события** имеет источник, но не имеет специально предназначенной цели. Данное **Событие** может отправлять и получать сигналы, если оно входит в состав *Стандартного потока операций*. Данное **Событие** может лишь получать **Сигналы**, если оно присоединено к границе **Действия**. **Событие типа Сигнал** отличается от **События** типа **Ошибка** тем, что посредством **Сигнала** указывается более общее и не являющееся ошибкой условие для прерывания выполнения **Действия** (например, успешное выполнение другого **Действия**). Область применения **Сигнала** шире, чем у **Ошибки**. Если **Событие** данного типа используется в качестве запускающего триггер, содержащийся в нем маркер должен иметь заливку (верхняя фигура). Если оно используется в качестве реагирующего на *триггер*, маркер отображается без заливки (нижняя фигура). Атрибуты элемента `Signal` смотрите в соответствующем разделе.	**триггер** 
Множественный	Появление на диаграмме **Промежуточного события** данного типа означает, что для него используется несколько *триггеров*. Если данное **Событие** включено с состав *Стандартного потока операций*, оно может, как реагировать на *триггер*, так и запускать его. Если данное **Событие** присоединено к границе **Действия**, оно может лишь реагировать на *триггер*. Если данное **Событие** используется в качестве реагирующего на *триггер*, для него НЕОБХОДИМО указать один-единственный *триггер*. В этом случае маркер отображается без заливки (нижняя фигура). Если данное **Событие** используется в качестве *запускающего триггер* (так же, как и **Множественное Конечное событие**), то все указанные для него *триггеры* будут запущены. В этом случае маркер имеет заливку (верхняя фигура). Для **Промежуточных событий Множественного типа** не существует конкретного подкласса `EventDefinition`. Если **Промежуточное событие** связано с более, чем одним соответствующим элементом `EventDefinition`, то данное **Событие** содержит маркер, выполненный в виде пятиугольника.	**Возбуждающее триггер**  **Обрабатывающее триггер** 
Параллельный Множественный	Появление на диаграмме **Промежуточного события** данного типа означает, что для него используется несколько *триггеров*. Если данное **Событие** включено с состав *Стандартного потока операций*, оно может лишь реагировать на *триггер*. Если данное **Событие** присоединено к границе **Действия**, оно также может лишь реагировать на *триггер*. В отличие от **Множественного Промежуточного события**, все	**Обрабатывающее триггер** 

	указанные для **Параллельного Множественного Промежуточного события** *триггеры* ДОЛЖНЫ БЫТЬ запущены. Оно должно отображаться с маркером, выполненным в виде знака «плюс» без заливки. Для **Промежуточных событий Параллельного Множественного типа** не существует конкретного подкласса `EventDefinition`. Если **Промежуточное событие** связано с более чем одним соответствующим элементом `EventDefinition`, а значение атрибута `parallelMultiple` данного **События** равно «*true*»,то оно содержит соответствующий маркер.	

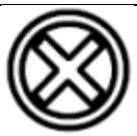
Промежуточное событие, соединенное с Действием

Таблица 10.90 содержит информацию о **Промежуточных событиях**, которые могут быть соединены с границами **Действий**.

Таблица 10.90 – Типы Промежуточного события, соединенного с Действием

Тип триггера	Описание	Маркер
Сообщение	**Сообщение** поступает от Участника и запускает **Событие**. В случае, если **Промежуточное событие типа Сообщение** присоединено к границе **Действия**, то в момент его запуска *Стандартный поток операций* становится *Потоком исключений*. Граница **События** данного типа, прерывающего выполнение **Действия**, к которому присоединено, отображается целой (верхняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*true*». Граница **События** данного типа, не прерывающего выполнение **Действия**, к которому присоединено, выполнена пунктиром (нижняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*false*». Текущий *Участник*, от которого было получено **Сообщение**, определяется посредством соединения графического элемента **События** с *Участником* при помощи **Потока сообщений**. В **Процессе** это отображается в рамках **Взаимодействия** (см. таблицу 10.1).	**Прерывающее**  **Непрерывающее** 
Таймер	Для запуска **Промежуточного события Таймер** могут быть установлены определенные параметры даты/времени или цикличности (к примеру, каждый понедельник в 9.00 утра). В случае, если **Промежуточное событие Таймер** присоединено к границе **Действия**, то в момент его запуска *Стандартный поток операций* становится *Потоком исключений.* Граница **События** данного типа, прерывающего	**Прерывающее**  **Непрерывающее**

	выполнение **Действия**, к которому присоединено, отображается целой (верхняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*true*». Граница **События** данного типа, не прерывающего выполнение **Действия**, к которому присоединено, выполнена пунктиром (нижняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*false*».	
Эскалация	**Промежуточное событие Эскалация** используется в **Процессе** для обработки определенной Эскалации. Если такое **Событие** присоединено в границе **Действия**, оно реагирует на Эскалацию. В отличие от **Ошибки**, по умолчанию прерывающей выполнение **Действия**, Эскалация не оказывает такого влияния на **Действие**, к которому присоединено. Однако разработчик модели может включить в состав **Процесса** и прерывающее **Действие Промежуточное событие Эскалация**. Графически это отображается следующим образом: • Граница **События** данного типа, прерывающего выполнение **Действия**, к которому присоединено, отображается целой (верхняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута cancelActivity этого **Действия** по умолчанию равно «*true*». • Граница **События** данного типа, не прерывающего выполнение **Действия**, к которому присоединено, выполнена пунктиром (нижняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута cancelActivity этого **Действия** по умолчанию равно «*false*».	**Прерывающее**  **Непрерывающее** 
Ошибка	**Промежуточное событие Ошибка**, реагирующее на *триггер*, может лишь присоединяться к границе **Действия**, следовательно, оно НЕ МОЖЕТ входить в состав *Стандартного потока операций*. Использование данного типа **События** в таком контексте подразумевает то, что оно реагирует на определенную ошибку (ловит ошибку). В случае, если ошибка не указана, оно также может реагировать на любую появившуюся ошибку. Обратите внимание, что **Промежуточное событие Ошибка** всегда прерывает выполнение **Действия**, к которому присоединено. Это значит, что не существует **События** данного типа, которое не прерывало бы выполнение **Действия**. Граница такого **События** отображается целой.	**Прерывающее** 
Отмена	**Промежуточное событие Отмена** используется в рамках **Подпроцесса Транзакция**. **Событие** данного типа ДОЛЖНО БЫТЬ присоединено к границе	**Прерывающее**

	Подпроцесса. Оно ДОЛЖНО запускаться в том случае, если *Поток операций* достигает **Конечного события Отмена** данного **Подпроцесса**. Оно также ДОЛЖНО запускаться в случае, если в ходе выполнения Транзакции получено сообщение об отмене (от `TransactionProtocol`). Обратите внимание, что **Промежуточное событие Отмена** всегда прерывает выполнение **Действия**, к которому присоединено. Это значит, что не существует **События** данного типа, которое не прерывало бы выполнение **Действия**. Граница такого **События** отображается целой.	
Компенсация	При присоединении к границе **Действия Промежуточное событие Компенсация** реагирует на **Событие Компенсация,** следовательно, маркер этого **События** ДОЛЖЕН БЫТЬ без заливки. **Событие** данного типа активируется запущенной *компенсацией*, целью которой является **Действие**. Когда **Событие** запущено, происходит выполнение **Действия Компенсация**, связанного с данным **Событием**. Обратите внимание, что **Событие Компенсация** не может быть ни прерывающим, ни непрерывающим. Оно запускается только после завершения выполнения **Действия**, к которому присоединено, и не может его прерывать. Граница такого **События** всегда отображается целой.	
Условие	**Промежуточное событие Условие** запускается, когда определенное условие становится верным («*true*»). Условие представляет собой `Expression` тип. Атрибуты элемента `Expression` тип смотрите в соответствующем разделе. При запуске Условия, присоединенного к границам **Действи**я, *Стандартный поток операций* становится *Потоком исключений.* Граница **События** данного типа, прерывающего выполнение **Действия,** к которому присоединено, отображается целой (верхняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*true*». Граница **События** данного типа, не прерывающего выполнение **Действия**, к которому присоединено, выполнена пунктиром (нижняя фигура). Обратите внимание, что согласно **BPMN,** значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*false*».	**Прерывающее**  **Непрерывающее** 
Сигнал	**Промежуточное событие Сигнал**, присоединенное к границе **Действия,** используется для получения сигнала. При запуске Сигнала, присоединенного к границам **Действия**, *Стандартный поток операций* становится *Потоком исключений*. **Событие типа Сигнал** отличается от **События** типа **Ошибка** тем, что	**Прерывающее**  **Непрерывающее**

	посредством Сигнала указывается более общее и не являющееся ошибкой условие для прерывания выполнения **Действия** (например, успешное выполнение другого **Действия**). Область применения Сигнала шире, чем у Ошибки. Граница **События** данного типа, прерывающего выполнение **Действия**, к которому присоединено, отображается целой (верхняя фигура). Обратите внимание, что согласно **BPMN,** значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*true*». Граница **События** данного типа, не прерывающего выполнение **Действия**, к которому присоединено, выполнена пунктиром (нижняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*false*».	
Множественный	Появление на диаграмме **Промежуточного события** данного типа означает, что для него используется несколько *триггеров*. Если данное **Событие** присоединено к границе **Действия**, оно может лишь реагировать на *триггер*. Если данное **Событие** присоединено к границе **Действия**, оно может лишь реагировать на *триггер*. Если для данного **События** НЕОБХОДИМ один-единственный *триггер*, то маркер данного **События** в момент его запуска будет отображаться без заливки, а *Стандартный поток операций* станет *Потоком исключений*. Для **Промежуточных событий Множественного тип**а не существует конкретного подкласса `EventDefinition`. Если данное **Промежуточное событие** связано с более чем одним соответствующим элементом `EventDefinition`, то данное **Событие** содержит маркер, выполненный в виде пятиугольника. Граница **События** данного типа, прерывающего выполнение **Действия**, к которому присоединено, отображается целой (верхняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*true*». Граница **События** данного типа, не прерывающего выполнение **Действия**, к которому присоединено, выполнена пунктиром (нижняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*false*».	**Прерывающее**  **Непрерывающее** 
Параллельный Множественный	Появление на диаграмме **Промежуточного события** данного типа означает, что для него используется несколько *триггеров*. Если данное **Событие** присоединено к границе **Действия**, оно может лишь реагировать на триггер. В отличие от **Множественного**	**Прерывающее** 

	Промежуточного события, все указанные для **Параллельного Множественного Промежуточного события** *триггеры* ДОЛЖНЫ БЫТЬ запущены. Оно должно отображаться с маркером, выполненным в виде знака «плюс» без заливки. Для **Промежуточных событий Параллельного Множественного типа** не существует конкретного подкласса `EventDefinition`. Если **Промежуточное событие** связано с более чем одним соответствующим элементом `EventDefinition`, а значение атрибута `parallelMultiple` данного **События** равно «*true*»,то оно содержит соответствующий маркер. Граница **События** данного типа, прерывающего выполнение **Действия**, к которому присоединено, отображается целой (верхняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута cancelActivity этого **Действия** по умолчанию равно «*true*». Граница **События** данного типа, не прерывающего выполнение **Действия**, к которому присоединено, выполнена пунктиром (нижняя фигура). Обратите внимание, что согласно **BPMN**, значение атрибута `cancelActivity` этого **Действия** по умолчанию равно «*false*».	**Непрерывающее** 

Атрибуты Событий, присоединенных к границам элементов

Элемент `BoundaryEvent` - **Промежуточное событие**, присоединенное к границам других Элементов потока - наследует атрибуты и ассоциации элемента `CatchEvent` (см. таблицу 8.44). Таблица 10.91 содержит информацию о дополнительных атрибутах и ассоциациях элемента `BoundaryEvent`.

Таблица 10.91 – Атрибуты элемента BoundaryEvent

Название атрибута	Описание/использование
attachedTo: Activity	Данный атрибут указывает на **Действие**, к которому присоединено **Промежуточное событие.**
cancelActivity: boolean	Данный атрибут указывает на то, должно ли быть отменено выполнение **Действия** или нет. Другими словами, он указывает на то, какой тип имеет присоединенное к **Действию** и *реагирующее на триггер* **Промежуточное событие:** `Ошибка` или `Эскалация`. Если отмены выполнения **Действия** не происходит, то многочисленные экземпляры **События** могут выполняться одновременно. Данный атрибут не может быть использован для **Событий типа Ошибка** (где его значение всегда равно «*true*») или **Компенсация** (для которого он не используется в принципе).

Таблица 10.92 содержит информацию, которая позволяет определить, возможно ли использование атрибута `cancelActivity` для **Промежуточного события**, присоединенного к **Действию**, в зависимости от значения `EventDefinition`, на которое это **Событие** реагирует.

Таблица 10.92 – Возможные значения атрибута cancelActivity

Тип Промежуточного события	Возможные значения атрибута cancelActivity
Неопределенный	Нет значений, т.к. данное **Событие** не может быть присоединено к границе **Действия**.
Сообщение	*True*/*false*
Таймер	*True*/*false*
Эскалация	*True*/*false*
Ошибка	*True*
Отмена	*True*
Компенсация	Нет значений, т.к. **Действие** уже выполнено и не может быть более отменено после запуска компенсации.
Условие	*True*/*false*
Сигнал	*True*/*false*
Множественный	*True*/*false* (если все триггеры События допускают использование данного атрибута). Данная опция является наиболее ограничительной, т.е. в случае возникновения ошибки или отмены *триггеров* используется Yes.

Соединение с Действием

Промежуточное событие может быть присоединено к границе **Действия** при соблюдении следующих требований:

- Непосредственно к границе Действия МОЖЕТ БЫТЬ присоединено одно или более **Промежуточных событий.**
    - Присоединенное **Промежуточное событие** ДОЛЖНО БЫТЬ одного из следующих типов (`EventDefinition`): `Сообщение (Message), Таймер (Timer), Ошибка (Error), Эскалация (Escalation), Отмена (Cancel), Компенсация (Compensation), Условие (Conditional), Сигнал (Signal), Множественный (Multiple), Параллельный Множественный (Parallel Multiple).`
- **Промежуточное событие типа Отмена** МОЖЕТ БЫТЬ присоединено к границе **Подпроцесса** только при значении атрибута `Transaction` **Подпроцесса**, равном «*true*».

Соединение с Потоком операций

Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они МОГУТ являться *целями* и *источниками* **Потока Операций**, обратитесь к пункту 7.5.1 «Правила Соединения Потоков Операций».

- Если **Промежуточное Событие** присоединено к границе **Действия**, то:
    - **Промежуточное Событие** НЕ ДОЛЖНО являться целью **Потока Операций** или иметь каких-либо *Входящих* **Потоков Операций.**
        - **Промежуточное Событие** ДОЛЖНО являться источником **Потока Операций.**
        - **Промежуточное Событие** МОЖЕТ являться источником множества **Потоков Операций**. Для каждого такого **Потока Операций** ДОЛЖЕН БЫТЬ создан новый параллельный маршрут.
            - Исключением является **Промежуточное Событие Компенсация**, которое НЕ ДОЛЖНО ИМЕТЬ *Исходящих* **Потоков Операций**, однако, МОЖЕТ иметь *исходящую* **Ассоциацию.**
- В состав *Стандартного потока операций* МОГУТ входить **Промежуточные события** следующих типов (EventDefinition): `Неопределенное (None), Сообщение (Message), Таймер (Timer), Эскалация (Escalation), Компенсация (Compensation), Условие (Conditional), Связь (Link), Сигнал (Signal), Множественный (Multiple), Параллельный Множественный (Parallel Multiple).` В *Стандартном потоке операций* НЕ ДОЛЖНЫ участвовать **Промежуточные события** типов `Отмена (Cancel)` и `Ошибка (Error).`
    - Если **Промежуточное событие** включено в состав *Стандартного потока операций*, то:
        - **Промежуточное событие** ДОЛЖНО являться целью **Потока операций.**

Примечание: В отличие от **BPMN 2.0**, **BPMN 1.2** допускает возможность отсутствия у некоторых **Промежуточных событий** *Входящих* **Потоков операций.**

- **Промежуточное событие** МОЖЕТ иметь множество *Входящих* **Потоков операций**.

Примечание: Если **Промежуточное событие** имеет множество *Входящих* **потоков операций**, то **Поток операций** считается неконтролируемым. Это означает, что **Событие** будет запущено по прибытии *токена* по одному из маршрутов (для того, чтобы *отреагировать на триггер* или запустить его). Данное **Событие** не будет ожидать прибытия других *токенов* по оставшимся маршрутам. Если по тому же или какому-либо другому маршруту прибывает другой *токен*, запускается другой экземпляр **События**. Если возникает необходимость в контроле **Потока операций**, все маршруты должны быть соединены посредством **Шлюза**, предшествующего **Событию** (дополнительную информацию о **Шлюзах** смотрите в разделе 10.5 «Шлюзы»).

- **Промежуточное событие** ДОЛЖНО являться источником **Потока операций.**
- **Промежуточное событие** МОЖЕТ иметь множество *Исходящих* **Потоков операций**. Для каждого такого **Потока операций** ДОЛЖЕН БЫТЬ создан новый параллельный маршрут.
    - Исключением является используемое в качестве источника **Промежуточное событие Связь**. Наличие у такого **События** *Исходящих* **Потоков операций** НЕ ОБЯЗАТЕЛЬНО.
- **Промежуточное событие** НЕ ДОЛЖНО являться *источником* и *целью* **Потока операций** одновременно.

Для использования **Промежуточного события Связь** в качестве Соединителя страниц или объекта «GOTO» необходимо учитывать следующие требования:

- **Промежуточное событие Связь** МОЖЕТ являться как целью (**Событие Связь** как *цель*), так и источником (**Событие Связь** как *источник*) **Потока операций**. Однако оно НЕ ДОЛЖНО являться и *целью*, и *источником* одновременно.
    - Одно **Событие Связь**, используемое как *цель*, МОЖЕТ иметь множество **Событий Связь**, используемых как *источники*.
    - Одно **Событие Связь**, используемое как *источник*, НЕ ДОЛЖНО иметь множество **Событий Связь**, используемых как *цели*.

Соединение с Потоком сообщений

Для того, чтобы увидеть полный список графических элементов и узнать, каким образом они могут являться *целями* **Потока сообщений**, обратитесь к пункту 7.5.2 «Правила Соединения Потоков Сообщений».

Примечание: Все **Потоки сообщений** ДОЛЖНЫ соединять два разных **Пула**. Они МОГУТ быть присоединены к границам **Пула**, а также к Элементам потока внутри этого **Пула**. Они НЕ ДОЛЖНЫ использоваться для соединения двух элементов внутри одного **Пула**.

- **Промежуточное событие Сообщение** МОЖЕТ являться *целью* **Потока сообщений**. Оно может иметь один *Входящий* **Поток сообщений**.
- **Промежуточное событие Сообщение** МОЖЕТ являться *источником* **Потока сообщений**. Оно может иметь один *Исходящий* **поток сообщений.**
- **Промежуточное событие Сообщение** МОЖЕТ быть соединено как *с входящим*, так и *исходящим* **Потоком операций**, однако, не с *входящим* и *исходящим* **Потоками операций** одновременно.

10.4.5. Элементы EventDefinition

Элементы `EventDefinition` относятся к *триггерам обрабатывающих* их **Событий** (**Стартовых** и некоторых **Промежуточных событий**) и *результатам определяющих* их **Событий** (**Конечных** и некоторых **Промежуточных событий**). Типами элементов `EventDefinition` являются: `CancelEventDefinition`, `CompensationEventDefinition`, `ConditionalEventDefinition`, `ErrorEventDefinition`, `EscalationEventDefinition`, `MessageEventDefinition`, `LinkEventDefinition`, `SignalEventDefinition`, `TerminateEventDefinition` и `TimerEventDefinition` (см. таблицу 10.93). **Событие Неопределенного типа** определяется **Событием**, для которого не указано значение `EventDefinition`. **Событие Множественного типа** определяется **Событием**, для которого указано более одного значения `EventDefinition`. Для разных типов **Событий** (**Стартовых, Конечных, Промежуточных**) используются подгруппы доступных типов значений `EventDefinition`.

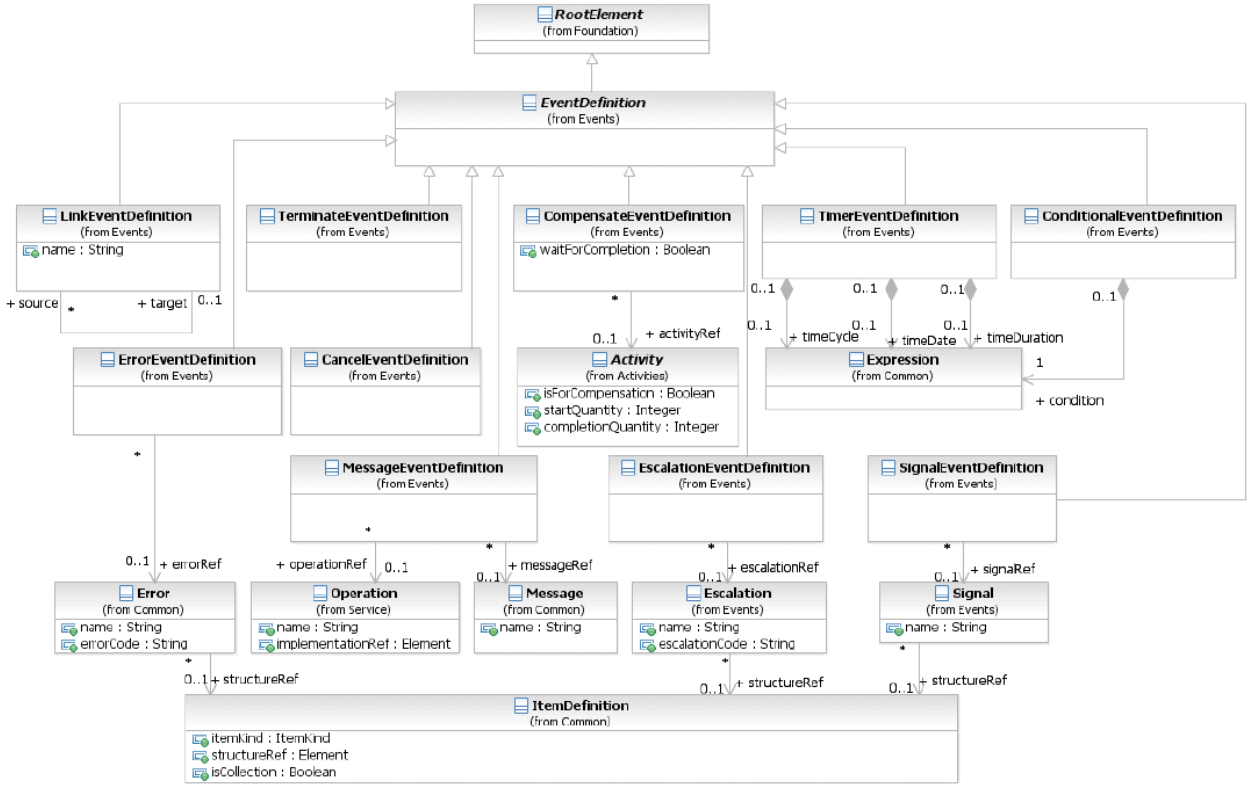
Таблица 10.93 – Типы Событий и соответствующие им маркеры

Типы Событий	**Стартовое событие**			**Промежуточное событие**				**Конечное событие**
	Процессы высокого уровня	Событийные Подпро-цессы *Прерываю-щие События*	Событийные Подпроцессы *Непреры-вающие События*	Обрабаты-вающие триггер	Присоеди-ненные к границе *Прерывающие*	*Присоединен-ные к границе Непрерываю-щие*	Воз-буждающ ие триггер	
Неопре-деленный								
Сообще-ние								
Таймер								
Ошибка								
Эскала-ция								
Отмена								
Компен-сация								
Условие								
Связь								
Сигнал								
Завер-шение								
Множест-вен.								
Парал-лельный Множест-венный								

Следующие подразделы содержат информацию об общих для всех элементов `EventDefinition` атрибутах, а также об особых атрибутах для тех элементов `EventDefinition`, которые могут иметь дополнительные атрибуты. Обратите внимание, что ни Отмена, ни Завершение не имеют дополнительных атрибутов.

Метамодель класса EventDefinition

На фигуре 10.73 изображена диаграмма классов абстрактного класса `EventDefinition`. Если определен один из подтипов `EventDefinition` (например, `TimerEventDefinition`), он хранится в `Definitions` либо в `EventDefinition`, хранимом в **Событии.**



Фигура 10.73 – Диаграмма классов элемента EventDefinition

Элемент `EventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `RootElement`. Однако он не может иметь каких-либо дополнительных атрибутов или ассоциаций.
Подклассы `ErrorEventDefinition, EscalationEventDefinition` и `SignalEventDefinition` содержат атрибуты, используемые для хранения данных. Таким образом, данные могут являться частью пакета **События**. Подкласс `MessageEventDefinition` содержит атрибут, относящийся к **Сообщению**, которое является частью пакета **Взаимоотношения**.
Следующие подразделы содержат информацию о подтипах элементов `EventDefinition`.

События типа Отмена

События типа Отмена используются только при моделировании **Подпроцесса Транзакция**. Существует две разновидности таких **Событий**: *реагирующее на триггер* **Промежуточное событие** и **Конечное событие**.

- *Реагирующее на триггер* **Промежуточное событие Отмена** ДОЛЖНО быть присоединено к границе **Подпроцесса Транзакция**. Оно НЕ МОЖЕТ входить в состав *Стандартного потока операций*.
- **Конечное событие Отмена** ДОЛЖНО использоваться только в рамках **Подпроцесса Транзакция**. Следовательно, оно НЕ МОЖЕТ использоваться в **Подпроцессах** и **Процессах** других типов.

На фигуре 10.74 отображены разновидности **Событий типа Отмена**.

Фигура 10.74 - События типа Отмена

Элемент `CancelEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`.

Compensation Event События типа Компенсация

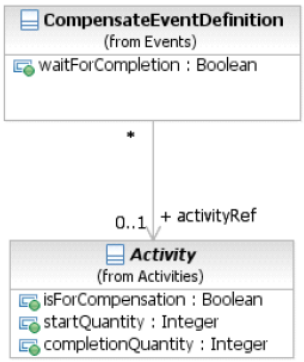
События типа Компенсация используются при моделировании запускающей и обрабатывающей *компенсации*. Существует четыре разновидности таких **Событий**: **Стартовое событие, Промежуточное событие** (как *реагирующее на триггер*, так и *определяющее результат*) и **Конечное событие**.

- **Стартовое событие Компенсация** МОЖЕТ НЕ входить в состав высокоуровневых **Процессов**.
- **Стартовое событие Компенсация** МОЖЕТ входить в состав **Подпроцессов**, работающих с данными.
- *Реагирующее на триггер* **Промежуточное событие Компенсация** ДОЛЖНО БЫТЬ присоединено только к границе **Действия**, следовательно, оно НЕ МОЖЕТ входить в состав *Стандартного потока операций*.
- *Определяющее результат* **Промежуточное событие Компенсация** МОЖЕТ являться частью *Стандартного потока операций*.
- **Конечное событие Компенсация** МОЖЕТ использоваться и в **Подпроцессах**, и в **Процессах**.

На фигуре 10.75 отображены разновидности **Событий типа Компенсация**.

Фигура 10.75 - События типа Компенсация

Фигура 10.76 отображает диаграмму класса `CompensationEventDefinition`



Фигура 10.76 - Диаграмма классов элемента CompensationEventDefinition

Элемент `CompensationEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.94 содержит информацию о дополнительных атрибутах и ассоциациях элемента `CompensationEventDefinition`.

Таблица 10.94 – Атрибуты и ассоциации элемента CompensationEventDefinition

Название атрибута	Описание/использование
activityRef: Activity [0..1]	**Стартовое событие**. Данное **Событие** реагирует на возникшую

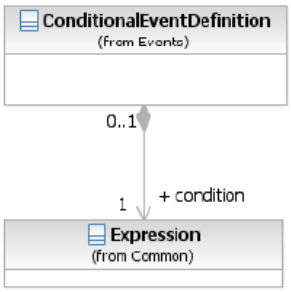
	компенсацию **Подпроцесса**. НЕ ТРЕБУЕТСЯ никакой дополнительной информации. **Подпроцесс**, работающий с данными, предоставит Id, необходимый для установки соответствия между **Событием типа Компенсация** и **Событием**, запустившем *компенсацию*. Или же произойдет переход *компенсации*. **Конечное событие.** ДОЛЖНО БЫТЬ предоставлено **Действие**, *компенсация* которого необходима. В случае, если такого **Действия** нет, *компенсация* перейдет на все выполненные **Действия** текущего **Подпроцесса** или экземпляра **Процесса** (зависит от уровня). **Промежуточное событие в составе** *Стандартного потока операций*. ДОЛЖНО БЫТЬ предоставлено **Действие**, компенсация которого необходима. В случае, если такого **Действия** нет, *компенсация* перейдет на все выполненные **Действия** текущего **Подпроцесса** или экземпляра **Процесса** (зависит от уровня). Таким образом запускается *компенсация*. **Промежуточное событие,** присоединенное к границе **Действия**. Данное **Событие** реагирует на *компенсацию*. НЕ ТРЕБУЕТСЯ никакой дополнительной информации. **Действие**, к которому присоединено это **Событие**, предоставит Id, необходимый для установки соответствия между **Событием типа Компенсация** и **Событием**, запустившем *компенсацию*. Или же произойдет переход *компенсации*.
waitForCompletion: boolean = true	**Промежуточное событие,** *определяющее результат.* Данный атрибут определяет, будет ли данное **Событие** ожидать выполнения запущенной *компенсации* (значение по умолчанию), или запустит *компенсации* и немедленно продолжится (поведение, определенное в **BPMN 1.2**).

События типа Условие

На фигуре 10.77 отображены разновидности **Событий типа Условие**.

Фигура 10.77 - События типа Условие

Фигура 10.78 отображает диаграмму класса `ConditionalEventDefinition`



Фигура 10.78 - Диаграмма класса ConditionalEventDefinition

Элемент `ConditionalEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.95 содержит информацию о дополнительных ассоциациях элемента `ConditionalEventDefinition`.

Таблица 10.95 – Атрибуты и ассоциации элемента CompensationEventDefinition

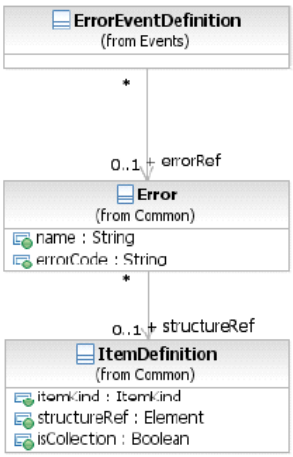
Название атрибута	Описание/использование
condition: Expression	Данное `выражение` может быть не полностью определено и создано с использованием ествественного языка. Если тип **Процесса** - выполняемый (значение его атрибута `isExecutable` равно «*true*»), а тип *триггера* – `Условный`, то ДОЛЖНО БЫТЬ указано значение `FormalExpression`.

События типа Ошибка

На фигуре 10.79 отображены разновидности **Событий типа Ошибка**.

Фигура 10.79 - События типа Ошибка

Фигура 10.80 отображает диаграмму класса `ErrorEventDefinition`



Фигура 10.80 - Диаграмма классов элемента ErrorEventDefinition

Элемент `ErrorEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.96 содержит информацию о дополнительных ассоциациях элемента `ErrorEventDefinition`.

Таблица 10.96 – Атрибуты и ассоциации элемента ErrorEventDefinition

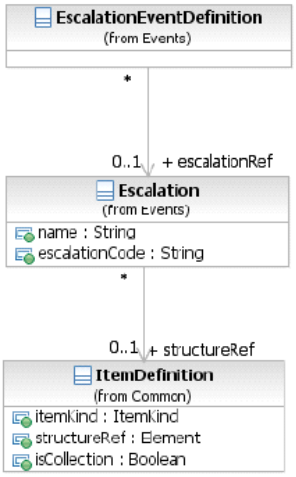
Название атрибута	Описание/использование
error: Error [0..1]	Если *триггер* имеет тип `Ошибка`, то для `FormalExpression` МОЖЕТ БЫТЬ указанно значение `Error`.

Событие Эскалация

На фигуре 10.81 отображены разновидности **Событий типа Эскалация**.

Фигура 10.81 - События типа Эскалация

Фигура 10.82 отображает диаграмму класса `EscalationEventDefinition`



Фигура 10.82 - Диаграмма класса EscalationEventDefinition

Элемент `EscalationEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.97 содержит информацию о дополнительных атрибутах и ассоциациях элемента `EscalationEventDefinition`.

Таблица 10.97 – Атрибуты и ассоциации элемента EscalationEventDefinition

Название атрибута	Описание/использование
escalationRef: Escalation [0..1]	Если *триггер* имеет тип `Эскалация`, то МОЖЕТ БЫТЬ указанно значение `Escalation`.

События типа Связь

Посредством Связи соединяются два участка **Процесса**. **События типа Связь** могут применяться для отображения цикличности либо для того, чтобы избежать использования слишком длинных цепочек **Потоков операций**. Использование **Событий Связь** недопустимо для отображения любых соединений на одном уровне **Процесса** (например, с их помощью нельзя соединять *родительский* **Процесс** с **Подпроцессом**).

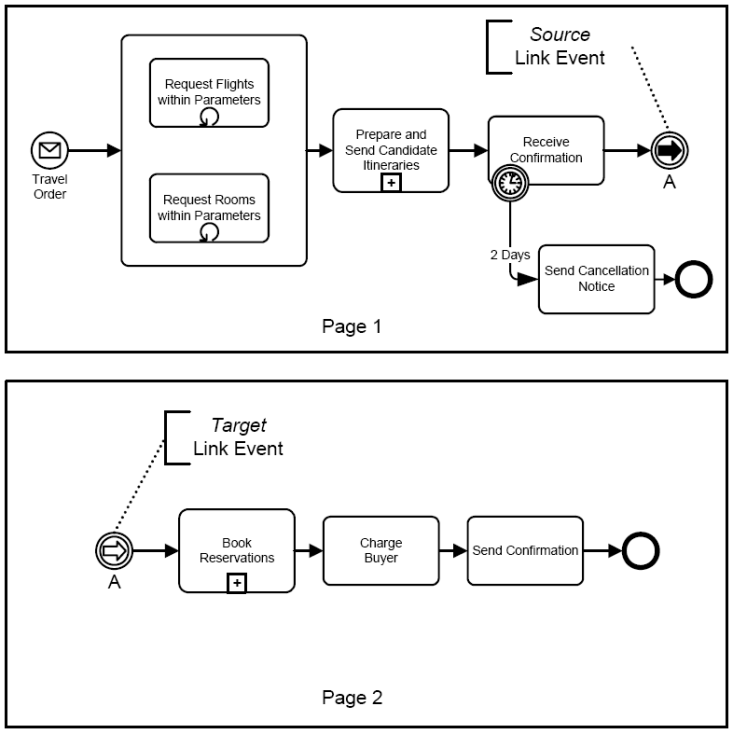
На фигуре 10.83 отображены разновидности **Событий типа Связь.**

Фигура 10.83 - События типа Связь

Парные **Промежуточные события** этого типа могут быть использовать в качестве «соединителя страниц». Такое соединение удобно в тех случаях, когда необходимо напечатать **Процесс**, размещенный на нескольких страницах. Они также могут использоваться для перехода к определенным точкам (в качестве общих объектов «GOTO») в рамках какого-либо одного уровня **Процесса**. В качестве источников может быть использовано любое количество таких **Событий**, однако, в качестве цели может использоваться лишь одно **Промежуточное событие типа Связь**. В случае, если такое **Промежуточное событие** является вторым из пары, маркер данного **События** отображается без заливки (см. фигуру 10.84, нижняя часть фигуры, слева). Если же такое **Промежуточное событие** является первым (т.е. используется для указания на второе), маркер **События** имеет заливку (см. фигуру 10.84, верхняя часть фигуры, справа).

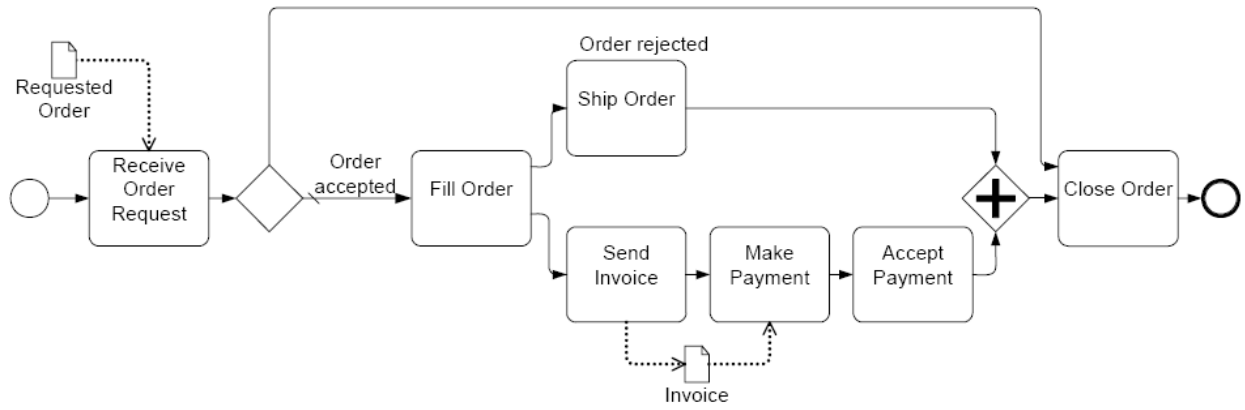
Поскольку протяженность модели **Процесса** обычно превышает длину одной страницы, очень часто возникает проблема соединения **Потока Операций** на участке таких разрывов. Одним из решений этой проблемы является

добавление в модель «соединителей страниц». Таким образом, указывается места, где на предыдущей странице заканчивается ход **Потока Операций**, и где он возобновляется на следующей странице. В качестве таких соединителей страниц **BPMN** предлагает использовать парные **Промежуточные События Типа Связь** (см. фигуру 10.84. Обратите внимание, что фигура состоит из двух частей, каждая из которых представляет собой страницу, а не отдельно взятый **Пул**). Первое из **Событий** отображается в конце одной страницы. Оно имеет название и соединено с одним *Входящим* **Потоком Операций** (соответственно, не имеет *Исходящего* **Потока Операций**). Второе **Событие** отображается в начале следующей страницы. Оно имеет то же название, что и первое, и соединено с одним *Исходящим* **Потоком Операций** (соответственно, не имеет *Входящего* **Потока Операций**).

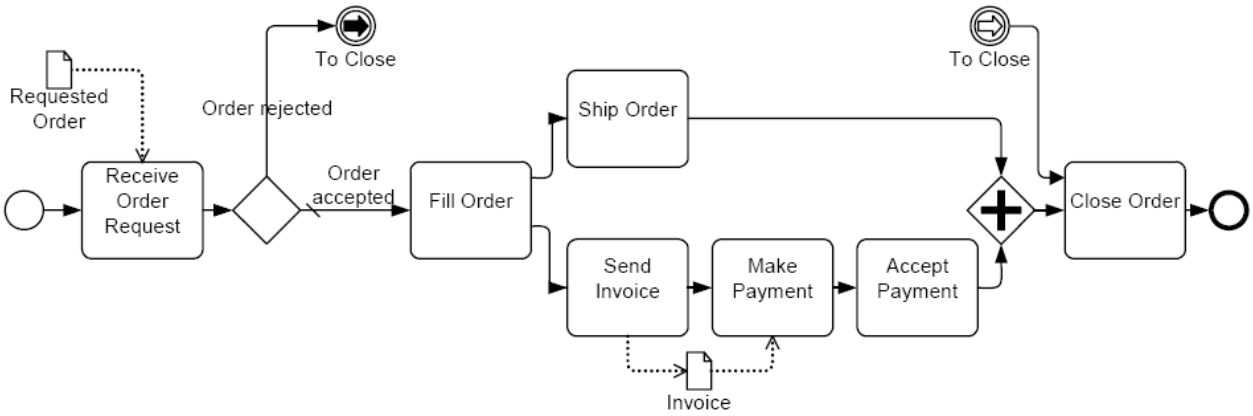


Фигура 10.84 – Промежуточные события типа Связь, выступающие в качестве соединителей страниц

Другое предназначение **Промежуточных событий типа Связь** – использование их в качестве объектов «GOTO». В этом случае они выступают в той же роли, что и соединители страниц (см. описание выше), но с одним исключением - несколько **Промежуточных событий Связь**, используемых для этой цели, могут появляться в любом месте диаграммы, будь то одна и та же страница или несколько страниц. Используемые в качестве объектов «GOTO» **События Связь** предназначены для уменьшения длины цепочек **Потоков операций** с большой протяженностью, которые, по мнению некоторых разработчиков моделей **Процессов**, трудно отслеживать. Объекты «GOTO» используются для предупреждения отображения слишком протяженных **Потоков операций** (см. фигуры 10.85 и 10.86). Обе диаграммы представляют одинаковое поведение. Как показано на фигуре 10.86, если маршрут «Отклонение заказа» отходит от **Шлюза**, то *токен*, пересекающий **Поток операций**, достигает **Промежуточное события Связь**, *используемое в качестве источника*, и перепрыгивает к *используемому в качестве цели* **Промежуточному событию Связь**. Затем он продолжает ход с нисходящим **Потоком операций**. В данном случае, **Процесс** ведет себя так, как если бы непосредственно **Поток операций** соединял два элемента.

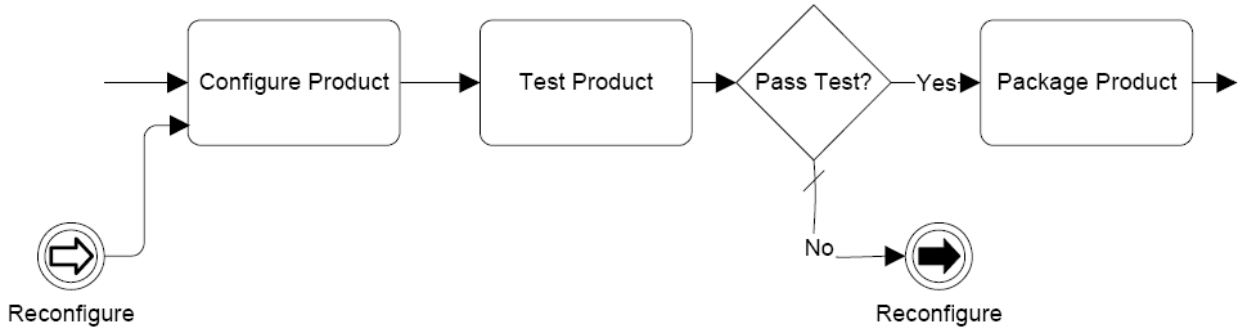


Фигура 10.85 – Процесс, содержащий протяженный Поток операций



Фигура 10.86 – Процесс, использующий в качестве объектов «GOTO» Промежуточные события Связь

В некоторых методологиях предпочтение отдается **Потокам операций** с хронологическим направлением. Такие **Потоки операций** не могут иметь непосредственную связь с объектами, расположенными в направлении, обратном направлению **Потока операций**. Конечно, определенная логика в таком отображении **Потоков операций** есть, однако, как быть в ситуациях, когда необходимо использование циклов? **Промежуточные события Связь** как раз-таки могут использоваться для отображения соединения **Потока операций** с объектами, расположенными в обратном ему направлении, а также для отображения *циклов*, причем, без нарушения ограничений использования **Потока операций** (см. фигуру 10.87).



Фигура 10.87 – Процесс, где для отображения цикла используется Промежуточное события Связь

Элемент `LinkEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.98 содержит информацию о дополнительных атрибутах элемента `LinkEventDefinition`.

Таблица 10.98 – Атрибуты элемента LinkEventDefinition

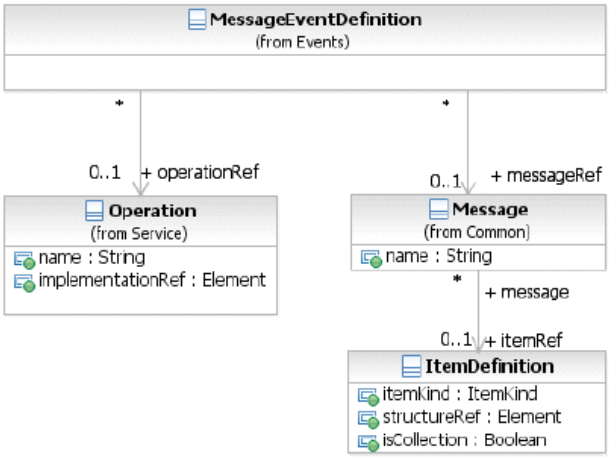
Название атрибута	Описание/использование
name: string	Если используется *триггер* типа *Связь*, ДОЛЖНО БЫТЬ указано имя.
sources: LinkEventDef-inition [1..*]	Данный атрибут используется для указания ссылки на соответствующий элемент LinkEventDefinition и указывает на предназначение Промежуточного события Связь (источник, цель).
target: LinkEventDefini-tion [1]	Данный атрибут используется для указания ссылки на соответствующий элемент LinkEventDefinition и указывает на предназначение Промежуточного события Связь (источник, цель).

Событие Сообщение

На фигуре 10.88 отображены разновидности **Событий типа Сообщение.**

Фигура 10.88 - События типа Сообщение

Фигура 10.89 отображает диаграмму класса `MessageEventDefinition`



Фигура 10.89 - Диаграмма класса MessageEventDefinition

Элемент `MessageEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.99 содержит информацию о дополнительных ассоциациях элемента `MessageEventDefinition`.

Таблица 10.99 Ассоциации элемента MessageEventDefinition

Название атрибута	Описание/использование
messageRef: Message [0..1]	Если значение атрибута `isExecutable` **Процесса** равно «*true*», ДОЛЖНО БЫТЬ указано **Сообщение**.
operationRef: Operation [0..1]	Данный атрибут указывает на **Операцию**, используемую для **События типа Сообщение**. Значение этого атрибута ДОЛЖНО указываться для выполняемых **Процессов**.

События Множественного типа

Стартовое событие Множественного типа.

Если *триггер* имеет тип Множественный, то способов запуска **Процесса** будет несколько, однако, лишь один из *триггеров* необходим для его запуска. Посредством подклассов `EventDefinition` указывается то, какие *триггеры* будут задействованы.

Конечное событие Множественного типа.

Если *триггер* имеет тип Множественный, то путей завершения **Процесса** будет несколько, и все они будут задействованы. Посредством подклассов `EventDefinition` указывается то, с какими *результатами* завершится **Процесс**.

Промежуточное событие Множественного типа в составе *Стандартного потока операций***.**

Если *триггер* имеет тип Множественный, то для его *обработки* НЕОБХОДИМО использование лишь одного значения `EventDefinition`. Если **Событие** используется с целью инициирования *триггера*, принимаются во внимание все значения `EventDefinitions`, которые и *определяют*, каким будет *результат*.

Промежуточное событие Множественного типа, присоединенное к границе **Действия.**

Если *триггер* имеет тип Множественный, то для его *обработки* НЕОБХОДИМО использование лишь одного значения `EventDefinition`.

На фигуре 10.90 отображены разновидности **Событий Множественного типа**.

Фигура 10.90 – События Множественного типа

События Неопределенного типа

Для **Событий Неопределенного типа** не указано значение `EventDefinition`. Существует три разновидности **Событий** этого типа: **Стартовое событие, Промежуточное событие**, используемые для *обработки триггера*, и **Конечное событие** (см. фигуру 10.91).

- **Стартовые события Неопределенного типа** МОГУТ входить в состав **Процесса** высокого уровня или любого **Подпроцесса** (кроме **Подпроцесса**, работающего с данными).

- **Стартовые события Неопределенного типа** МОГУТ НЕ входить в состав **Подпроцесса**, работающего с данными.
- **Промежуточные события Неопределенного типа**, *обрабатывающие триггер*, ДОЛЖНЫ включаться только в *Стандартный поток операций* и, соответственно, НЕ МОГУТ быть присоединены к границе **Действия**.
- **Конечные события Неопределенного типа** МОГУТ входить в состав любого **Подпроцесса** или **Процесса**.

На фигуре 10.91 отображены разновидности **Событий Неопределенного типа**.

Фигура 10.91 – События Неопределенного типа

События Параллельного Множественного типа

Стартовое событие Параллельного Множественного типа.

Если *триггер* имеет `тип Множественный`, то для запуска **Процесса** НЕОБХОДИМО наличие нескольких способов, и все они будут задействованы. Посредством подклассов `EventDefinition` указывается то, какие *триггеры* будут использоваться. Обратите внимание, что значение атрибута `parallelMultiple` **События** данного типа ДОЛЖНО БЫТЬ равно «*true*».

Промежуточное событие Параллельного Множественного типа в составе **Стандартного потока операций.**

Если *триггер* имеет тип `Множественный`, то для его инициирования НЕОБХОДИМЫ все указанные значения `EventDefinition`. Обратите внимание, что значение атрибута `parallelMultiple` **События** данного типа ДОЛЖНО БЫТЬ равно «*true*».

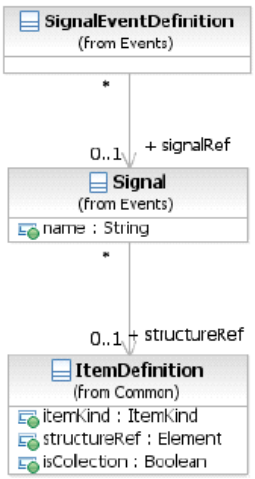
Промежуточное событие Параллельного Множественного типа, присоединенное к границе **Действия.**

Если *триггер* имеет `тип Множественный`, то для его инициирования НЕОБХОДИМЫ все указанные значения `EventDefinition`. Обратите внимание, что значение атрибута `parallelMultiple` **События** данного типа ДОЛЖНО БЫТЬ равно «*true*».

На фигуре 10.92 отображены разновидности **Событий Параллельного Множественного типа**.

Фигура 10.92 – События Параллельного Множественного типа

События типа Сигнал



Фигура 10.93 - Диаграмма классов элемента SignalEventDefinition

На фигуре 10.94 отображены разновидности **Событий типа Сигнал**.

Фигура 10.94 – События типа Сигнал

Элемент `SignalEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.100 содержит информацию о дополнительных ассоциациях элемента `SignalEventDefinition`.

Таблица 10.100 Ассоциации элемента SignalEventDefinition

Название атрибута	Описание/использование
signalRef: Signal [0..1]	Если *триггер* имеет тип **Сигнал**, то будет подан сигнал.

Событие типа Завершение

На фигуре 10.95 отображено **Событие типа Завершение.**

Фигура 10.95 – Событие типа Завершение

Элемент TerminateEventDefinition наследует атрибуты и ассоциации элемента BaseElement (см. таблицу 8.5) благодаря его связи с элементом EventDefinition.

События типа Таймер

На фигуре 10.96 отображены разновидности **Событий типа Таймер**.

Фигура 10.96 – События типа Таймер

Элемент `TimerEventDefinition` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5) благодаря его связи с элементом `EventDefinition`. Таблица 10.101 содержит информацию о дополнительных ассоциациях элемента `TimerEventDefinition`.

Таблица 10.101 Ассоциации элемента TimerEventDefinition

Название атрибута	Описание/использование
timeDate: Expression [0..1]	Если *триггер* имеет тип Таймер, то МОЖЕТ БЫТЬ указано значение `timeDate`. Атрибуты Событий типа Таймер взаимно исключаемы, а в случае, если указано значение какого-то из этих атрибутов, то значение атрибута `timeDate` указываться НЕ ДОЛЖНО (если значение атрибута `isExecutable` **Процесса** равно «*true*»). Возвращаемый тип атрибута `timeDate` ДОЛЖЕН соответствовать формату отображения даты и времени ISO-8601.
timeCycle: Expression [0..1]	Если *триггер* имеет тип Таймер, то МОЖЕТ БЫТЬ указано значение `timeCycle`. Атрибуты Событий типа Таймер взаимно исключаемы, а в случае, если указано значение какого-то из этих атрибутов, то значение атрибута `timeCycle` указываться НЕ ДОЛЖНО (если значение атрибута `isExecutable` **Процесса** равно «*true*»). Возвращаемый тип атрибута `timeCycle` ДОЛЖЕН соответствовать формату отображения повторяющихся интервалов времени ISO-8601.
timeDuration: Expression [0..1]	Если *триггер* имеет тип Таймер, то МОЖЕТ БЫТЬ указано значение `timeDuration`. Атрибуты Событий типа Таймер взаимно исключаемы, а в случае, если указано значение какого-то из этих атрибутов, то значение атрибута `timeDuration` указываться НЕ ДОЛЖНО (если значение атрибута `isExecutable` **Процесса** равно «*true*»). Возвращаемый тип атрибута `timeDuration` ДОЛЖЕН соответствовать формату отображения интервалов времени ISO-8601.

10.4.6. Обработка Событий

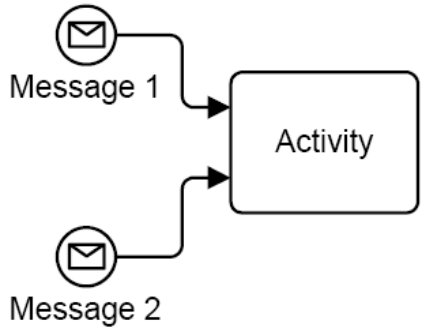
BPMN позволяет использовать продвинутые способы управления **Событиями**, возникающими в ходе выполнения **Процесса** (например, обработка имеющегося **События**). Кроме того, **BPMN** поддерживает добавление **События** в **Процесс** (к примеру, инициирование запуска **События**). Обработка **Событий** включает в себя *обработку* и *инициирование* **Событий**, а также достижение результата в ходе выполнения **Процесса**. Существует три типа *обработчиков Событий*: обработчики, запускающие Событие, *обработчики*, входящие в состав Стандартного **Потока Операций**, и *обработчики*, присоединенные к **Действию** либо через

границу **События**, либо посредством других *встроенных обработчиков* (в **Подпроцессе**, работающем с данными).

Обработка Стартовых событий

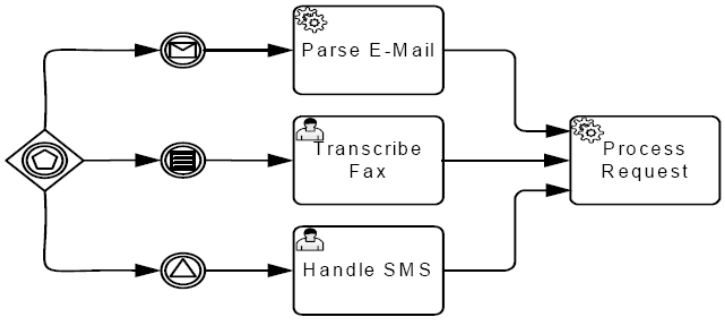
BPMN предлагаются различные способы запуска **Процесса**. Обработка одиночного **Стартового события** подразумевает запуск нового экземпляра **Процесса** всякий раз, когда оно возникает. **Потоки операций**, *отходящие* от такого **События**, имею стандартное направление. Под обработкой множественного **Стартового события** подразумевается использование необходимых сценариев, выбранных из предлагаемых **BPMN.**

Эксклюзивный запуск Процесса. Большинство общепринятых сценариев запуска **Процесса** предполагают запуск его экземпляра с помощью одного из возможных **Стартовых событий**. Всякий раз, когда такое **Событие** появляется на диаграмме **Процесса**, формируется новый его экземпляр. Из приведенных ниже примеров видно, как два **События** соединены с одним **Действием** (см. фигуру 10.97). В ходе выполнения **Процесс** появление любого из этих **Событий** определяет последующее создание нового экземпляра данного экземпляра **Процесса**, а также запускает **Действие**. Обратите внимание, что одиночное **Стартовое событие Множественного типа**, содержащее значения `MessageEventDefinition`, будет вести себя также.



Фигура 10.97 – Эксклюзивный запуск Процесса

Фигура 10.98 отображает Процесс, запускаемый с помощью основанного на событиях Шлюза.



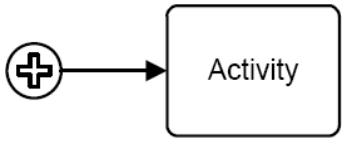
Фигура 10.98 – Процесс, запускаемый с помощью основанного на событиях Шлюза

В данном случае **Стартовое Событие**, отображаемое первым на фигуре, запускает новый экземпляр **Процесса**, а затем происходит ожидание появления других **Событий**, берущих начало от того же самого **Шлюза**. Это соответствует семантике исполнения, характерной для Основанного на данных **Эксклюзивного Шлюза**. Обратите внимание, что это пример лишь одного сценария, где **Шлюз** может отображаться без *Входящего* **Потока Операций**.

Для запуска **Процесса BPMN** позволяет использовать множественные группы Основанных на данных **Шлюзов**. Они участвуют в одном Диалоге и, следовательно, имеют одну и ту же корреляционную информацию.

В данном случае должно быть достигнуто одно **Событие** каждой группы. Первое **Событие** формирует новый экземпляр **Процесса**, а последующие сортируются для данного экземпляра, определенного благодаря содержащейся в нем корреляционной информации.

Синхронизация Событий. Если разработчику модели требуется объединить в одном **Процессе** несколько отдельных **Стартовых событий**, он ДОЛЖЕН отобразить это в соответствии со схемой, приведенной в фигуре 10.99



Фигура 10.99 – Синхронизация Событий при запуске Процесса

Стартовое событие Параллельного Множественного типа МОЖЕТ объединять несколько отдельных **Стартовых событий**, каждое из которых ДОЛЖНО хотя бы раз происходить для того, чтобы сформировался новый экземпляр **Процесса. Потоки операций**, отходящие от такого **Событи**я, имеют стандартное направление. Для получения более подробной информации об *обработке* **Стартовых событий** см. раздел 13.4.1.

Обработка Событий Стандартного потока операций (Промежуточных событий)

Обработка **Промежуточного события** заключается в ожидании запуска данного **События** по достижению его **Потоком операций. Промежуточное событие** завершается сразу после запуска. **Потоки операций**, отходящие от такого **События**, имеют стандартное направление.

Обработка Событий, присоединенных к границам Действия (Промежуточных событий, соединенных с границей Действия, и Событийных Подпроцессов)

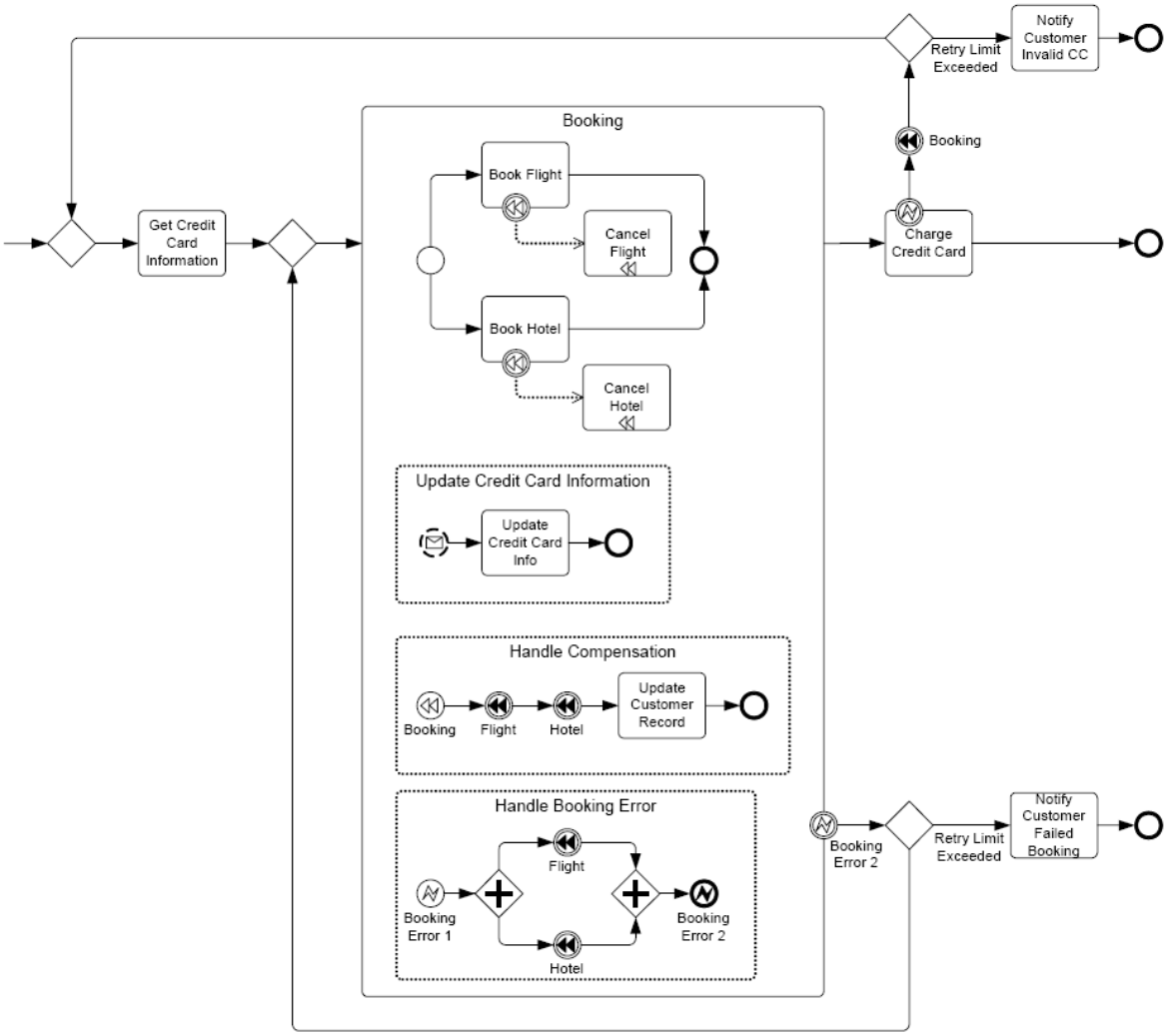
Обработка **Промежуточного события**, соединенного с границами **Действия**, заключается в завершении запущенного **События** и либо отмене выполнения **Действия**, к которому данное **Событие** присоединено (как результат от **Действия** направляется *Стандартный поток операций*), либо запуске *обработчика Событий* без отмены выполнения **Действия** (только для **Событий типа Сообщение, Сигнал, таймер, Условие**; использование **Ошибки** запрещено).

Промежуточное Событие, прерывающее **Действие,** к которому присоединено, определяется посредством установки значения «*true*» для атрибута `cancelActivity`. В какой бы момент ни происходило это **Событие**, выполнение соответствующего ему **Действия** прерывается. При этом формируется *токен*, следующий по направлению **Потока операций** и запускающий следующий элемент **Процесса** (связанный с **Событием** с помощью безусловного **Потока операций**, называемого *Потоком исключений*).

Промежуточное Событие, не прерывающее **Действие**, к которому присоединено, определяется посредством установки значения «*false*» для атрибута `cancelActivity`. В какой бы момент ни происходило это **Событие**, выполнение соответствующего ему **Действия** продолжается. Поскольку параллельно с продолжение выполнения **Действия** для направленного от **События Потока операций** формируется *токен*, ДОЛЖНО учитываться то, что данный маршрут сливается с основным **Потоком операций Процесса**. В данном случае *токен*, как правило, завершается собственным **Конечным Событием**.

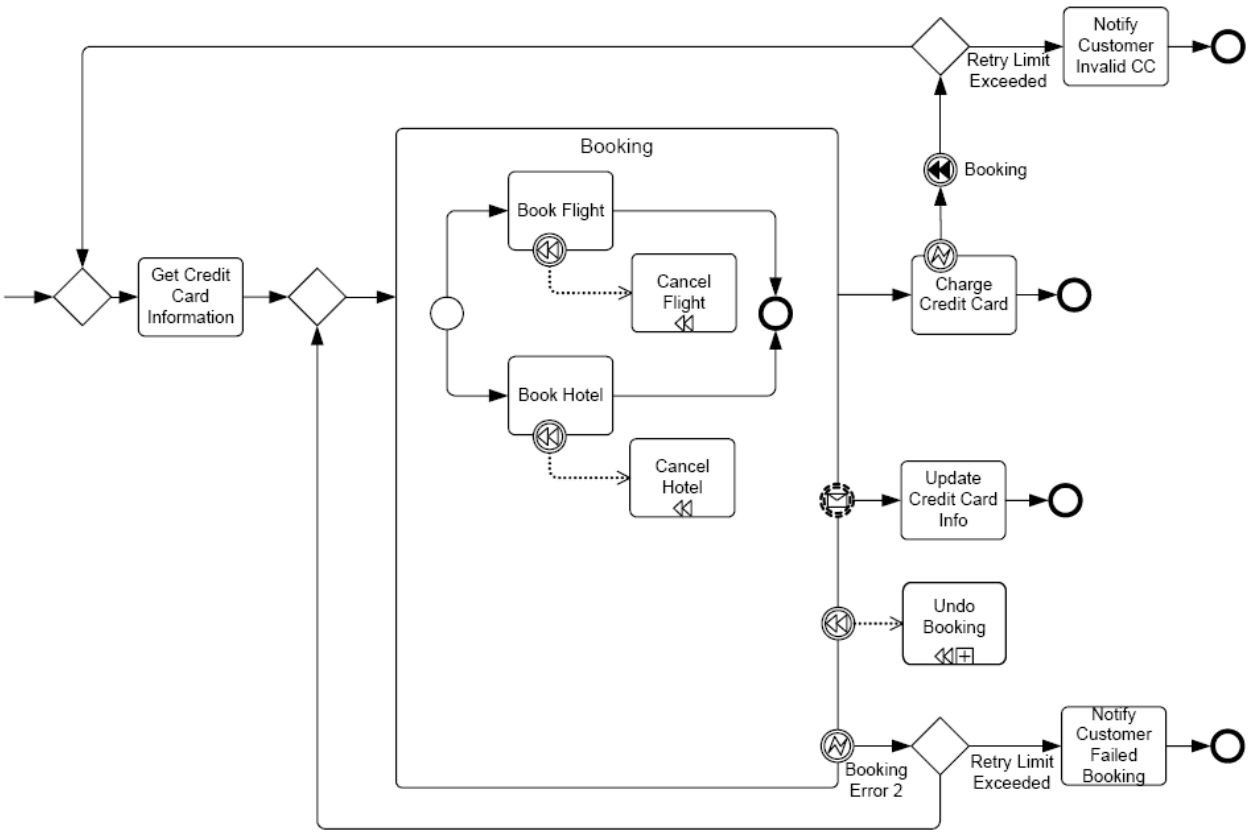
На фигуре 10.100 отображен фрагмент **Процесса** заказа тура, содержащий **Подпроцесс**. Данный **Процесс** состоит из основной части и трех **Подпроцессов**, работающих с **Событиями**, которые направлены на обработку **Событий** в рамках одного контекста: **Событийный Подпроцесс типа Ошибка** – для отмены работы **Подпроцесса**; **Событийный Подпроцесс типа Сообщение** – для обновления статуса

Подпроцесса при ожидании разрешения продолжать его выполнение; **Событийный Подпроцесс типа Компенсация.**



Фигура 10.100 – Пример Процесса, содержащего работающие Событийные Подпроцессы для встроенной обработки Событий

На фигуре 10.101 отображен фрагмент вышеописанного **Процесса** бронирования тура, содержащий не **Подпроцессы**, а обработчики присоединенных к **Действиям Событий**. Обратите внимание, что в данном случае обработчикам **Событий** не доступен контекст **Подпроцесса** «Booking»; они работают за его пределами. Это значит, что *компенсация* здесь представлена в виде черного ящика.



Фигура 10.101 – Пример Процесса, содержащего обработчик присоединенных к Действиям Событий

Необходимо учитывать разницу между *прерывающими* и *непрерывающими* **Событиями**. Обработка этих **Событий** описана в разделах выше. Если **Событие** является прерывающим (**Ошибка, Эскалация, Сообщение, Сигнал, Таймер, Условие, Множественное, Параллельное Множественное**), то для **Событий** одного типа ДОЛЖЕН использоваться только один **Подпроцесс**, работающий с **Событиями**. Это исключает последующее использование для них каких-либо других непрерывающих обработчиков.

Причина такого ограничения заключается в особенностях прерывающего **Событийного Подпроцесса** и **Событий**, присоединенных к **Действиям**, поскольку они нарушают стандартное выполнение родительского Действия, а после их выполнения родительское **Действие** моментально прерывается. Одновременное использование вышеперечисленных обработчиков здесь запрещено, и лишь один из них может применяться за один раз. Это, однако, не накладывает ограничения на определение нескольких прерывающих обработчиков, если при этом каждый из указанных обработчиков относится к **Событию** типа, отличного от других.

Если **Событие** является непрерывающим (**Эскалация, Сообщение, Сигнал, Таймер, Условие, Множественное, Параллельное Множественное)**, то для **Событий** одного типа может быть использовано любое количество работающих с данными **Подпроцессов**; при этом все эти **Подпроцессы** могут выполнять одновременно. В ходе выполнения порядок их вызова не определен. Вышеописанные ограничения работают и для **Событий**, присоединенных к **Действиям**. В ходе выполнения непрерывающего **Подпроцесса**, работающего с данными, выполнение родительского **Действия** происходит в обычном режиме.

В случае, если для обработки **Событий Подпроцесс** использует и **Событийный Подпроцесс**, и обработчик прикрепленного к границам **Действия Событие**, и при этом они имеют одно и то же значение `EventDefinition`, должно учитываться следующее:

- Если работающий с данными **Подпроцесс** повторно инициирует начало **Событие** после его завершения, то запускается присоединенное к **Действию Событие**.

- Если работающий с данными **Подпроцесс** завершается без повторного инициирования начала **Событие**, то **Действие** должно быть выполнено, а ход Стандартного **Потока Операций** - возобновлен. В других случаях работающий с данными **Подпроцесс** «поглощает» **Действие.**

Обработчики прерывающих Событий (Ошибка, Эскалация, Сообщение, Сигнал, Таймер, Условие, Параллельный Множественный)

Значение атрибута `cancelActivity` *обработчиков* прерывающих **Событий** равно «*true*». Неважно, на каком этапе **Процесса** возникает **Событие**, обрабатывается ли оно на границе с **Действием** или в общем потоке, выполнение относящегося к нему **Действия** прерывается. Если указан обработчик **Ошибки** общего потока (в **Подпроцессе**), он работает в рамках данного **Подпроцесса**. Если в наличие есть присоединенное в границе **Действия Событие Ошибка**, то **Потоки операций** направляются от этого **События**. Выполнение родительского **Действия** отменяется либо после завершения обработки **Ошибки,** либо при отхождении от границ **События Потока операций.**

В состав описанного выше **Подпроцесса** «Booking» входит *Обработчик События Ошибка*, который и определяет, что должно произойти в случае возникновения в данном **Подпроцессе** *ошибки*, т.е. в случае, если уже сделанный заказ отменяется с *компенсацией*. В данном случае *Обработчик События Ошибка* продолжает работать за пределами **Подпроцесса** через присоединенное к его границе **Событие Ошибка**.

Обработчики непрерывающих Событий (Эскалация, Сообщение, Сигнал, Таймер, Условие, Множественный Параллельный Множественный)

Значение атрибута `cancelActivity` *обработчиков* прерывающих *Событий* равно «*false*».

Если используется **Событийный Подпроцесс**, то на каком бы этапе **Процесса** не возникло **Событие**, оно завершается, а соответствующий **Подпроцесс** выполняется. Если задействовано несколько **Событий**, выполняемых параллельно, они обрабатываются одновременно. Это значит, что одновременно формируются несколько экземпляров **Событийного Подпроцесса.** *Непрерывающее* **Стартовое событие** указывает на то, что экземпляры **Событийного Подпроцесса** выполняются одновременно для соответствия требованиям выполнения такого **Процесса.**

Если используется присоединенное к **Действию Событие**, то на каком бы этапе **Процесс**а не возникло **Событие**, обработчик работает одновременно с выполнением **Действия**. Если для такого **События** также указан и **Событийный Подпроцесс** (в **Подпроцессе**), обработчик **События** работает в контексте указанного **Подпроцесса**. После этого от границы **События** отходят **Потоки операций**. Поскольку параллельно с выполнением **Действия** для *Исходящего* **потока операций** формируется *токен*, ДОЛЖНО учитываться то, что при слиянии данного **Потока операций** с основным **Потоком операций Процесса**, *токен* завершается своим **Конечным событием.**

В приведенном выше примере **Процесса** *Обработчик Событий* позволяет обновлять данные кредитной карты В ходе выполнения **Подпроцесса** «Booking». Он запускается **Событием**, содержащим данные кредитной карты. Такое **Сообщение** может быть получено вне зависимости от того, когда поток управления находится в основной части **Подпроцесса**. При получении такое **Сообщение, Действия**, содержащиеся в соответствующем *обработчике Событий*, выполняются одновременно с **Действиями**, включенными в основной состав **Подпроцесса**.

См. раздел «Промежуточные события» для получения информации о семантике присоединенных к **Действиям Промежуточных событий**, а также раздел «Подпроцессы, работающие с данными» для получения информации об операционной семантике непрерывающих **Подпроцессов**, работающих с данными.

Обработка Конечных событий

При использовании **Конечного события Завершение** выполнение всех оставшихся в **Процессе Действий** завершается.

Использование **Конечного события Отмена** разрешено только в **Подпроцессе Транзакция**, поскольку оно отменяет **Подпроцесс** и прекращает выполнение соответствующей *Транзакции* **Подпроцесса**.

При использовании других типов **Конечных событий** их поведение соответствует тому, что определено посредством `EventDefinition`. В случае отсутствия последующих активных **Действий** экземпляр **Подпроцесса/Процесса** считается выполненным.

10.4.7. Рамки

Под *рамками* подразумевается контекст, в котором осуществляется выполнение **Действия**. Сюда входят:

- Доступные **Объекты Данных** (включая **Входные** и **Выходные Данные**).
- Доступные **События** для *обработки* и *инициирования триггеров*.
- **Диалоги**, имеющие быть в данном контексте.

Вообще, *рамками* обособлена одна основная цепочка **Действий**, берущая начало в крайнем пункте (на границе). И наоборот, все **Действия** заключены в *рамки*, которые могут быть вложены в иерархическом порядке.
В ходе выполнения *рамки* могут иметь несколько своих экземпляров. Они также вкладываются в иерархическом порядке в соответствии с тем, когда были сформированы. В каждом экземпляре рамок могут действовать несколько *токенов*.
Жизненный цикл экземпляра *рамок*, в свою очередь, может быть:

- Активирован
- Выполняется
- Выполнен
- Компенсируется
- Компенсирован
- С ошибкой
- Отменяется
- Отменен

В **BPMN** описаны графические элементы диаграммы, которые могут быть заключены в *рамки*, а именно:

- **Хореография (Choreography)**
- **Пул**
- **Подпроцесс**
- **Задача**
- **Действие**
- *Многоуровневые элементы*

Рамками определяется семантика:

- Видимости **Объектов Данных** (включая **Входные** и **Выходные Данные**).
- Выполнения **Событий**.
- Начала/завершения выполнения *токена*.

Заключенные в границы **Объекты данных**, **События** и `родственные им элементы` (`correlation keys`) могут отображаться на диаграмме или быть скрыты.

10.4.8. Представление XML-схемы для пакета События

Таблица 10.102 – XML–схема элемента для BoundaryEvent

```
<xsd:element name="boundaryEvent" type="tBoundaryEvent" substitutionGroup="flowElement"/>
<xsd:complexType name="tBoundaryEvent">
    <xsd:complexContent>
        <xsd:extension base="tCatchEvent">
            <xsd:attribute name="cancelActivity" type="xsd:boolean" default="true"/>
            <xsd:attribute name="attachedToRef" type="xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.103 – XML–схема элемента для CancelEventDefinition

```
<xsd:element name="cancelEventDefinition" type="tCancelEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tCancelEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.104 – XML–схема для элемента CatchEvent

```
<xsd:element name="catchEvent" type="tCatchEvent"/>
<xsd:complexType name="tCatchEvent" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tEvent">
            <xsd:sequence>
                <xsd:element ref="dataOutput" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="dataOutputAssociation" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="outputSet" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="eventDefinition" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="eventDefinitionRef" type="xsd:QName" minOccurs="0"
                maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="parallelMultiple" type="xsd:boolean" default="false"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.105 – XML–схема для элемента CancelEventDefinition

```
<xsd:element name="cancelEventDefinition" type="tCancelEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tCancelEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.106 – XML–схема для элемента CompensateEventDefinition

```
<xsd:element name="compensateEventDefinition" type="tCompensateEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tCompensateEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition">
            <xsd:attribute name="waitForCompletion" type="xsd:boolean"/>
            <xsd:attribute name="activityRef" type="xsd:QName"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.107 – XML–схема для элемента ConditionalEventDefinition

```
<xsd:element name="conditionalEventDefinition" type="tConditionalEventDefinition" substitutionGroup="eventDef-
```

```
            inition"/>
<xsd:complexType name="tConditionalEventDefinition">
      <xsd:complexContent>
            <xsd:extension base="tEventDefinition">
                  <xsd:sequence>
                        <xsd:element name="condition" type="tExpression"/>
                  </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.108 – XML–схема для элемента ErrorEventDefinition

```
<xsd:element name="errorEventDefinition" type="tErrorEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tErrorEventDefinition">
      <xsd:complexContent>
            <xsd:extension base="tEventDefinition">
                  <xsd:attribute name="errorRef" type="xsd:QName"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.109 – XML–схема для элемента EscalationEventDefinition

```
<xsd:element name="escalationEventDefinition" type="tEscalationEventDefinition"
            substitutionGroup="eventDefinition"/>
<xsd:complexType name="tEscalationEventDefinition">
      <xsd:complexContent>
            <xsd:extension base="tEventDefinition">
                  <xsd:attribute name="escalationRef" type="xsd:QName"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.110 – XML–схема для элемента Event

```
<xsd:element name="event" type="tEvent" substitutionGroup="flowElement"/>
<xsd:complexType name="tEvent" abstract="true">
      <xsd:complexContent>
            <xsd:extension base="tFlowNode"/>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.111 – XML–схема для элемента EventDefinition

```
<xsd:element name="eventDefinition" type="tEventDefinition"/>
<xsd:complexType name="tEventDefinition" abstract="true">
      <xsd:complexContent>
            <xsd:extension base="tBaseElement"/>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.112 – XML–схема для элемента ImplicitThrowEvent

```
<xsd:element name="implicitThrowEvent" type="tImplicitThrowEvent" substitutionGroup="flowElement"/>
<xsd:complexType name="tImplicitThrowEvent">
      <xsd:complexContent>
            <xsd:extension base="tThrowEvent"/>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.113 – XML–схема для элемента ImplicitThrowEvent

```
<xsd:element name="intermediateCatchEvent" type="tIntermediateCatchEvent" substitutionGroup="flowElement"/ >
```

```
<xsd:complexType name="tIntermediateCatchEvent">
      <xsd:complexContent>
            <xsd:extension base="tCatchEvent"/>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.114 – XML–схема для элемента IntermediateThrowEvent

```
<xsd:element name="intermediateThrowEvent" type="tIntermediateThrowEvent" substitutionGroup="flowEle-
      ment"/>
<xsd:complexType name="tIntermediateThrowEvent">
      <xsd:complexContent>
            <xsd:extension base="tThrowEvent"/>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.115 – XML–схема для элемента LinkEventDefinition

```
<xsd:element name="linkEventDefinition" type="tLinkEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tLinkEventDefinition">
      <xsd:complexContent>
            <xsd:extension base="tEventDefinition">
                  <xsd:sequence>
                        <xsd:element name="source" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element name="target" type="xsd:QName" minOccurs="0" maxOccurs="1"/>
                  </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string" use="required"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.116 – XML–схема для элемента LinkEventDefinition

```
<xsd:element name="messageEventDefinition" type="tMessageEventDefinition" substitutionGroup="eventDefini-
      tion"/>
<xsd:complexType name="tMessageEventDefinition">
      <xsd:complexContent>
            <xsd:extension base="tEventDefinition">
                  <xsd:sequence>
                        <xsd:element name="operationRef" type="xsd:QName" minOccurs="0" maxOccurs="1"/>
                  </xsd:sequence>
            <xsd:attribute name="messageRef" type="xsd:QName"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.117 – XML–схема для элемента Signal

```
<xsd:element name="signal" type="tSignal" substitutionGroup="reusableElement"/>
<xsd:complexType name="tSignal">
      <xsd:complexContent>
            <xsd:extension base="tRootElement">
                  <xsd:attribute name="name" type="xsd:string"/>
                  <xsd:attribute name="structureRef" type="xsd:QName"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.118 – XML–схема для элемента SignalEventDefinition

```
<xsd:element name="signalEventDefinition" type="tSignalEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tSignalEventDefinition">
      <xsd:complexContent>
            <xsd:extension base="tEventDefinition">
                  <xsd:attribute name="signalRef" type="xsd:QName"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.119 – XML–схема элемента StartEvent

```
<xsd:element name="startEvent" type="tStartEvent" substitutionGroup="flowElement"/>
<xsd:complexType name="tStartEvent">
    <xsd:complexContent>
        <xsd:extension base="tCatchEvent">
            <xsd:attribute name="isInterrupting" type="xsd:boolean" default="true"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.120 – XML–схема для элемента TerminateEventDefinition

```
<xsd:element name="terminateEventDefinition" type="tTerminateEventDefinition" substitutionGroup="eventDefini-
        tion"/>
<xsd:complexType name="tTerminateEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.121 – XML–схема для элемента ThrowEvent

```
<xsd:element name="throwEvent" type="tThrowEvent"/>
<xsd:complexType name="tThrowEvent" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tEvent">
            <xsd:sequence>
                <xsd:element ref="dataInput" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="dataInputAssociation" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="inputSet" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="eventDefinition" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="eventDefinitionRef" type="xsd:QName" minOccurs="0" maxOc-
                        curs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.122 – XML–схема для элемента TimerEventDefinition

```
<xsd:element name="timerEventDefinition" type="tTimerEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tTimerEventDefinition">
    <xsd:complexContent>
        <xsd:extension base="tEventDefinition">
            <xsd:choice>
                <xsd:element name="timeDate" type="tExpression" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="timeDuration" type="tExpression" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="timeCycle" type="tExpression" minOccurs="0" maxOccurs="1"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

10.5. Шлюзы

Шлюзы используются для контроля схождений и расхождений множественных **Потоков операций** в рамках **Процесса.** Необходимость в использовании **Шлюзов** отпадает в случае, если не возникает надобности контролировать **Поток операций**. Термин «**Шлюз**» подразумевает пропускное устройство, которое либо позволяет осуществлять переход через него, либо нет. Таким образом, как только *Токены* подходят к **Шлюзу**, то при его активизации они могут объединиться у входа в **Шлюз** и/или разделиться при выходе из **Шлюза**.

Графический элемент **Шлюз** представляет собой ромб, используемый во многих нотациях схем производственных процессов для изображения ветвления и знакомый большинству разработчиков моделей.

- Шлюз представляет собой ромб, который ДОЛЖЕН БЫТЬ выполнен одинарной тонкой линией (см. фигуру 10.102)
    - Текст, цвет, размер, а также линии, используемые для изображения **Шлюза**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».

Фигура 10.102 – Графический элемент Шлюз

Как и **Действия, Шлюзы** способны поглощать или формировать дополнительные токены, эффективно контролируя при этом исполнительную семантику **Процесса**. Основные отличия **Шлюза** от **Действия** заключаются в том, что посредством **Шлюза** нельзя отобразить выполнение какого-либо действия, а также в том, что **Шлюз** не оказывает никакого влияния на оценку **Процесса** (стоимость, время и т.д.)

При помощи **Шлюзов** можно указать все типы поведения **Потоков Операций Бизнес-Процесса**: условие/ветвление (Эксклюзивный, Включающий, Комплексный), слияние, раздвоение, соединение. Таким образом, если раньше изображение ромба обычно использовалось для отображения эксклюзивных условий, то благодаря **BPMN**, область его использования расширилась. На данный момент ромб может использоваться для отображения любого вида контроля **Потока операции**. Все **Шлюзы** имеют маркер, расположенный внутри графического элемента и указывающий на то, какой тип имеет тот или иной **Шлюз** (см. фигуру 10.103).

Эксклюзивный

Основанный на событиях

Параллельный, основанный на событиях

Включающий

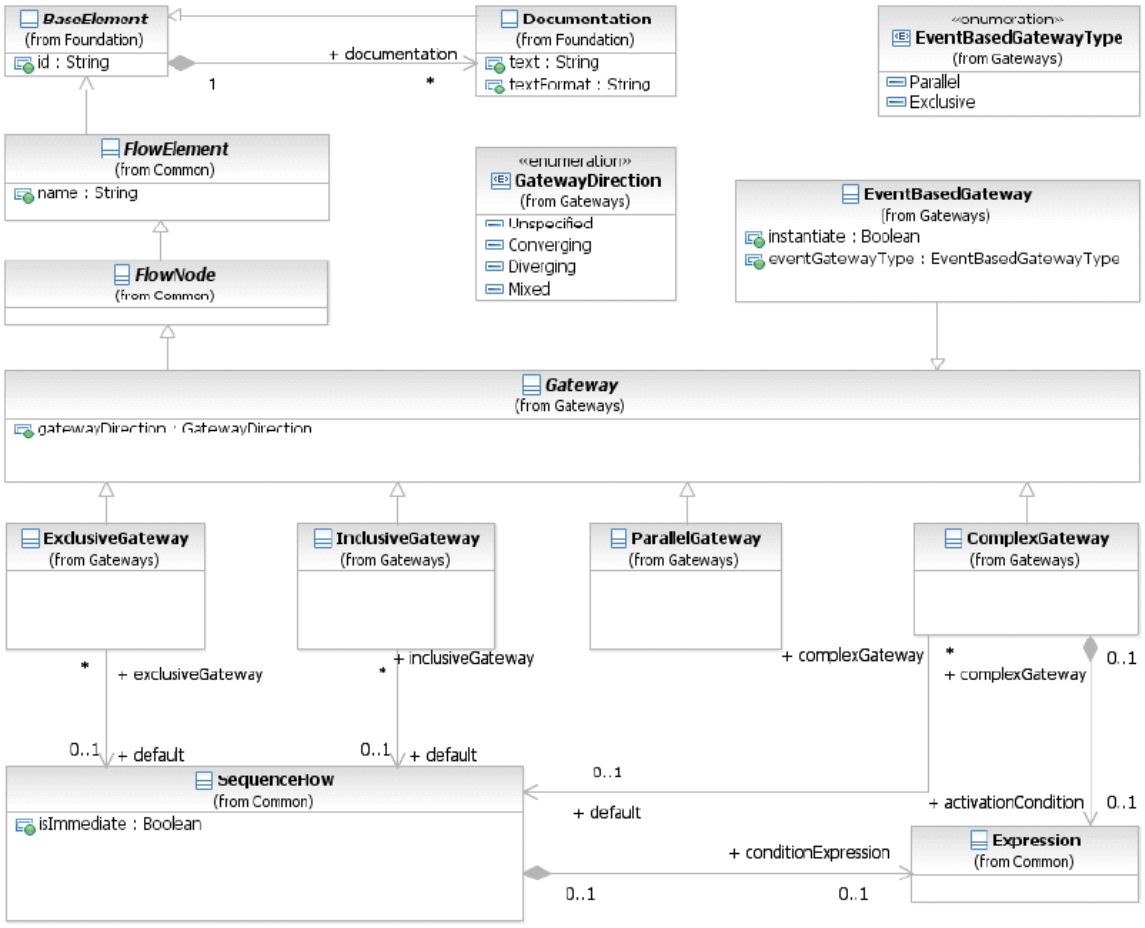
Комплексный

Параллельный

Фигура 10.103 – Типы Шлюзов

Шлюзы контролируют ход как сходящихся, так и расходящихся **Потоков Операций**. Это означает, что отдельно взятый Шлюз может быть соединен с несколькими Входящими и Исходящими **Потоками Операций** одновременно. Разработчики моделей и инструменты моделирования могут использовать два последовательных

Шлюза для соединения и последующего разделения **Потоков Операций**, т.е. одновременно использовать лишь одну его функцию.



Фигура 10.104 – Диаграмма классов элемента Gateway

В данном подразделе представлена общая информация о **Шлюзах**, а семантика исполнения описана в разделе 13.3.

10.5.1. Соединение с Потоками операций

Примечание: Несмотря на то, что **Шлюз** представляет собой ромб, Входящие и Исходящие **Потоки Операций** могут присоединяться к любой точке его границы, а не только к углам.

Данные правила относятся к использованию всех типов **Шлюзов.** Дополнительные правила соединения **Потока операций** для каждого типа **Шлюзов** описаны в другом подразделе этого документа.

- **Шлюз** МОЖЕТ БЫТЬ целью **Потока сообщений** и соединяться с любым количеством *Входящих* **Потоков Операций** (ни одного, один и более).
    - В случае, если **Шлюз** не имеет ни одного *Входящего* **Потока Операций**, а **Процесс** не содержит в своем составе **Стартового События**, то расхождение **Шлюза**, зависящее от типа **Шлюза** (см. ниже), ДОЛЖНО БЫТЬ выполнено при создании экземпляра **Процесса**.
- **Шлюз** МОЖЕТ являться источником **Процесса** и соединяться с любым количеством *Исходящих* **Потоков Операций** (ни одного, один и более).
- **Шлюз** ДОЛЖЕН БЫТЬ соединен либо с несколькими *Входящими*, либо с несколькими *Исходящими* **Потоками Операций**, т.е. он ДОЛЖЕН соединять или разводить маршруты.

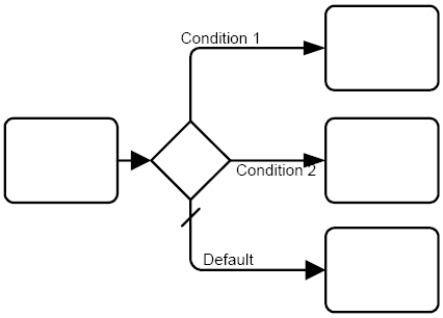
- **Шлюз**, значение атрибута `gatewayDirection` которого равно unspecified, МОЖЕТ иметь как несколько *входящих*, так и несколько *исходящих* **Потоков Операций**.
- **Шлюз**, значение атрибута `gatewayDirection` которого равно mixed, ДОЛЖЕН БЫТЬ одновременно соединен как с *входящими*, так и с *исходящими* **Потоками Операций**.
- **Шлюз**, значение атрибута `gatewayDirection` которого равно converging, ДОЛЖЕН БЫТЬ соединен с несколькими *Входящими* **Потоками Операций**. При этом количество *Исходящих* **Потоков Операций** для него ДОЛЖНО БЫТЬ ограничено.
- **Шлюз**, значение атрибута `gatewayDirection` которого равно diverging, ДОЛЖЕН БЫТЬ соединен с несколькими *Исходящими* **Потоками Операций**. При этом количество *Входящих* **Потоков Операций** для него ДОЛЖНО БЫТЬ ограничено.

10.5.2. Эксклюзивный Шлюз

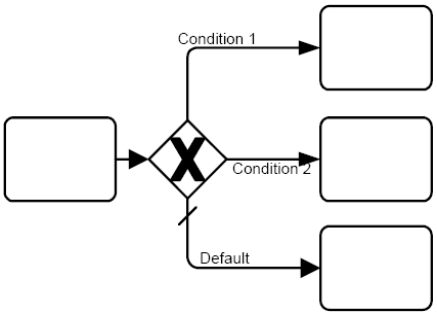
Эксклюзивные Шлюзы (Условия) включаются в состав **Бизнес-Процесса** для разделения **Потока Операций** на несколько альтернативных маршрутов. Для данного экземпляра **Процесса** может быть выбран лишь один из предложенных маршрутов.

Условие можно представить себе в виде вопроса, который появляется в какой-то точке **Процесса** и предполагает несколько вариантов ответов. Каждый из предлагаемых ответов ассоциирован с условным выражением, которое, в свою очередь, взаимодействует с направленным от **Шлюза** *Исходящим* **Потоком Операций**.

- Графический элемент **Эксклюзивный Шлюз** МОЖЕТ содержать внутренний маркер, выполненный в виде «X» (см. фигуру 10.106), что позволяет отличить данный тип **Шлюза** от других. Использование данного маркера НЕ является ОБЯЗАТЕЛЬНЫМ условием (см. фигуру 10.105).
    - На диаграмме НЕ ДОЛЖНО быть одновременно нескольких **Шлюзов** данного типа, содержащих маркер и отображаемых без маркера. Все такие **Шлюзы** ДОЛЖНЫ либо иметь маркер, либо отображаться без него.



Фигура 10.105 – Отображаемое без маркера Эксклюзивное Условие (Шлюз), основанное на данных

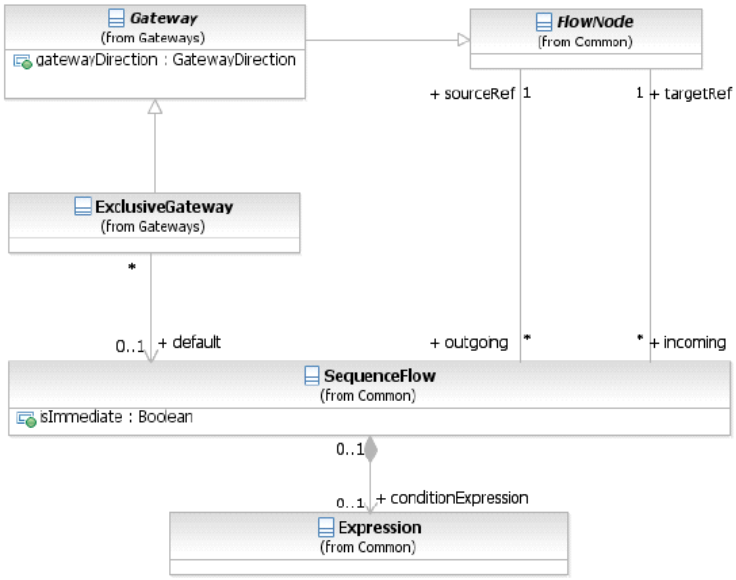


Фигура 10.106 – Отображаемое с маркером Эксклюзивное Условие (Шлюз), основанное на данных

Примечание: Первый из вышеприведенных фрагментов отображает предпочтительное изображение **Эксклюзивного Шлюза**. Это означает, что в данном документе **Эксклюзивные Шлюзы** отображаются без маркера.

Маршрут по умолчанию может быть указан или не указан. В случае, если маршрут по умолчанию указан, значит, ни одно из значений условного `выражения` не равно «*true*». Если же маршрут по умолчанию не указан, а **Процесс** выполняется так, что ни одно из значений условного `выражения` не равно «*true*», возникает исключение.

Объединяющий **Эксклюзивный Шлюз** используется для объединения нескольких альтернативных маршрутов в один. *Токен* каждого из *Входящих* **Потоков Операций** назначается на *Исходящий* **Поток Операций** без синхронизации.



Фигура 10.107 – Диаграмма классов элемента ExclusiveGateway

Элемент `ExclusiveGateway` наследует атрибуты и ассоциации элемента `Gateway` (см. таблицу 8.46). Таблица 10.123 содержит информацию о дополнительных атрибутах и ассоциациях элемента `ExclusiveGateway`.

Таблица 10.123 Атрибуты и ассоциации элемента ExclusiveGateway

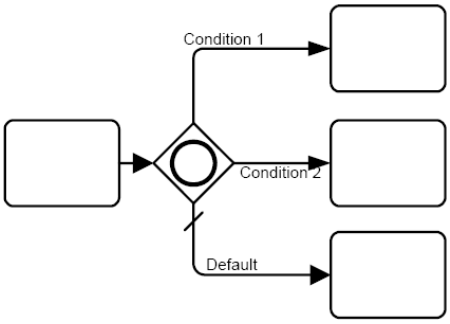
Название атрибута	Описание/использование
default: SequenceFlow [0..1]	Данный атрибут указывает **Поток операций**, который наследует *токен* в случае, если ни одно из значений `условного выражения` другого Исходящего **потока операций** не равно «*true*». **Поток операций** по умолчанию не должен содержать `условного выражения`. Любые такие `выражения` ДОЛЖНЫ БЫТЬ проигнорированы.

10.5.3. Неэксклюзивный Шлюз

Разделяющие **Неэксклюзивные Шлюзы** (Условия) используются для разделения **Потока операций** на несколько альтернативных и параллельных маршрутов. Для данного экземпляра **Процесса** может быть выбран

лишь один из предложенных маршрутов. В отличие от **Эксклюзивных Шлюзов**, значения всех используемых для данного типа **Шлюзов** условных `выражений` определены. В данном случае значение одного из условных `выражений`, равное «*true*», не исключает определения значения любого другого условного `выражения`. *Токен* проходит через все **Потоки операций**, определенные как «*true*». Поскольку каждый маршрут является независимым, МОГУТ БЫТЬ использованы любые комбинации этих маршрутов (от нуля и более); однако в действительности, должен быть выбран по-меньшей мере один маршрут.

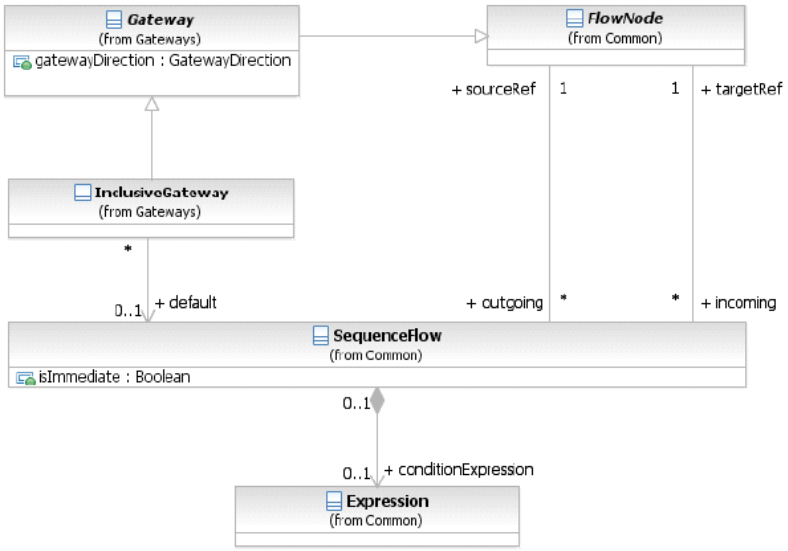
- Графический элемент **Неэксклюзивный Шлюз** ДОЛЖЕН содержать внутренний маркер, выполненный в виде круга (см. фигуру 10.108), что позволяет отличить данный тип **Шлюза** от других.



Фигура 10.108 – Пример Неэксклюзивного Шлюза

Маршрут по умолчанию может быть указан или не указан. В случае, если маршрут по умолчанию указан, значит, ни одно из значений условного выражения не равно «*true*». Если же маршрут по умолчанию не указан, а **Процесс** выполняется так, что ни одно из значений условного выражения не равно «*true*», возникает исключение.

Объединяющие **Неэксклюзивные Шлюзы** используются для объединения нескольких альтернативных и параллельных маршрутов в один. *Токен* потока управления, поступающий в **Неэксклюзивный Шлюз**, МОЖЕТ БЫТЬ синхронизирован с другими *токенами*, поступающими в данный **Шлюз** позже. Предпочтительный способ синхронизации маршрутов посредством **Неэксклюзивного Шлюза** описан в данном подразделе.



Фигура 10.109 – Диаграмма классов элемента InclusiveGateway

Элемент `InclusiveGateway` наследует атрибуты и ассоциации элемента `Gateway` (см. таблицу 8.46). Таблица 10.124 содержит информацию о дополнительных атрибутах и ассоциациях элемента `InclusiveGateway`.

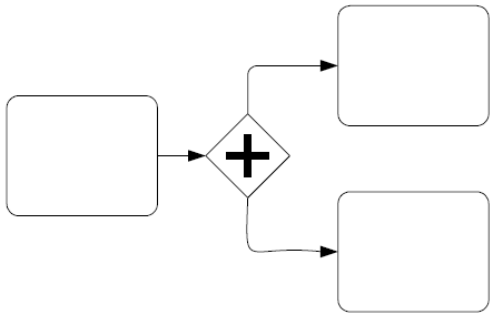
Таблица 10.124 Атрибуты и ассоциации элемента InclusiveGateway

Название атрибута	Описание/использование
default: SequenceFlow [0..1]	Данный атрибут указывает **Поток операций**, который наследует *токен* в случае, если ни одно из значений условного `выражения` других **Потоков операций** не равно «*true*». **Поток операций** по умолчанию не должен содержать условного `выражения`. Любые такие `выражения` ДОЛЖНЫ БЫТЬ проигнорированы.

10.5.4. Параллельный Шлюз

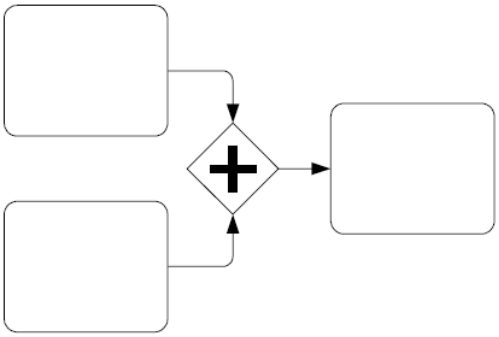
Параллельные Шлюзы используются для синхронизации (объединения) и создания параллельных маршрутов.

- Графический элемент **Параллельный Шлюз** ДОЛЖЕН содержать внутренний маркер, выполненный в виде знака «+» (см. фигуру 10.110), что позволяет отличить данный тип **Шлюза** от других.



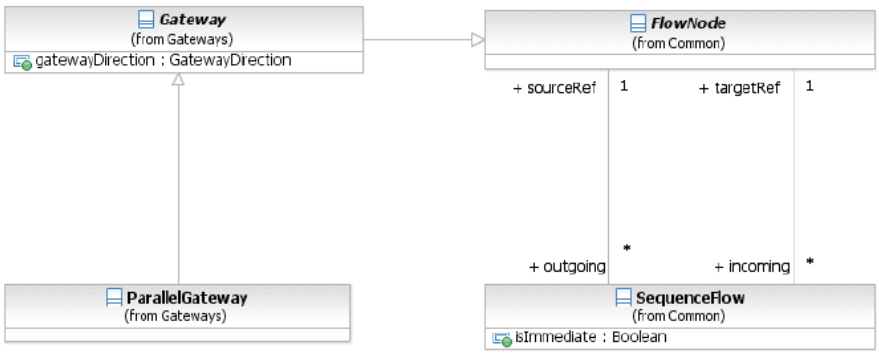
Фигура 10.110 – Пример Параллельного Шлюза

Параллельный Шлюз используется для синхронизации параллельных маршрутов.



Фигура 10.111 – Пример Параллельного Шлюза, используемого для синхронизации

С помощью **Параллельного Шлюза** параллельные маршруты создаются без необходимости проверки каких-либо условий. Благодаря работе **Шлюза** данного типа, любой *Исходящий* **Поток Операций** получает *токен*. **Параллельный Шлюз** ожидает поступления всех *Входящих* **Потоков Операций** до того, как от него отойдет *Исходящий* **Поток Операций**.



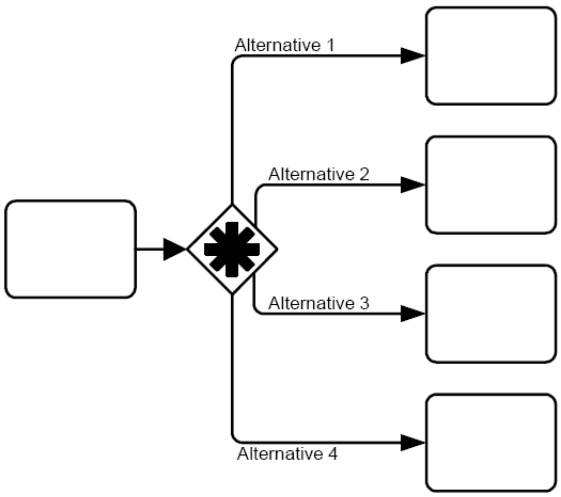
Фигура 10.112 – Диаграмма классов элемента ParallelGateway

Элемент **ParallelGateway** наследует атрибуты и ассоциации элемента **Gateway** (см. таблицу 8.46), однако, не может иметь каких-либо других атрибутов и ассоциаций.

10.5.5. Комплексные Шлюзы

Комплексные Шлюзы используются для моделирования поведения **Потоков Операций** в случае комплексной синхронизации. Для точного описания такого поведения используется `выражение ExpressionactivationCondition`. Данное `выражение` может использоваться, к примеру, для указания того, что для активации **Шлюза** необходимо наличие трех *токенов Входящих* **Потоков Операций** из пяти. Как и при разделяющем поведении **Неэксклюзивного Шлюза,** условиями *Исходящих* **Потоков Операций** определяется то, какие *токены* сформирует **Шлюз.** Токены, прибывающие позже по двум оставшимся маршрутам, вызывают сброс **Шлюза**, а для *Исходящих* **Потоков Операций** формируются новые *токены*. Для определения того, будет ли **Шлюз** ожидать оставшиеся *токены* для того, чтобы осуществился его сброс, используется семантика синхронизации **Неэксклюзивного Шлюза.**

- Графический элемент **Комплексный Шлюз** ДОЛЖЕН содержать внутренний маркер, выполненный в виде символа «*» (см. фигуру 10.113), что позволяет отличить данный тип Шлюза от других.

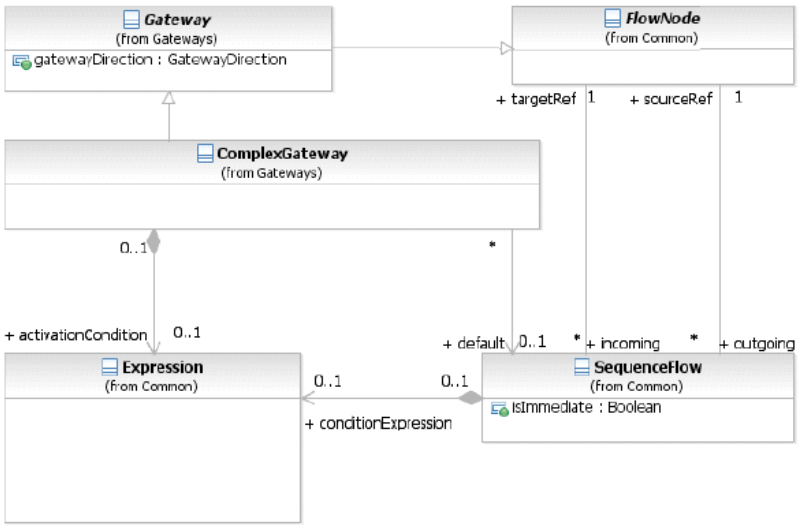


Фигура 10.113 – Пример Комплексного Шлюза

В отличие от других типов **Шлюзов**, **Комплексный Шлюз** имеет различные внутренние состояния, определяемые с помощью булевского атрибута `waitingForStart`. Исходное значение данного атрибута равно «*true*», но после активации оно меняется на «*false*». Этот атрибут может использоваться для определения того, где при активации **Шлюза** для *Исходящих* **Потоков Операций** формируются *токены*, а также того, где они

формируются при его сбросе. РЕКОМЕНДУЕТСЯ выбрать, будут ли *токены* формироваться для каждого из *Исходящих* **Потоков Операций** при активации **Шлюза** или при его сбросе, но не одновременно. По-меньшей мере один *Исходящий* **Поток Операций** должен получить *токен* при активации, но в таком случае при сбросе **Шлюза** *токен* формироваться НЕ ДОЛЖЕН.

На фигуре 10.114 отображена диаграмма класса `ComplexGateway`.



Фигура 10.114 – Диаграмма классов элемента ComplexGateway

Элемент `ComplexGateway` наследует атрибуты и ассоциации элемента `Gateway` (см. таблицу 8.46). Таблица 10.125 содержит информацию о дополнительных ассоциациях элемента `ComplexGateway`.

Таблица 10.125 Ассоциации элемента ComplexGateway

Название атрибута	Описание/использование
activationCondition: Expression [0..1]	Определяет, *токены* каких *Входящих* **Потоков Операций** будут синхронизированы для активации **Шлюза**.
default: SequenceFlow [0..1]	Указывает на **Поток Операций**, который получит *токен* в случае, если ни одно из `условных выражений` других **Потоков Операций** не равно «*true*». **Поток Операций** по умолчанию не должен содержать `условного выражения`. Любые такие выражения ДОЛЖНЫ БЫТЬ проигнорированы.
activationCount: integer	В ходе выполнения ссылается на количество *токенов Входящего* **Потока Операций**, направленного к **Комплексному Шлюзу**.
waitingForStart: boolean = true	Указывает на внутреннее состояние **Комплексного Шлюза**, который может находиться в ожидании активации (=*true*) или сброса (=*false*).

10.5.6. Шлюз, основанный на Событиях

В **Процессе Шлюзы**, основанные на **Событиях**, представляет собой точку ветвления, в которой направления альтернативных маршрутов зависят от происходящих **Событий**, а не от определения значений выражений, использующих данные **Процесса**, как это происходит при использовании **Эксклюзивного** и **Неэксклюзивного Шлюзов**. Определенное **событие**, чаще всего – получение **сообщения**, определяет то, какой из маршрутов будет выбран в данном случае. В основном, *решение* принимается одним из *Участников* **Процесса** на основании данных, которые в самом **Процессе** не видны. В этом случае и становится нужен **Шлюз**, основанный на **Событиях**.

Например, компания ждет подтверждения заказчиком выполнения ряда **Действий**. В случае положительного ответа эти **Действия** будут выполнены. В случае отрицательного ответа компания выполнит ряд других **Действий**. Ответ заказчика – **Сообщение** определенного рода - определяет выбор необходимого маршрута. Таким образом, сообщения «Да» и «Нет» являются разными, а не просто одинаковыми сообщениями с разными значениями свойства. Получение сообщения может быть выполнено посредством **Промежуточного события Сообщение** или **Задачи Получение**. Кроме *триггера* **Сообщения** могут использоваться и другие *триггеры* **Промежуточного события**, например, Таймер.

Как и все остальные типы **Шлюзов, Шлюз**, основанный на **Событиях**, отображается на диаграмме **Процесса** в виде ромба, внутри которого расположен соответствующий маркер.

- **Шлюз**, основанный на **Событиях**, представляет собой ромб, который ДОЛЖЕН БЫТЬ выполнен одинарной тонкой линией.
    - Текст, цвет, размер, а также линии, используемые для изображения **Шлюза**, ДОЛЖНЫ соответствовать правилам, указанным в разделе «Использование Текста, Цвета и Линий в Моделировании Диаграмм».
- Графический элемент **Шлюз**, основанный на **Событиях**, содержит внутренний маркер, который ДОЛЖЕН быть идентичным маркеру *обрабатывающего триггер* **Промежуточного события** (см. фигуру 10.115).

10.115 – Графический элемент Шлюз, основанный на Событиях

В отличие от поведения других типов **Шлюзов**, поведение **Шлюза**, основанного на **Событиях**, определяется не его типом, а конфигурацией элементов.

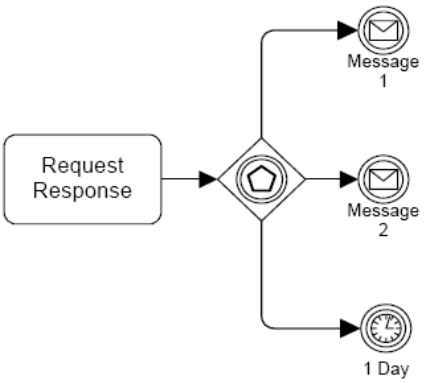
- **Шлюз**, основанный на **Событиях**, ДОЛЖЕН содержать два или более *Исходящих* **Потоков Операций**.
    - *Исходящие* **Потоки Операций**, направленные от **Шлюза** данного типа, НЕ ДОЛЖНЫ содержать `условных выражений`.

Элементы, являющиеся целью направленных от **Шлюза** *Исходящих* **Потоков Операций**, являются частью конфигурации **Шлюза**.

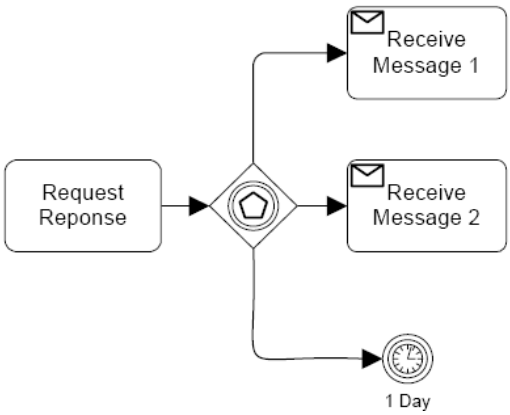
- Конфигурация **Шлюза**, основанного на **Событиях**, задается посредством *Исходящих* **потоков операций**, целями которых являются **Промежуточные события** или **Задачи Получение** в любых комбинациях (см. фигуры 10.116 и 10.117). Исключениями являются ситуации, когда:
    - Для конфигурации используются **Промежуточные события Сообщение**. В данном случае для конфигурации **Шлюза Задачи Получение** использоваться НЕ ДОЛЖНЫ (и наоборот).
        - **Задачи Получение**, используемые для конфигурации **Шлюза**, НЕ ДОЛЖНЫ быть соединены с какими-либо **Промежуточными событиями**.

- Только следующие типы **Промежуточных событий** верны для **Шлюза**, основанного на **События**: `Сообщение, Сигнал, Таймер, Условие, Множественный` (включающий лишь вышеуказанные *триггеры*). Таким образом, для данного **Шлюза** не используются **Промежуточные события** следующих типов: `Ошибка, Отмена, Компенсация, Связь`.

- Элементы, являющиеся целями в конфигурации **Шлюза**, основанного на **Событиях**, НЕ ДОЛЖНЫ иметь дополнительных *Входящих* **потоков операций** (исходящих не от данного **Шлюза**).



Фигура 10.116 – Пример Шлюза, основанного на Событиях, использующего Промежуточные события Сообщение



Фигура 10.117 – Пример Шлюза, основанного на Событиях, использующего Задачи Получение

Когда запускается первое **Событие**, входящее в состав конфигурации **Шлюза**, основанного на **Событиях**, используется отходящий от этого **События** маршрут (в данном случае *токен* отсылается с исходящим от **События Потоком операций**). После этого все остальные маршруты конфигурации **Шлюза** являются неверными. В общем, конфигурация такого **Шлюза** представляет собой подобие состязания, где побеждает **Событие**, запущенное раньше остальных.

Существуют различные типы **Шлюзов**, основанных на **Событиях**, которые могут быть использованы в начале **Процессе**. Поведение **Шлюза** и, соответственно, его маркер будут изменяться.

Шлюзы, основанные на **Событиях**, могут быть использованы для запуска экземпляра **Процесса**. Значение атрибута `instantiate` **Шлюза** по умолчанию равно «*false*». Однако если оно изменяется на «*true*», то при запуске первого в конфигурации **Шлюза События** запускается и экземпляр **Процесса.**

- В случае, если значение атрибута `instantiate` **Шлюза** равно «*true*», то данный **Шлюз** будет иметь маркер, идентичный маркеру **Стартового события Множественного типа** (см. фигуру 10.118).

Фигура 10.118 – Эксклюзивный, основанный на Событиях Шлюз, используемый для запуска экземпляра Процесса.

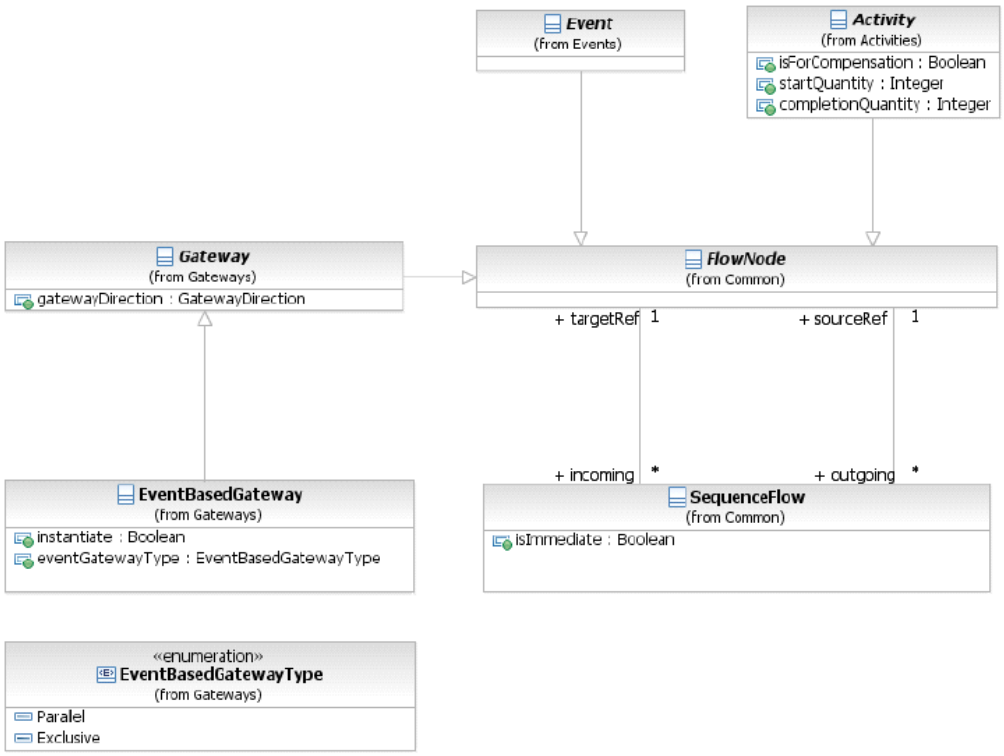
Для того, чтобы данный **Шлюз** мог инициировать запуск экземпляра **События,** НЕ ДОЛЖЕН быть соединен ни с одним Входящим **Потоком Операций**.

В некоторых случаях разработчик модели может инициировать запуск экземпляра **Процесса** посредством одного сообщения из набора, хотя для выполнения этого экземпляра **Процесса** необходим весь набор сообщений. В такой ситуации используется другой тип **Шлюза**, основанного на **Событиях**.

- В случае, если значение атрибута `instantiate` **Шлюза** равно «*true*», а значение атрибута `eventGatewayType` – «`Parallel`», то данный **Шлюз** будет иметь маркер, идентичный маркеру **Стартового события Параллельного Множественного типа** (см. фигуру 10.119).
    - Для того, чтобы значение атрибута `eventGatewayType` было равно «`Parallel`», значение атрибута `instantiate` **Шлюза** ДОЛЖНО БЫТЬ равно «*true*». Это означает, что для **Шлюза**, основанного на **Событиях**, который не используется для запуска экземпляра **Процесса**, значение атрибута `eventGatewayType` ДОЛЖЕН БЫТЬ равно «`Exclusive`», как для стандартного **Параллельного Шлюза**, используемого для объединения параллельных **Событий** в середине **Процесса**.

Фигура 10.119 – Параллельный, основанный на Событиях Шлюз, используемый для запуска экземпляра Процесса.

Параллельный Шлюз, основанный на **Событиях**, также представляет собой подобие состязания, где, однако, при запуске первого **События** и экземпляра **Процесса** остальные **События** конфигурации **Шлюза** с дистанции не снимаются. Эти оставшиеся **События** продолжают ждать своего запуска до момента успешного завершения **Процесса**. В данном случае, **Сообщение**, запускающее **События** конфигурации **Шлюза**, ДОЛЖНО обладать той же корреляционной информацией.



Фигура 10.120 – Диаграмма классов элемента EventBasedGateway

Элемент `EventBasedGateway` наследует атрибуты и ассоциации элемента `Gateway` (см. таблицу 8.46). Таблица 10.127 содержит информацию о дополнительных атрибутах и ассоциациях элемента `EventBasedGateway`.

Таблица 10.127 Атрибуты и ассоциации элемента EventBasedGateway

Название атрибута	Описание/использование
instantiate: boolean = false	В случае, если значение данного атрибута равно «*true*», получение одного из **Сообщений** запустит экземпляр **Процесса**.
eventGatewayType: EventGate-wayType = Exclusive { Exclusive | Parallel }	Атрибут `eventGatewayType` определяет поведение **Шлюза**, используемого для запуска экземпляра **Процесса** (см. описание выше). Если значение атрибута `instantiate` равно «*true*», то значение данного атрибута не может быть никаким другим, кроме «`parallel`».

Основанный на **Событиях Шлюз** (или множество таких **Шлюзов**) может быть помещен в начало **Потока операций Процесса** и не иметь *Входящих* **Потоков Операций**. Возможно одновременное использование стандартных **Стартовых событий** и основанных на данных **Шлюзов**.

10.5.7. Представление XML-схемы для пакета Шлюза

Таблица 10.128 - XML-схема для элемента ComplexGateway

```
<xsd:element name="complexGateway" type="tComplexGateway" substitutionGroup="flowElement"/>
<xsd:complexType name="tComplexGateway">
      <xsd:complexContent>
```

```
            <xsd:extension base="tGateway">
                <xsd:sequence>
                    <xsd:element name="activationCondition" type="tExpression" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
                <xsd:attribute name="default" type="xsd:IDREF"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.129 - XML-схема для элемента EventBasedGateway

```
<xsd:complexType name="tEventBasedGateway">
        <xsd:complexContent>
            <xsd:extension base="tGateway">
                <xsd:attribute name="instantiate" type="xsd:boolean" default="false"/>
                <xsd:attribute name="eventGatewayType" type="tEventBasedGatewayType" default="Exclusive"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tEventBasedGatewayType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Exclusive"/>
            <xsd:enumeration value="Parallel"/>
        </xsd:restriction>
</xsd:simpleType>
```

Таблица 10.130 - XML-схема для элемента ExclusiveGateway

```
<xsd:element name="exclusiveGateway" type="tExclusiveGateway" substitutionGroup="flowElement"/>
<xsd:complexType name="tExclusiveGateway">
        <xsd:complexContent>
            <xsd:extension base="tGateway">
                <xsd:attribute name="default" type="xsd:IDREF" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.131 - XML-схема для элемента InclusiveGateway

```
<xsd:element name="gateway" type="tGateway" abstract="true"/>
<xsd:complexType name="tGateway">
        <xsd:complexContent>
            <xsd:extension base="tFlowElement">
                <xsd:attribute name="gatewayDirection" type="tGatewayDirection" default="Unspecified"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tGatewayDirection">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Unspecified"/>
            <xsd:enumeration value="Converging"/>
            <xsd:enumeration value="Diverging"/>
            <xsd:enumeration value="Mixed"/>
        </xsd:restriction>
</xsd:simpleType>
```

Таблица 10.132 - XML-схема для элемента Gateway

```
<xsd:element name="inclusiveGateway" type="tInclusiveGateway" substitutionGroup="flowElement"/>
<xsd:complexType name="tInclusiveGateway">
        <xsd:complexContent>
            <xsd:extension base="tGateway">
                <xsd:attribute name="default" type="xsd:IDREF" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.133 - XML-схема для элемента ParallelGateway

```
<xsd:element name="parallelGateway" type="tParallelGateway" substitutionGroup="flowElement"/>
<xsd:complexType name="tParallelGateway">
      <xsd:complexContent>
            <xsd:extension base="tGateway"/>
      </xsd:complexContent>
</xsd:complexType>
```

10.6. Компенсация

Компенсация подразумевает возврат к исходной точке цепочки успешно выполненных действий по причине того, что результат этих действий или побочные эффекты больше не представляют ценности и должны быть отменены. *Компенсации* не происходит в случае, если **Действие** ещё запущено; в данном случае оно должно быть отменено. **Отмена**, в свою очередь, может привести к *компенсации* успешно выполненных операций в составе действия (например, в **Подпроцессе**).

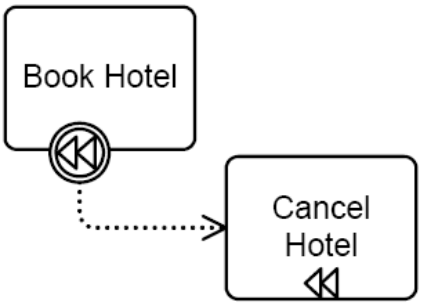
Компенсация выполняется при помощи *обработчика компенсации*, который выполняет шаги, необходимые для отмены результатов **Действия**. В **Подпроцессе** *обработчик компенсации* имеет доступ к данным этого **Подпроцесса** в момент его завершения, т.е. делает моментальный снимок набора данных **Подпроцесса** («snapshot data»).

Компенсация запускается *инициирующим триггер* **Событием Компенсация**, вызываемым обычно *обработчиком ошибок* (если **Событие** является частью процедуры отмены **Действия**) либо другим *обработчиком компенсации*, что происходит реже. Такое **Событие** определяет **Действие**, которое будет *компенсировано* (скрыто или явно).

10.6.1. Обработчик компенсации

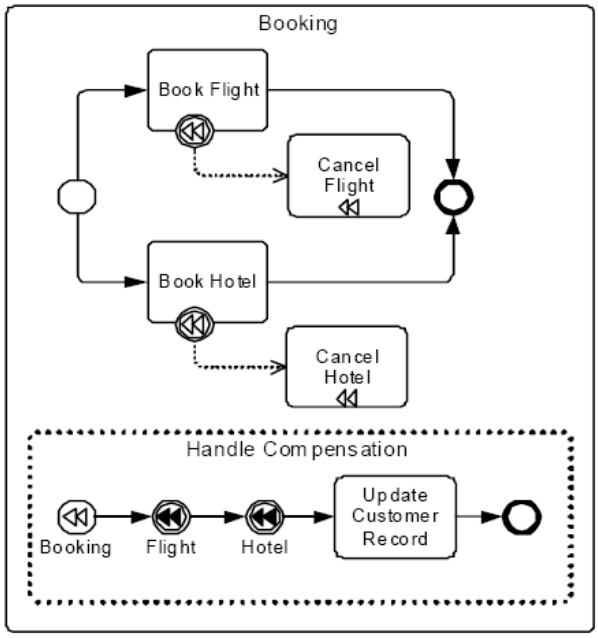
Обработчик компенсации представляет собой набор **Действий**, не соединенных с остальными частями модели **Процесса**. Начальной точкой *обработчика компенсации* является *обрабатывающее триггер* **Событие Компенсация**. Такое **Событие** либо присоединено к границе **Действия**, либо является **Стартовым событием** данного *обработчика* (например, в **Событийном подпроцессе Компенсация**).

Обработчик компенсации, присоединенный к **Действию** посредством расположенного на его границе **События**, может производить *компенсацию* **Действия** лишь по типу «черного ящика». Такой вид *компенсации* создается при помощи особого **Действия Компенсация**, присоединенного к **Событию** на границе **Действия** посредством **Ассоциации** (см. фигуру 10.121). Графический элемент **Действия Компенсация** (**Задача** или **Подпроцесс**) имеет внутренний маркер, указывающий на то, что данное **Действие** является *компенсацией* и находится за пределами *Стандартного потока операций* **Процесса**.



Фигура 10.121 – Компенсация, использующая Событие на границе Действия

Событийный Подпроцесс Компенсация находится внутри **Процесса** или **Подпроцесса** (см. фигуру 10.122). Как и **Действие Компенсация**, **Событийный Подпроцесс Компенсация** находится за пределами *Стандартного потока операций* **Процесса**. Если граница графического элемента такого **Подпроцесса** выполнена пунктиром, это означает, что **Подпроцесс** имеет доступ к данным родительского **Процесса,** т.е. делает моментальный снимок набора данных **Процесса** в момент его завершения («snapshot data»). **Событийный Подпроцесс Компенсация** может время от времени вызывать *компенсацию* **Действий** в составе родительского **Процесса**.



Фигура 10.122 – Обзор диаграммы класса

При необходимости можно указать, что **Подпроцесс** может быть *компенсирован* без предварительного указания *обработчика компенсации*. В случае, если значение атрибута `compensable` **Подпроцесса** указано, это означает, что указана и *компенсация* по умолчанию (т.е. время от времени производится компенсация всех успешно выполненных **Действий Подпроцесса**).

На фигуре 10.122 отображен заказной **Событийный подпроцесс Компенсация**, вызываемый **Стартовым событием Компенсация**. Обратите внимание, что данный *обработчик компенсации* отличается от *компенсации*, установленной по умолчанию, тем, что выполняет **Действия Компенсация** в порядке, отличном от порядка в следующем примере. Он также содержит дополнительное **Действие**, дополняющее логику **Процесса**, которое не может выводиться из самого **Подпроцесса**.

10.6.2. Механизмы запуска компенсации

Запуск Компенсации осуществляется посредством *инициирующего компенсацию* **События**, как **Промежуточного**, так и **Конечного**. Оно ссылается на **Действие**, компенсацию которого необходимо произвести. Если же из контекста ясно, какое **Действие** будет компенсировано, указывать ссылку не обязательно, – **Событие** относится к текущему **Действию**. Стандартным сценарием **Запуска Компенсации** является ситуация, в которой строчный обработчик возникающих в **Подпроцессе** *ошибок* не может устранить ошибку, в результате чего запускается *компенсация* данного **Подпроцесса**. Если в общем контексте не указано ни одного предназначенного для *компенсации* **Действия,** все **Действия Подпроцесса** будут компенсированы. По умолчанию, *компенсация* запускается синхронно. Это означает, что *инициирующее компенсацию* **Событие** будет ожидать завершения работы запущенного *обработчика компенсации*. *Компенсация* также может быть запущена вне зависимости от того, завершилась ли работа *обработчика компенсации* или нет. Для этого атрибут `waitForCompletion` **События** должен иметь значение «*false*».

Множественные экземпляры обычно используются в **Циклах** и **Многоэкземплярных Подпроцессах**. Каждый из них имеет свой собственный экземпляр **Событийного Подпроцесса Компенсация**, имеющего доступ к определенным данным **Подпроцесса**, которые были актуальными на время выполнения каждого из экземпляров. **Запуск компенсации Многоэкземплярного Подпроцесса** в индивидуальном порядке инициирует осуществление *компенсации* всех экземпляров данного **Подпроцесс**а в рамках контекста. В случае, если указано, что *компенсация* должна быть выполнена посредством *обработчика компенсации*, расположенного на границе **Подпроцесса**, такой обработчик также запускается для каждого экземпляра этого **Подпроцесса** в рамках контекста.

10.6.3. Взаимодействие обработчика ошибки с компенсацией

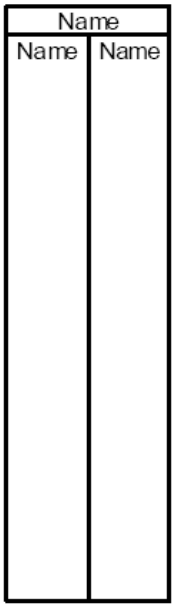
Следующее описание содержит информацию о том, как взаимодействуют *обработчик событий* и *компенсация*:

- *Компенсация* руководствуется принципом предполагаемого прерывания (presumed abort principle), следствием чего является аннулирование *компенсацией* результатов выполнения **Действия**.
- Когда ход выполнения **Действия** нарушается по причине возникновения *ошибки*, *обработчик ошибки* должен гарантировать, что в случае завершения своей работы (хотя бы раз) не возникнет необходимость в *компенсации*.
- Если для какого-либо **Подпроцесса** или *ошибки* не указан **Событийный Подп**роцесс, содержащий *ошибку*, то в случае возникновения *ошибки* поведением по умолчанию должен быть автоматический вызов *компенсации* для всех содержащихся в данном **Подпроцессе Действий**, гарантирующий возможность проверки и мониторинга.

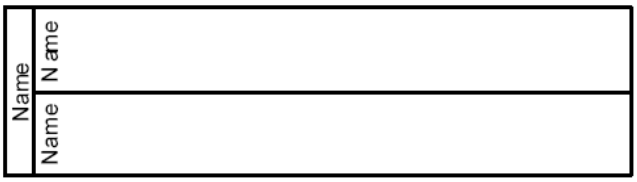
10.7. Дорожки

Дорожка представляет собой подразделение внутри **Процесса** (как правило, внутри **Пула**) и простирается на всю длину данного **Процесса** как горизонтально (см. фигуру 10.124), так и вертикально (см. фигуру 10.123). По желанию разработчика модели, текст, относящийся к **Дорожке** (например, название **Дорожки** или атрибут какого-либо элемента **Процесса**), может быть помещен в любом месте данной **Дорожки** и располагаться в любом направлении. В представленных ниже примерах название **Дорожки** располагается в виде баннера в левой части (в горизонтальном **Пуле**) или вверху (вертикальном **Пуле**) рядом с линией, отделяющей название **Дорожки** от названия **Пула**. Однако такое размещение текста не является обязательным.

- **Дорожка** представляет собой стандартный прямоугольник, который ДОЛЖЕН БЫТЬ выполнен одинарной жирной линией (см. фигуры 10.123 и 10.124).
    - Название **Дорожки** МОЖЕТ БЫТЬ помещено в любом месте данного графического элемента, однако, оно НЕ ДОЛЖНО отделяться от остального содержимого **Дорожки** границей в виде линии (за исключением случаев, когда в **Дорожке** содержатся дополнительные **Дорожки** (sub-Lanes)).

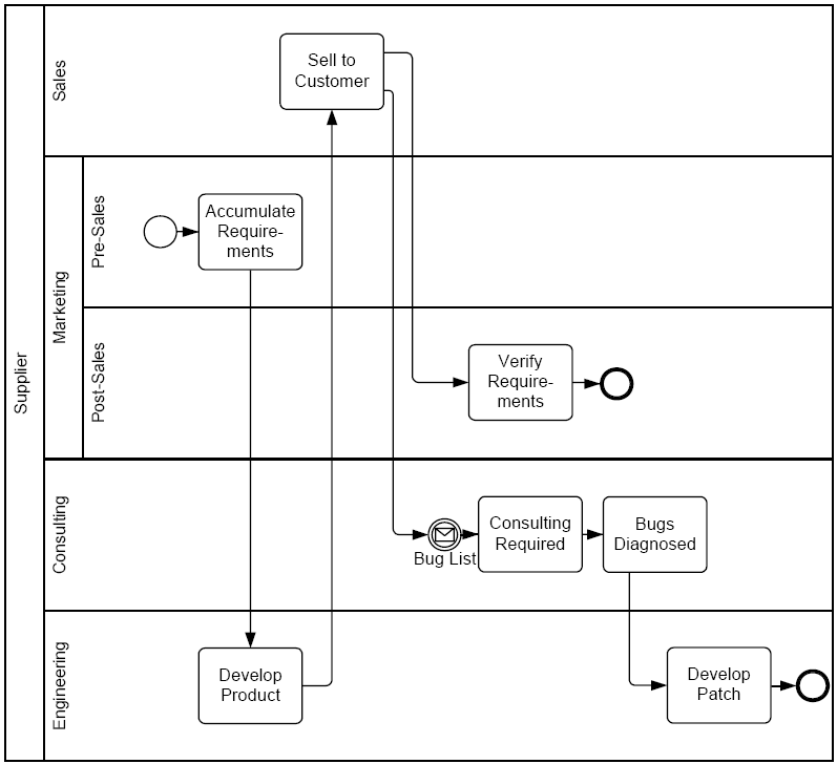


Фигура 10.123 – Пул, содержащий две вертикальные Дорожки



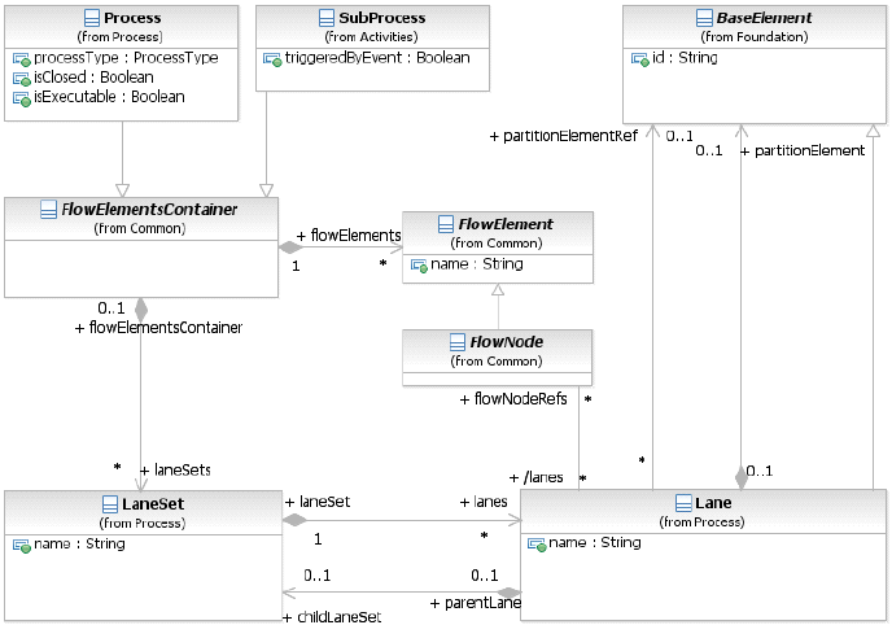
Фигура 10.124 – Пул, содержащий две горизонтальные Дорожки

Дорожки используются для организации и категоризации **Действий**, расположенных внутри **Пула**. Содержание **Дорожек** зависит от разработчика модели. Спецификация **BPMN** не указывает на то, какую функцию в **Процессе** должны выполнять **Дорожки**, которые часто используются в качестве внутренних ролей (например, Менеджер, Партнер), систем (например, прикладные системы предприятия), внутренних отделов (например, поставки, бюджет) и т.д. Также **Дорожки** могут быть вложены (см. фигуру 10.125) или установлены в матрице. Например, может существовать внешнее множество **Дорожек** отделов предприятия, а также внутреннее множество **Дорожек** для ролей, существующих внутри каждого отдела.



Фигура 10.125 – отображение диаграммы класса Lane. Указанная Дорожка содержится в элементе LaneSet Процесса.

На фигуре 10.126 отображена диаграмма классов элемента `Lanes`. Если значение данного элемента указано, оно хранится в элементе `LaneSet`, который содержится в **Процессе**.



Фигура 10.126 – Диаграмма классов элемента Lane

Посредством элемента `LaneSet` определяется контейнер для одной или более **Дорожек**. В **Процессе** могут содержаться один или несколько элементов `LaneSet`, каждый из которых (а также его **Дорожка**) могут по-другому разбивать *узлы Потока операций*.

Элемент `LaneSet` наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.134 содержит информацию о дополнительных атрибутах и ассоциациях элемента `LaneSet`.

Таблица 10.134 Атрибуты и ассоциации элемента LaneSet

Название атрибута	Описание/использование
name: sting [0..1]	Название `LaneSet`. На диаграмме **BPMN** элемент `LaneSet` не отображается, поэтому его название также не отображается на ней.
process: Process	**Процесс**, содержащий данный элемент `LaneSet`.
lanes: Lane [0..*]	Одно или более подразделений внутри графического элемента **Дорожка**, определяющие характерное для данного `LaneSet` разделение.
parentLane: Lane [0..1]	Ссылка на **Дорожку,** являющуюся родительской для данного `LaneSet`. The reference to a **Lane** element which is the parent of this `LaneSet`.

Отдельно взятый графический элемент **Дорожка** является одним подразделением элемента `LaneSet` и может определять разделительный элемент, который, в свою очередь, определяет значение и тип элемента. Посредством разделительного элемента может указываться список *узлов Потока операций*, которые необходимо разделить внутри **Дорожки**. Все **Дорожки** одного `LaneSet` ДОЛЖНЫ указывать разделительный элемент одного и того же типа (к примеру, все **Дорожки** одного `LaneSet` ссылаются на Ресурс в качестве разделительного элемента, однако, каждая **Дорожка** ссылается на отдельный экземпляр Ресурса).

Элемент **Lane** наследует атрибуты и ассоциации элемента `BaseElement` (см. таблицу 8.5). Таблица 10.135 содержит информацию о дополнительных атрибутах и ассоциациях элемента **Lane.**

Таблица 10.135 Атрибуты и ассоциации элемента Lane

Название атрибута	Описание/использование
name: string	Название **Дорожки**.
partitionElement: BaseElement [0..1]	Ссылка на элемент `BaseElement`, указывающая на значение и тип разделения. Посредством данного разделительного элемента в **BPMN** могут указываться элементы `FlowElement`, которые необходимо разделить внутри данной **Дорожки.**
partitionElementRef: BaseElement [0..1]	Ссылка на элемент `BaseElement`, указывающая на значение и тип разделения. Посредством данного разделительного элемента в **BPMN** могут указываться элементы `FlowElement`, которые необходимо разделить внутри данной **Дорожки**.
childLaneSet: LaneSet [0..1]	Ссылка на элемент `LaneSet` для вложенных **Дорожек**.
flowNodeRefs: FlowNode [0..*]	Список элементов `FlowNode`, которые необходимо разделить в **Дорожке** в соответствии со значением элемента

	`partitionElement`, являющегося частью элемента **Lane**.

10.8. Экземпляры Процесса, Немоделируемые Действия и Публичный Процесс

Процесс может выполняться несколько раз, и каждый раз ожидается, что все шаги данного **Процесса** будут выполняться в соответствии с последовательностью их расположения. К примеру, **Процесс** на фигуре 10.1 повторяется каждую пятницу, а в каждом из его экземпляров сначала выполняется **Задача** «Receive Issue List», затем **Задача** «Review Issue List» и т.д., т.е. ожидается последовательность выполнения, отображенная на диаграмме. Также ожидается, что каждый экземпляр этого **Процесса** является верным для данной модели, однако, некоторые экземпляры таковыми не являются (к примеру, если **Процесс** содержит **Действия** для ручного выполнения, а у исполнителей этих **Действий** нет соответствующей инструкции по их выполнению).

В некоторых программах моделирования для выполнения **Процесса** предусмотрено использование большего числа **Действий** и **Событий**, нежели указано в модели данного **Процесса**. Это позволяет осуществлять переход к нужному шагу без внесения изменений в саму модель. Например, изображенные на фигуре 10.1 экземпляры **Процесса** могут выполнять дополнительное **Действие** на отрезке между **Задачами** «Receive Issue List» и «Review Issue List». Данные экземпляры являются верными для этого **Процесса**, поскольку они все так же выполняют содержащиеся в нем **Действия** (в указанном порядке и в указанных условиях).

Существует 2 способа определения того, что в **Процессе** возможно использование немоделируемых **Действий**:

- В случае, если значение атрибута `isClosed` Процесса равно «*false*», или его значение вообще не установлено, то в экземпляре **Процесса** МОГУТ присутствовать такие взаимоотношения, как отправка и получения **Сообщений** и **События**, при этом модель данного **Процесса** не содержит соответствующих элементов. Однако в определенном **Потоке операций** использование немоделируемых взаимоотношений может быть ограничено (см. следующий пункт). В случае, если значение атрибута isClosed Процесса равно «*true*», то экземпляр **Процесса** НЕ МОЖЕТ содержать такие взаимоотношения, как отправка и получения **Сообщений** и **События**, если в данном **Процессе** им не соответствуют дополнительные графические элементы. Это ограничение отменяет выполнение любого немоделируемого взаимоотношения, допустимого для какого-либо из описанных ниже **Потоков операций**.

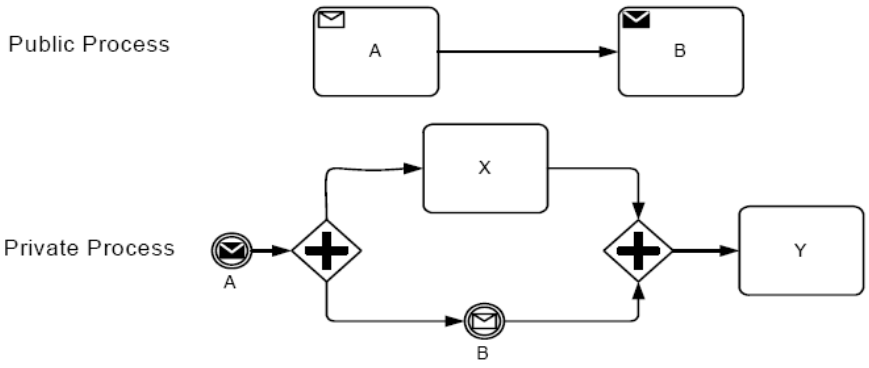
- В случае, если значение атрибута `isImmediate` **Потока операций Процесса** равно «*false*», то в ходе данного **Процесса** МОГУТ БЫТЬ выполнены дополнительные немоделируемые **Действия** и взаимоотношения. Такие **Действия** входят в состав **Потока операций**. В случае, если значение атрибута `isImmediate` равно «*true*», то немоделируемые **Действия** НЕ МОГУТ быть выполнены в ходе следования **Потока операций**. В *невыполняемых* **Процессах** (т.е. тех, чей атрибут `isExecutable` имеет значение «false» в качестве установленного или значения по умолчанию) **Поток операций,** значение атрибута «`isImmediate`» которого не установлено, обрабатывается так, как если бы это значение было равно «false». В *выполняемых* **Процессах** (т.е. тех, чей атрибут `isExecutable` имеет значение «true» в качестве установленного или значения по умолчанию) **Поток операций**, значение атрибута «`isImmediate`» которого не установлено, обрабатывается так, как если бы это значение было равно «*true*». Значение атрибута `isImmediate` *Выполняемого* **Процесса** не может быть «*false*».

Ограничение использования немоделируемых **Действий**, определенное посредством значений атрибутов `isClosed` и `isImmediate`, может быть применено только при выполнении **Процессов** и его экземпляров, содержащих такое ограничение. Немоделируемые **Действия** МОГУТ встречаться в экземплярах других **Процессов**.

В случае, если **Процессом** предусмотрено возможное использование немоделируемых **Действий**, то такие **Действия** могут появляться и в моделях других **Процессов** и их экземплярах, выполнение которых может быть верным для исходного **Процесса**. Рассмотрим пример **Процесса**, схожего с **Процессом** на фигуре 10.1, содержащим дополнительное **Действие** на отрезке между **Задачами** «Receive Issue List» и «Review Issue List». **Процесс** на фигуре 10.1 может содержать атрибут `isClosed` или `isImmediate`, что позволяет **Действиям** появляться на этом отрезке. По мере выполнения **Процесса** на фигуре 10.1 для него становятся верными экземпляры другого **Процесса**, содержащего дополнительные шаги на отрезке между **Задачами** «Receive Issue List» и «Review Issue List»). Разработчики модели могут указать, что все экземпляры одного **Процесса** становятся верными для другого **Процесса**, и эта связь осуществляется посредством поддерживающей ассоциации между **Процессами**. В ходе разработки моделей данных **Процессов** такая ассоциация может не учитываться, поскольку она является лишь отображением намерений разработчиков.

Как правило, такая поддержка осуществляется между *приватным* и *публичным* **Процессами** (см. Общее представление). *Публичный* **Процесс** содержит **Действия**, видимые для внешних объектов (например, *Участников* **Взаимоотношения**), в то время как *приватный* **Процесс** содержит остальные **Действия**, недоступные для внешних объектов. Скрытые **Действия** *приватного* **Процесса** для *публичного* **Процесса** не моделируются. Однако предполагается, что экземпляры *приватного* **Процесса** будут доступны для внешних объектов, как если бы они были экземплярами *публичного* **Процесса**. Это означает, что *приватный* **Процесс** поддерживает *публичный* (ожидается, что все экземпляры *приватного* **Процесса** будут верны и для *публичного*).

Процесс, поддерживающий другой **Процесс** (как в случае с *приватным* и *публичным* **Процессами**), не должен полностью его копировать. НЕОБХОДИМО лишь, чтобы экземпляры этого **Процесса** появлялись так же, как если бы они были экземплярам другого. В верхней части фигуры 10.127 отображен *публичный* **Процесс** с **Задачами Отправка** и **Получение**. Поддерживающий его *приватный* **Процесс** отображен в нижней части фигуры. *Приватный* **Процесс** отправляет и получает те же сообщения, что и *публичный*, однако, для выполнения этих операций вместо **Задач** в нем используются **События**. В нем также содержатся **Действия**, не *входящие* в состав *публичного* **Процесса**. При этом все экземпляры *приватного* **Процесса** будут появляться так, как если бы они были экземплярами *публичного*, т.к. сообщения отсылаются и принимаются в ТРЕБУЕМОМ *публичным* **Процессом** порядке. В *публичном* **Процессе** допускается появление немоделируемых **Действий**.

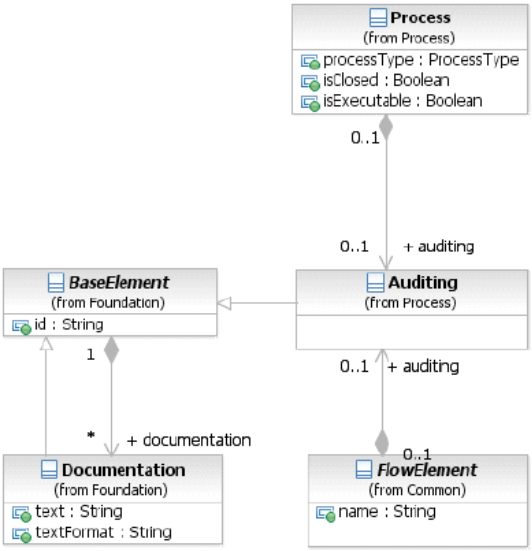


Фигура 10.127 – Процесс, поддерживающий другой Процесс

Фактически, *публичный* **Процесс** выглядит как недостаточно углубленный *приватный*. То, что не указано в *публичном* **Процессе**, есть в *приватном*. К примеру, если ни один из **Потоков операций**, направленных от **Эксклюзивного Шлюза**, не содержит `условного выражения`, то в *приватном* **Процессе** будет указано то, какое из **Действий**, служащих целями **Потоков операций**, появится. Другим примером является **Событие Таймер**, для которого не указано значение элемента `EventDefinition`. То, когда выключится таймер, указывается в *приватном* **Процессе**.

10.9. Аудирование

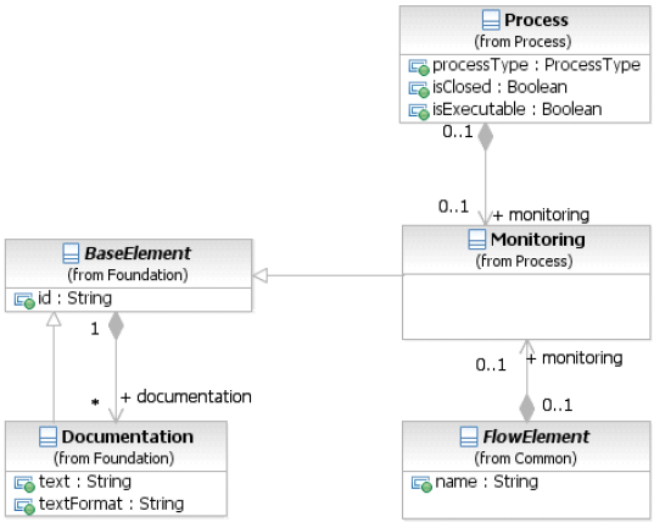
Элемент `Auditing` и его ассоциации позволяют указывать значения атрибутов, связанных с проверкой. Посредством его использования достигается способность **BPMN** к расширению. Данный элемент используется *Элементам потока* и **Процессами**. Однако данный документ не содержит описания атрибутов проверки, т.к. в **BPMN 2.0** указан собственный набор атрибутов и семантики для них.



Фигура 10.128 – Диаграмма классов элемента Auditing

10.10. Мониторинг

Элемент `Monitoring` и его ассоциации позволяют указывать значения атрибутов, связанных с проверкой. Посредством его использования достигается способность BPMN к расширению. Данный элемент используется *Элементам потока* и Процессами. Однако данный документ не содержит описания атрибутов проверки, т.к. в BPMN 2.0 указан собственный набор атрибутов и семантики для них.



Фигура 10.129 – Диаграмма классов элемента Monitoring

10.11. Представление XML-схемы для пакета Процесса

Таблица 10.136 – XML–схема для элемента Process

```
<xsd:element name="process" type="tProcess" substitutionGroup="rootElement"/>
<xsd:complexType name="tProcess">
    <xsd:complexContent>
        <xsd:extension base="tCallableElement">
            <xsd:sequence>
                <xsd:element ref="auditing" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="monitoring" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="processRole" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="laneSet" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="flowElement" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="artifact" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="resourceRole" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="correlationSubcription" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="supports" type="xsd:QName" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="processType" type="tProcessType" default="None"/>
            <xsd:attribute name="isExecutable" type="xsd:boolean"use="optional"/>
            <xsd:attribute name="isClosed" type="xsd:boolean" default="false"/>
            <xsd:attribute name="definitionalCollaborationRef" type="xsd:QName" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="tProcessType">
    <xsd:restriction base="xsd:string">
<xsd:enumeration value="None"/>
        <xsd:enumeration value="Public"/>
        <xsd:enumeration value="Private"/>
    </xsd:restriction>
</xsd:simpleType>
```

Таблица 10.137 – XML–схема для элемента Auditing

```
<xsd:element name="auditing" type="tAuditing"/>
<xsd:complexType name="tAuditing">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement"/>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.138 – XML–схема для элемента GlobalTask

```
<xsd:element name="globalTask" type="tGlobalTask" substitutionGroup="rootElement"/>
<xsd:complexType name="tGlobalTask">
    <xsd:complexContent>
        <xsd:extension base="tCallableElement">
            <xsd:sequence>
                <xsd:element ref="resourceRole" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.139 – XML–схема для элемента Lane

```
<xsd:element name="lane" type="tLane"/>
<xsd:complexType name="tLane">
    <xsd:complexContent>
        <xsd:extension base="tBaseElement">
            <xsd:sequence>
                <xsd:element name="partitionElement" type="tBaseElement" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="flowNodeRef" type="xsd:IDREF" minOccurs="0"
maxOccurs="unbounded"/>
                <xsd:element name="childLaneSet" type="tLaneSet" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
```

```
                <xsd:attribute name="name" type="xsd:string"/>
                <xsd:attribute name="partitionElementRef" type="xsd:QName"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.140 – XML–схема для элемента LaneSet

```
<xsd:element name="laneSet" type="tLaneSet"/>
<xsd:complexType name="tLaneSet">
<xsd:complexContent>
            <xsd:extension base="tBaseElement">
                <xsd:sequence>
                    <xsd:element ref="lane" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.141 – XML–схема для элемента Monitoring

```
<xsd:element name="monitoring" type="tMonitoring"/>
<xsd:complexType name="tMonitoring">
        <xsd:complexContent>
            <xsd:extension base="tBaseElement"/>
        </xsd:complexContent>
</xsd:complexType>
```

Таблица 10.142 – XML–схема для элемента Performer

```
<xsd:element name="performer" type="tPerformer" substitutionGroup="resourceRole"/>
<xsd:complexType name="tPerformer">
        <xsd:complexContent>
            <xsd:extension base="tResourceRole"/>
        </xsd:complexContent>
</xsd:complexType>
```