

Министерство образования и науки Российской Федерации
Московский государственный университет экономики,
статистики и информатики (МЭСИ)

И.Г. Фёдоров

Моделирование бизнес-процессов в нотации BPMN2.0

Научно практическое издание

Москва, 2013

УДК 004
ББК 65.23
Ф 333

Рецензенты: Божко В.П. профессор, д.э.н.
Сотников А.Н. профессор, д.ф-м.н.

Фёдоров И. Г.

Моделирование бизнес-процессов в нотации BPMN2.0:
Монография, Москва 2013 г. МЭСИ. – 255 стр.

В книге рассмотрены вопросы создания исполняемых моделей бизнес-процессов в нотации BPMN 2.0. Показано использование нотации для моделирования процессов протекающих внутри одной организации, а так же процессов межорганизационного взаимодействия, возникающих, например, при реализации электронной коммерции. Рассмотрено моделирование следующих видов диаграмм: схем оркестровки, взаимодействия процессов, диалогов, процессной хореографии. Предлагаемое издание полностью покрывает все типы моделей, реализуемых с помощью нотации BPMN.

Книга предназначена для широкого круга специалистов, занятых моделированием бизнес-процессов. Она окажется особенно полезной тем из них, которые владеют приемами аналитического моделирования бизнес-процесса и хотели бы перейти к разработке исполняемых моделей.

ISBN 978-5-7764-0772-7

© Фёдоров И.Г., 2013
© МЭСИ, 2013

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	17
А. Модель бизнес-процесса	17
Б. Нотация моделирования	17
В. Моделирование бизнес-процессов	17
Г. Почему моделирование в нотации BPMN?	19
Д. Чем хороша нотация BPMN	20
Е. Что изложено в этой книге	21
Ж. Кому рекомендуется эта книга	22
З. Что использовалось при написании этой книги	23
И. Благодарности	24
1. БИЗНЕС-ПРОЦЕССЫ И ПРОЦЕСНОЕ УПРАВЛЕНИЕ	25
1.1. Производительность труда в непроизводственной сфере	26
1.2. Причины низкой производительности	27
1.3. Недопустимо подменять процессное управление автоматизацией	27
1.4. Процесс как конвейер	28
1.5. Бизнес-процесс	28
1.6. Операции и действия	29
1.7. Имя бизнес-процесса	30
1.8. Управление бизнес-процессами	31
1.8.1. Три уровня управления бизнес-процессами	31
1.9. Типы информационных систем	33
1.9.1. Процессно-ориентированные информационные системы	33
1.9.2. Системы управляемые моделью	34
1.10. Системы управления бизнес-процессами	35
1.11. Классификация бизнес-процессов	36
1.11.1. Основные, вспомогательные и обеспечивающие	36
1.11.2. Сквозные процессы	37
1.11.3. Структурированные процессы	38
1.11.4. Внутри и межорганизационные процессы	40
1.11.5. Процессы автоматизированные и автоматические	41
1.12. Процесс и документооборот	41
1.13. Объект управления процесса	42
1.13.1. Маркер потока управления процесса	43
1.13.2. Параллельные потоки управления процесса	44
1.13.3. Многопоточные операции	44
1.14. Модель бизнес-процесса	45

1.14.1.	Процесс описывает работу	46
1.14.2.	Способы декомпозиции процесса	46
1.14.3.	Функциональные и процессные модели	47
1.15.	Исполняемая модель бизнес-процесса	47
1.16.	Экземпляр процесса	50
1.17.	Версия процесса	50
1.17.1.	Версия модели процесса	50
1.17.2.	Версионирование экземпляров процесса	50
1.18.	Стандарты описания бизнес-процессов	51
2.	СПЕЦИФИКАЦИЯ BPMN 2.0	54
2.1.	История развития нотации BPMN	55
2.2.	Область применения нотации BPMN	57
2.3.	Обзор основных элементов нотации	58
2.3.1.	Элементы управления	58
2.3.2.	Соединительные элементы	59
2.3.3.	Элементы данных	60
2.3.4.	Зоны ответственности	60
2.3.5.	Артефакты	61
2.4.	Субклассы нотации BPMN	61
2.4.1.	Пример использования элементов нотации BPMN	63
2.5.	Категории диаграмм бизнес-процессов	64
2.5.1.	Диаграммы оркестровки	65
2.6.	Схемы взаимодействия	70
2.7.	Хореография процессов	71
2.7.1.	Схема диалогов	72
2.8.	схемамы оркестровки, диалогов хореографии	72
3.	ОПЕРАЦИИ	73
3.1.	Виды операций	73
3.1.1.	Интерактивная операция	74
3.1.2.	Ручная операция	74
3.1.3.	Автоматическая операция	75
3.1.4.	Операция сценарий	75
3.1.5.	Операция бизнес-правило	75
3.1.6.	Операция отправка и получение сообщения	76
3.1.7.	Абстрактная операция	76
3.2.	Маркеры операции	77
3.2.1.	Маркер подпроцесса	77

3.2.2.	Маркер цикла	78
3.2.3.	Маркер параллельного выполнения	78
3.2.4.	Маркер последовательного выполнения	80
3.2.5.	Маркер Ad-Hoc операции	80
3.2.6.	Операция компенсация	81
3.2.7.	Комбинации маркеров подпроцесса	82
3.3.	Группа Операций	83
3.4.	Модель Процесса	84
3.5.	Подпроцесс	85
3.5.1.	Вложенный подпроцесс	85
3.5.2.	Повторно используемый подпроцесс	87
3.5.3.	Вызывающая операция	88
3.5.4.	Область действия данных при вызове глобально известного подпроцесса	89
3.5.5.	AD-Hoc Подпроцесс для конкретного случая	90
3.5.6.	Событийный подпроцесс	91
3.5.7.	Транзакционный процесс	94
4.	ПОТОКИ УПРАВЛЕНИЯ.	98
5.	ЛОГИЧЕСКИЕ ОПЕРАТОРЫ	99
5.1.	Логические операторы и бизнес правила	99
5.2.	Типы Логических операторов	100
5.2.1.	Логический оператор «И»	101
5.2.2.	Логический оператор «ИЛИ» управляемый данными (OR)	103
5.2.3.	Логический оператор «Исключающее ИЛИ» управляемый данными (XOR)	105
5.3.	Логический оператор комплексное условие	108
5.4.	Событийный оператор «Исключающее ИЛИ»	109
5.5.	Событийный оператор «Исключающее ИЛИ», (создает новый экземпляр процесса)	111
5.6.	Событийный оператор «И» (создает новый экземпляр процесса)	112
6.	СОБЫТИЯ	113
6.1.	Типы событий	114
6.2.	Классификация событий	115
6.2.1.	Начальные, промежуточные и конечные	115
6.2.2.	Генерирующие и обрабатывающие	115
6.2.3.	Независимые и прикрепленные	116

6.2.4.	Прерывающие и непрерывающие	117
6.3.	События и данные	117
6.4.	Сигнал	118
6.4.1.	Обработка сигналов	119
6.5.	Сообщения	121
6.5.1.	Потоки сообщений	122
6.5.2.	Диалоги	123
6.5.3.	Отправка и получение сообщений	125
6.5.4.	Семантика оправки и получения сообщений	126
6.5.5.	Явная адресация получателя сообщения	127
6.5.6.	Корелляция - неявная адресация получателя сообщения	127
6.6.	Начальные события	129
6.7.	Составное взаимоисключающее стартовое событие	131
6.8.	Составное параллельное стартовое событие	132
6.9.	Способы старта событийного подпроцесса	133
6.10.	Завершающие события	134
6.10.1.	Простое завершающее Событие	135
6.10.2.	Событие-прекращение	135
6.10.3.	Завершающее событие-сообщение	136
6.11.	События, устанавливающие статус завершения подпроцесса	137
6.11.1.	Событие-ошибка	138
6.11.2.	Событие-эскалация	138
6.11.3.	Событие-отмена	139
6.11.4.	Событие-компенсация	140
6.12.	Промежуточные события	140
6.12.1.	Промежуточные события, размещаемые в потоке управления процесса	141
6.12.2.	Простое промежуточное событие	143
6.12.3.	Промежуточное событие для работы с сообщениями	143
6.12.4.	Событие-таймер	144
6.12.5.	Промежуточное событие-условие	144
6.12.6.	Промежуточное Составное событие	145
6.12.7.	Промежуточное Составное параллельное событие	145
6.12.8.	Событие-ссылка	145
6.13.	События, прикрепляемые к границам операций	146
6.14.	Ретрансляция события	147
6.14.1.	Обработка прерывающих и непрерывающих событий	149
6.14.2.	Прерывающее событие-таймер	149
6.14.3.	Непрерывающее событие-таймер	150

7.	ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ	152
7.1.	Классификация исключительных ситуаций	152
7.2.	События для обработки исключительных ситуаций	154
7.3.	Место возникновения исключительных ситуаций	155
7.3.1.	Исключительная ситуация в операции	155
7.3.2.	Исключительная ситуация в процессе	156
7.3.3.	Исключительная ситуация во внешней среде	157
7.4.	Уровни обработки Исключительных ситуаций	158
7.4.1.	Обработка на уровне операции	158
7.4.2.	Обработка во вложенном подпроцессе	159
7.4.3.	Обработка в вызывающем процессе	159
7.5.	Действия после обработки исключительной ситуации	161
7.5.1.	Без возврата в точку прерывания;	161
7.5.2.	С возвратом в точку прерывания	161
7.6.	Влияние исключения на текущий процесс	161
7.6.1.	Исключения, прерывающие исполнение процесса	162
7.6.2.	Исключения, не прерывающие исполнение процесса	162
7.7.	Событие-эскалация	163
7.7.1.	Обработка эскалации	164
7.7.2.	Обработка непрерывающего события-эскалация	165
7.8.	Обработка компенсации	166
8.	ОБЪЕКТЫ ДАННЫХ	170
8.1.1.	Жизненный цикл и доступность объектов данных	172
8.1.2.	Способы отображения ассоциации на схеме процесса	173
8.1.3.	Статус обработки документа	174
8.1.4.	Коллекция объектов данных	174
8.1.5.	Хранилище данных	175
8.1.6.	Внешний вход и выход процесса	176
8.1.7.	Семантика исполнения ассоциации данных	177
8.1.8.	Полезная информационная нагрузка сигналов и оповещений	177
8.1.9.	Потоки данных и управления	178
8.2.	Замечание о работе с данными:	179
9.	ЗОНЫ ОТВЕТСТВЕННОСТИ (ДОРОЖКИ И ПУЛЫ)	181
9.1.	Ролевая модель	181
9.1.1.	Как проблема решается в BPM	183
9.2.	Пул и дорожка	183
9.2.1.	Пул	184

9.2.2.	Дорожка	184
9.3.	Отбор исполнителей	186
9.4.	Спецификация WS-Human Task	187
9.5.	Назначение исполнителей в WS-Human Task	189
10.	ПРОЦЕССНЫЕ ПАТТЕРНЫ	193
10.1.	Базовые процессные паттерны	193
10.1.1.	Последовательное выполнение (CP1)	194
10.1.2.	Параллельное выполнение (CP2)	194
10.1.3.	Синхронизация потоков (CP3)	195
10.1.4.	Альтернатива (CP4)	195
10.1.5.	Простое слияние (CP5)	196
10.1.6.	Множественный выбор (CP6)	198
10.2.	Паттерны слияния и синхронизации	199
10.2.1.	Структурированные паттерны	199
10.2.2.	Структурированное слияние с синхронизацией (CP7)	199
10.2.3.	Неструктурированные паттерны	200
10.2.4.	Множественное слияние (CP8)	202
10.2.5.	Дискриминатор (CP9)	202
10.3.	Итерации	205
10.3.1.	Многократное повторение (CP10)	205
10.3.2.	Итерация (CP21)	205
10.3.3.	Рекурсивное выполнение (CP22)	206
10.4.	Параллельное выполнение	206
10.4.1.	Параллельное выполнение без синхронизации (CP12)	207
10.4.2.	Параллельное выполнение с синхронизацией, число экземпляров известно на этапе моделирования (CP13)	208
10.4.3.	Параллельное выполнение, число экземпляров известно на этапе исполнения (CP14)	209
10.5.	Статус выполнения	209
10.5.1.	Отложенный выбор (CP16)	209
10.5.2.	Чередование маршрутов (CP17)	210
10.5.3.	координацию выполнения (CP18)	211
10.6.	Паттерны завершения	212
10.6.1.	Явное завершение (CP11)	212
10.6.2.	Прекратить выполнение задания (CP19)	213
10.6.3.	Прекратить выполнение процесса (CP20)	214
10.6.4.	Прекратить выполнение группы операций (CP25)	214
10.6.5.	Прекратить выполнение многопоточной операции (CP26)	215

10.7.	Синхронизация с помощью событий	216
10.7.1.	Синхронизация без запоминания оповещения (CP23)	216
10.7.2.	Синхронизация с запоминанием оповещения (CP24)	217
11.	ДИАГРАММЫ ВЗАИМОДЕЙСТВИЯ	218
11.1.	Уровни взаимодействия	218
11.2.	Множественные участники взаимодействия	220
11.3.	Паттерны межорганизационного взаимодействия	221
11.3.1.	Отправка сообщения	222
11.3.2.	Получение сообщения	223
11.3.3.	Отправка/получение сообщения	223
11.3.4.	Очередь входящих сообщений	224
11.3.5.	Веерная рассылка сообщений	225
11.3.6.	Список полученных сообщений	226
11.3.7.	Рассылка сообщений и обработка полученных результатов	227
11.3.8.	Множественный ответ	227
11.3.9.	Негарантированный ответ на запрос	228
11.3.10.	Широковещательная рассылка	229
11.3.11.	Запрос с перенаправлением	229
12.	СХЕМЫ ДИАЛОГОВ	231
12.1.	Мульти-пулы	233
12.2.	Комплексные диалоги	234
13.	СХЕМЫ ХОРЕОГРАФИИ	236
13.1.	Графические элементы хореографии	236
13.2.	Использование событий в хореографии	241
13.2.1.	Начальные события	241
13.2.2.	Завершающие события	241
13.3.	Промежуточные события	242
13.4.	Использование логических операторов в хореографии	244
14.	ЗАКЛЮЧЕНИЕ	250
ПЕРЕКРЕСТНЫЕ ССЫЛКИ		251
СПИСОК ЛИТЕРАТУРЫ:		255

СПИСОК ИЛЛЮСТРАЦИЙ:

Рисунок 2-1. История развития нотации BPMN	56
Рисунок 2-2. Основные графические элементы нотации BPMN	58
Рисунок 2-3. Субклассы нотации BPMN	62
Рисунок 2-4. Основной набор элементов нотации	63
Рисунок 2-5. Основной набор элементов	64
Рисунок 2-6. Закрытый процесс	65
Рисунок 2-8. Открытый процесс (раскрытый пул)	66
Рисунок 2-9. Пример аналитической схемы процесса	66
Рисунок 2-10. Пример исполняемой схемы процесса	67
Рисунок 2-11. Публичное взаимодействие, закрытые процессы	67
Рисунок 2-12. Публичное взаимодействие, один открытый процесс	68
Рисунок 2-13. Публичное взаимодействие, открытые процессы	68
Рисунок 2-14. Эквивалентные публичный (а) и приватный (б) процессы	70
Рисунок 2-15. Моделирование обмена сообщениями между участниками	71
Рисунок 2-16. Схема хореографии.	71
Рисунок 2-17. Схема диалога участников процесса	72
Рисунок 2-18. Соотношение между схемами оркестровки, диалогов и хореографии	72
Рисунок 3-1. Типы операций в нотации BPMN 2.0	73
Рисунок 3-2. Операции отправки и получение сообщения	76
Рисунок 3-3. Ad-hoc операции	81
Рисунок 3-4. Операция компенсация	82
Рисунок 3-5. Группа задач	83
Рисунок 3-6. Развернутый вложенный подпроцессы	86
Рисунок 3-7. Область действия переменных вложенного процесса	86
Рисунок 3-8. Подпроцесс с параллельно исполняемыми заданиями	87
Рисунок 3-9. Повторно используемый подпроцесс	90
Рисунок 3-10. Подпроцесс «двойного» назначения	88
Рисунок 3-11. Глобальная пользовательская операция	89
Рисунок 3-12. Вызывающая операция и глобальный процесс.	89
Рисунок 3-13. Свернутый и развернутый Ad-Hoc подпроцессы	90
Рисунок 3-14. Подпроцесс «Ad-Hoc» с зависимыми операциями	91
Рисунок 3-15. Развернутый и свернутый событийный подпроцессы	92
Рисунок 3-16. Событийный подпроцесс	93
Рисунок 3-17. Область действия переменных событийного подпроцесса	94
Рисунок 3-18. Свернутый транзакционный подпроцесс	95
Рисунок 3-19. Обработка транзакции	97

Рисунок 5-1. Логический оператор ветвления «И»	102
Рисунок 5-2. Логический оператор слияния «И»	102
Рисунок 5-3. Парные логические операторы "И"	103
Рисунок 5-4. Оператор ветвление «ИЛИ» управляемый данными	104
Рисунок 5-5. Оператор слияние «ИЛИ» управляемый данными	104
Рисунок 5-6. Парные логические операторы "ИЛИ".	105
Рисунок 5-7. Ветвление «Исключающее ИЛИ» управляемое данными	106
Рисунок 5-8. Слияние «Исключающее ИЛИ»	107
Рисунок 5-9. Парные операторы «Исключающее ИЛИ»	107
Рисунок 5-10. Оператор ветвления комплексное условие	108
Рисунок 5-11. Оператор слияния комплексное условие	109
Рисунок 5-12. Событийный оператор «исключающее ИЛИ»	110
Рисунок 5-13. Событийный оператор «исключающее ИЛИ», (создает новый экземпляр процесса)	111
Рисунок 5-14. Событийный оператор «И» (создает экземпляр процесса)	112
Рисунок 6-1. Классификация событий по местоположению в процессе	115
Рисунок 6-2. Классификация событий по типу обработки	116
Рисунок 6-3. Независимые и прикрепленные события	116
Рисунок 6-4. Классификация событий по влиянию на выполнение	117
Рисунок 6-5. Ассоциация данных в событиях	118
Рисунок 6-6. Пакетная обработка с использованием сигналов	121
Рисунок 6-7. Потоки сообщений	122
Рисунок 6-8. Инициирующее диалог и ответное сообщения	123
Рисунок 6-9. Диалог участников	124
Рисунок 6-10. Потоки сообщений на диаграммах: Оркестровки, Хореографии, Взаимодействия	124
Рисунок 6-11. Обмен сообщениями между двумя участниками	126
Рисунок 6-12. Явная адресация получателя сообщения	127
Рисунок 6-13. Подпроцесс без стартового события.	130
Рисунок 6-14. Процесс с несколькими точками старта	131
Рисунок 6-15. Процесс «двойного назначения»	131
Рисунок 6-16. Пример взаимоисключающего старта	132
Рисунок 6-17. Составное стартовое событие	132
Рисунок 6-18. Событийный подпроцесс для события эскалация	133
Рисунок 6-19. Простое завершающее событие	135
Рисунок 6-20. Событие-прекращение	136
Рисунок 6-21. Завершающее событие-сообщение	137
Рисунок 6-22. Завершающее событие-ошибка	138
Рисунок 6-23. Завершающее событие-эскалация	138

Рисунок 6-24. Пример завершение процесса	139
Рисунок 6-25. Завершающее событие-компенсация	140
Рисунок 6-26. Простое промежуточное событие, размещаемое в потоке	143
Рисунок 6-27. События-сообщения, размещенные в потоке	143
Рисунок 6-28. Промежуточное событие-таймер	144
Рисунок 6-29. Промежуточное событие-условие	144
Рисунок 6-30. Промежуточное событие-условие	145
Рисунок 6-31. Промежуточное составное параллельное событие	145
Рисунок 6-32. Событие-ссылка.	146
Рисунок 6-33. Ретрансляция ошибки из операции	148
Рисунок 6-34. Ретрансляция события	148
Рисунок 6-35. Обработка прерывающего прикрепленного события	150
Рисунок 6-36. Обработка непрерывающего прикрепленного события-таймер	150
Рисунок 6-37. Непрерывающее прикрепленное событие-таймер	151
Рисунок 7-1. Обработка события прикрепленного к границе операции	156
Рисунок 7-2. Простое бизнес исключение	157
Рисунок 7-3. Исключительная ситуация во внешней среде	158
Рисунок 7-4. Обработка бизнес исключения во вложенном подпроцессе	159
Рисунок 7-5. Обработка события прикрепленного к границе процесса.	160
Рисунок 7-6. Обработка ошибки без возврата в основной процесс	161
Рисунок 7-7. Обработка ошибки с возвратом в основной процесс	161
Рисунок 7-8. Исключение, прерывающее исполнение процесса	162
Рисунок 7-9. Исключение, непрерывающее исполнение процесса	163
Рисунок 7-10. Обработка прерывающего события-эскалация	165
Рисунок 7-11. Обработки непрерывающего события-эскалация	165
Рисунок 7-12. Обработки непрерывающего события-эскалация	166
Рисунок 7-13. Обработки события-компенсация	167
Рисунок 7-14. Откат с использованием событийного подпроцесса	168
Рисунок 7-15. Откат, инициируемый событием-компенсация	168
Рисунок 7-16. Пример использования отмены и компенсации	169
Рисунок 8-1. Вложенные подпроцессы	172
Рисунок 8-2. Зоны видимости объекта данных	172
Рисунок 8-3. Передача данных в повторно используемый подпроцесс	173
Рисунок 8-4. Направленная ассоциация данных	173
Рисунок 8-5. Ненаправленная ассоциация данных	174
Рисунок 8-6. Статус обработки информационного объекта	174
Рисунок 8-7. Коллекция объектов данных	175
Рисунок 8-8. Хранилище данных	176
Рисунок 8-9. Входные и выходные данные процесса	176

Рисунок 8-10. Чтение информации из внешнего хранилища	176
Рисунок 9-1, Иерархия дорожек процесса	186
Рисунок 9-2 Основные стадии выполнения операции	192
Рисунок 10-1, Последовательное исполнение	194
Рисунок 10-2. Параллельное исполнение	195
Рисунок 10-3, Синхронизация потоков	195
Рисунок 10-4, Выбор взаимоисключающих возможностей	196
Рисунок 10-5. Простое слияние	197
Рисунок 10-6, Множественный выбор	198
Рисунок 10-7. Парные операторы альтернативный выбор	200
Рисунок 10-8. Структурированное слияние с синхронизацией	200
Рисунок 10-9 Неструктурированный паттерн	201
Рисунок 10-10 Структурированный паттерн	201
Рисунок 10-11, Непарные узлы ветвления и слияния	202
Рисунок 10-12, Дискриминатор	203
Рисунок 10-13. Дискриминатор, счетчик	204
Рисунок 10-14. Многократное повторение	205
Рисунок 10-15. Итерация	206
Рисунок 10-16. Параллельное исполнение без синхронизации	208
Рисунок 10-17. Параллельное исполнение с синхронизацией (число экземпляров известно на этапе моделирования)	208
Рисунок 10-18. Параллельное исполнение с синхронизацией (число экземпляров известно на этапе исполнения)	209
Рисунок 10-19, Отложенный выбор	210
Рисунок 10-20. Чередование маршрутов	211
Рисунок 10-21. Координация исполнения	212
Рисунок 10-22. Явное завершение	213
Рисунок 10-23. Явное завершение, не детерминированная семантика исполнения	213
Рисунок 10-24. Прекратить исполнение задания	214
Рисунок 10-25. Прекратить исполнение процесса	214
Рисунок 10-26. Прекратить исполнение группы задач	215
Рисунок 10-27. Прекратить исполнение многопоточной задачи	216
Рисунок 10-28. Синхронизация без запоминания оповещения	217
Рисунок 11-1. Межорганизационное взаимодействие, закрытый процесс	219
Рисунок 11-2. Межорганизационное взаимодействие, открытый процесс	219
Рисунок 11-3. Детализация межорганизационного взаимодействия	220
Рисунок 11-4. Мульти-пул	221
Рисунок 11-5. Паттерн отправка сообщения	223

Рисунок 11-6. Паттерн получение сообщения	223
Рисунок 11-7. Паттерн отправка и получение сообщения	224
Рисунок 11-8. Очередь входящих сообщений	225
Рисунок 11-9. Веерная рассылка сообщений	226
Рисунок 11-10. Список полученных сообщений	226
Рисунок 11-11. Рассылка сообщений и обработка результатов	227
Рисунок 11-12. Негарантированный ответ на запрос	228
Рисунок 11-13. Широковещательная рассылка	229
Рисунок 11-14. Запрос с перенаправлением	230
Рисунок 12-1. Схемы взаимодействия (а) и диалога (б)	233
Рисунок 12-2. Диалог участников, мульти-пул	233
Рисунок 12-3. Глобальный диалог	234
Рисунок 12-4. Комплексный диало	234
Рисунок 12-5. Декомпозиция комплексного диалога	235
Рисунок 12-6. Комплексная диаграмма взаимодействия	235
Рисунок 13-1. Обмен сообщениями, оркестровка, хореография	237
Рисунок 13-2. Однонаправленная задача хореографии	237
Рисунок 13-3. Задача хореографии с двумя адресатами	238
Рисунок 13-4. Веерная рассылка запроса многим участникам	238
Рисунок 13-5. Циклическая рассылка запроса	239
Рисунок 13-6. Диаграмму взаимодействия, несколько пар диалогов	239
Рисунок 13-7. Комплексная задача хореографии	240
Рисунок 13-8. Глобальная задача хореографии	240
Рисунок 13-9. Пример ошибочной схемы хореографии	244
Рисунок 13-10. Использование логических операторов на схеме хорео-графии	246
Рисунок 13-11. Использование логических операторов на схеме хореографии	248
Рисунок 13-12. Использование логического оператора «И» на схеме хореографии	249

СПИСОК ТАБЛИЦ

Таблица 1-1. Классификация процессов по степени формализации	39
Таблица 3-1. Виды операций	74
Таблица 3-2. Маркеры операций	77
Таблица 3-3. Атрибуты циклического исполнения операции	78
Таблица 3-4. Семантика параллельного исполнения	79
Таблица 3-5. Комбинации маркеров подпроцесса	83

Таблица 4-1. Потоки управления	98
Таблица 5-1. Виды логических операторов	101
Таблица 6-1. Моделирование событий в нотации BPMN	114
Таблица 6-2. Графические элементы для работы с сигналами	119
Таблица 6-3. Операция и событие для отправки и получения сообщения	125
Таблица 6-4. Обзор начальных событий	129
Таблица 6-5. Обзор завершающих событий	134
Таблица 6-6. Обработчики событий, размещаемые в потоке управления	142
Таблица 6-7. События, прикрепляемые к операции	147
Таблица 8-1 Графические элементы для изображения данных	170
Таблица 11-1. Число участников взаимодействия	221
Таблица 12-1. Элементы схемы диалогов.	232
Таблица 13-1. Обзор начальных событий	241
Таблица 13-2. Обзор завершающих событий	242
Таблица 13-3. Обработчики промежуточных событий.	242
Таблица 13-4. Обработчики событий, прикрепляемые к задаче хореографии	243

ВВЕДЕНИЕ

А. МОДЕЛЬ БИЗНЕС-ПРОЦЕССА

Моделью принято называть некоторый материальный или мысленно представляемый объект или явление, являющийся упрощённой версией моделируемого объекта или явления (прототипа) и в достаточной степени повторяющий свойства, существенные для целей конкретного моделирования (опуская несущественные свойства, в которых он может отличаться от прототипа). Набор свойств модели определяется целями моделирования.

Модель бизнес-процесса есть описание порядка выполнения работ, приводящих к достижению вполне определенного и воспроизводимого результата. Под бизнес моделью принято понимать формализованное (графическое, табличное, текстовое, символьное) описание бизнес-процессов. Модель должна давать полное, точное и адекватное описание системы.

Б. НОТАЦИЯ МОДЕЛИРОВАНИЯ

Нотация моделирования (далее нотация) – совокупность графических элементов, которые используются для разработки моделей деятельности компании. Нотация – это синтаксис графического языка моделирования. Разрабатываемая с помощью соответствующей нотации модель бизнес-процесса должна реализовывать поведение, которое пользователь ожидает от соответствующей бизнес системы. Поведение – это семантика языка моделирования.

В. МОДЕЛИРОВАНИЕ БИЗНЕС-ПРОЦЕССОВ

Различают аналитическое моделирование бизнес-процессов и разработку исполняемых моделей бизнес-процесса. В реинжиниринге описание бизнес-процессов проводится с целью дальнейшей реорганизации. Для этого необходимо понять, как ис-

полняется процесс, поэтому достаточно ограничиться его укрупненным описанием. Чем детальнее описание, тем сложнее оно становится для восприятия и понимания, поэтому аналитики предпочитают опускать детали, зачастую достаточно важные. Обычно аналитические модели ограничиваются показом только самых вероятных сценариев исполнения, не содержат редко применяемых маршрутов обработки.

Диаграммы, описывающие бизнес-логику, визуально кажутся простыми и понятными, поскольку не включают бизнес-правила, графики исполнения, действия выполняемые, когда показатели процесса выходят за рамки допустимых диапазонов. Поэтому многие аналитики используют их для согласования с представителями бизнеса. Однако простота обманчива, разработчикам ИТ систем приходится повторно собирать пропущенные сведения, причем их представление о процессе может существенно отличаться от взглядов аналитика. Возникает опасная ситуация, модель не в полной мере описывает процесс, детали не фиксируются явно, а существует в головах программистов, что является одной из причин, почему модель процесса на бумаге не соответствует логике работы ИТ системы.

Моделирование часто рассматривается только как средство документирования процессов. Часто, обнаружив несоответствие модели процесса требованиям, предъявляемым к информационной системе, разработчики вносят соответствующие изменения прямо в программный код и не модифицируют соответствующие модели процессов. Вследствие чего модели быстро теряют свою актуальность. Может возникнуть вопрос, требуется ли нести высокие затраты на однократное моделирование бизнес-процессов предприятия, если полученные модели так быстро теряют свою актуальность.

Напротив, исполняемые модели должны показывать все возможные маршруты исполнения процесса, иначе работа соответствующей системы окажется невозможной. Управление бизнес-процессами порождает потребность глубокого описания всех мельчайших деталей исполнения, без которых последующее исполнение окажется невозможным. Если в ходе тес-

тирования выясняется, что модель реализует поведение, которое не в полной мере соответствует ожидаемого пользователем, то соответствующие изменения вносятся в саму модель. Как следствие, последняя не теряет своей актуальности. Выполняя разработку исполняемых моделей бизнес-процесса, предприятие осуществляет долговременные инвестиции в повышение эффективности своего бизнеса.

Г. ПОЧЕМУ МОДЕЛИРОВАНИЕ В НОТАЦИИ BPMN?

Для моделирования бизнес-процессов используются различные нотации. Например, широкое распространение получила нотация EPC фирмы Software AG, различные типы диаграмм потоков работ. Зачастую они являются проприетарными, используются только в составе программных средств этой фирмы. Все они находят широкое применение для построения аналитических моделей, но не предназначены для разработки исполняемых моделей. Эти нотации обладают семантикой, которая описывается в руководствах по моделированию в соответствующей нотации. Поскольку они не предназначены для разработки исполняемых моделей, у них отсутствует спецификация семантики исполнения.

Для создания исполняемых процессных моделей все чаще используется нотация BPMN. Она стала стандартом для лидеров ИТ. рынка и широко используется в программных системах, разработанных такими лидерами ИТ. рынка как IBM, ORACLE, SOFTWARE AG и др. Поддержка лидеров ИТ. отрасли обеспечивает этой нотации широкую популярность у бизнес-аналитиков и разработчиков ИТ. систем

Основная цель, которую ставили разработчики спецификации BPMN – создание стандартной нотации понятной широкому кругу бизнес пользователей: бизнес-аналитикам, создающим и улучшающим процессы компании, техническим разработчикам, ответственным за реализацию процессов, менеджерам, следящим за работой предприятия и управляющими им. Спецификация, BPMN призвана служить связующим зве-

ном между бизнес пользователями, которые в понятной форме могут специфицировать свои потребности, и ИТ разработчиками, которые реализуют поставленные требования, в разрабатываемых информационных системах.

Д. ЧЕМ ХОРОША НОТАЦИЯ BPMN

Нотация BPMN позволяет начать с разработки высокоуровневой аналитической модели, которая дает общее представление о характере исполнения бизнес-процесса. По мере роста понимания того, как должен исполняться бизнес-процесс, модель расширяется, уточняется и углубляется. Результатом моделирования становится исполняемая модель бизнес-процесса.

Преимущество нотации BPMN заключается в том, что она позволяет описать поведение ИТ системы с минимальным программированием. Таким образом, большую часть работ по созданию исполняемой модели может выполнить бизнес-аналитик, без участия программистов и разработчиков.

Применение нотации BPMN обеспечивает бизнес пользователям ряд преимуществ. В первую очередь, это уменьшение разрыва между моделями «Как-Есть» и «Как-Должно-Быть». Исполняемая модель помогает не только раскрыть и верифицировать модель бизнес-процесса, но и испытать ее в условиях реальной эксплуатации. Такие тесты позволяют эффективно выявлять и расширять узкие места процесса, найти более эффективные способы обработки информации.

На практике это означает, что работа с процессом всегда начинается с модели «Как-Есть», а переход к модели «Как-Должно-Быть» осуществляется путем проведения небольших эволюционных изменений, причем при постоянном контроле за изменением метрик процесса. Т.о. можно говорить, что полученные модели «Как-Должно-Быть» носят объективный и достоверный характер, поскольку базируются на результатах измерений, а не на субъективном представлении консультанта.

Интересно отметить, если аналитик и разработчик в паре работают над единой моделью, они при этом видят разный набор слоев модели. Благодаря работе с единой моделью упрощается обмен информацией, что способствует более быстрому и качественному решению задач.

щается взаимодействие между участниками команды, что выгодно отличает предлагаемый подход от традиционного, где разные категории разработчиков использует различные модели, что приводит к непониманию и ошибкам. Бизнес пользователи видят эту же модель. Благодаря этому удается сблизить представления бизнес пользователей и разработчиков о модели процесса.

Простота моделирования позволяет бизнес-аналитику с минимальной помощью разработчика не только создать работающий прототип, но и протестировать его работу, на самом раннем этапе выявить степень соответствия модели реальному бизнес-процессу и таким образом сделать процедуру верификации бизнес-процесса более объективной.

Е. ЧТО ИЗЛОЖЕНО В ЭТОЙ КНИГЕ

Стандарт BPMN является доступным в интернет документом. Он детально описывает особенности реализации нотации. Однако он предназначен в первую очередь для разработчиков программных средств BPMS и в меньшей степени предназначен для бизнес-аналитиков. Это вызывает необходимость создания справочных пособий и методических материалов, предназначенных для бизнес-аналитиков и раскрывающих особенности моделирования в нотации BPMN.

К сожалению, на отечественном рынке отсутствуют русскоязычные издания, которые бы систематично описывали саму нотацию, давали бы рекомендации по ее применению. Предлагаемая читателю книга предназначена, что бы заполнить этот пробел. В ней сделана попытка системно изложить особенности нотации и правила моделирования бизнес процессов в нотации BPMN, описать синтаксис моделирования и семантику исполнения модели..

Книга структурирована следующим образом. Во второй главе дается объяснение основных терминов процессного управления. Описаны основные понятия и термины бизнес моделирования. Третья глава дает общее описание основных возможностей нотации и типов моделей бизнес-процессов. Она

знакомит с группами элементов моделирования и типами моделей, реализуемых в нотации BPMN. Четвертая глава посвящена операциям бизнес-процесса и особенностям их моделирования. Операция есть базовый элемент нотации, правильное использование соответствующих графических элементов является залогом правильного моделирования. Пятая глава описывает потоки управления модели бизнес-процесса. Шестая глава посвящена логическим операторам, которые позволяют ввести в модель бизнес-процесса элементы управления. В этой же главе дается рекомендация по использованию логических операторов и бизнес правил. Седьмая глава описывает события, которые позволяют описать реакцию системы на изменения внешнего по отношению к процессу бизнес окружения. Восьмая глава описывает практику применения событий для моделирования реакции системы на возникающие ошибки и исключительные ситуации. Девятая глава посвящена связи модели процесса с данными, обрабатываемыми в ходе исполнения процесса. Десятая глава посвящена моделированию человеческих ресурсов участников бизнес-процесса. Одиннадцатая глава посвящена обзору процессных паттернов. Семантика поведения группы логических операций может существенно отличаться от семантики каждого из операторов, рассматриваемых по-отдельности. Паттерны являются предметом широкого обсуждения на конференциях, посвященных системам управления бизнес-процессами. Последующие главы описывают процессы межорганизационного взаимодействия, возникающего, например, при реализации электронной коммерции. Двенадцатая глава показывает моделирование диаграмм взаимодействия процессов. Глава тринадцатая описывает моделирование схем диалога. Глава четырнадцатая посвящена моделированию схем процессной хореографии. Предлагаемое издание полностью покрывает все типы моделей, реализуемых с помощью нотации BPMN.

Ж. КОМУ РЕКОМЕНДУЕТСЯ ЭТА КНИГА

Эта книга может быть рекомендована широкому кругу спе-

циалистов, занятых моделированием бизнес-процессов. Она окажется особенно полезной тем из них, которые владеют приемами аналитического моделирования бизнес-процесса и хотели бы перейти к разработке исполняемых моделей.

В первую очередь, материал окажется полезен бизнес-аналитикам и специалистам технологиям бизнеса. В большинстве организаций, например, финансового сектора существуют департаменты банковских, страховых и финансовых технологий. Не меньший интерес она может представлять для специалистов клиентского обслуживания, например, урегулирования убытков, управления рисками. Наконец, специалисты информационных технологий смогут ознакомиться с новыми принципами модель-ориентированной разработки процессных информационных систем. Издание окажется полезным аспирантам и студентам старших курсов, обучающимся по специальностям бизнес-аналитика и ИКТ.

3. ЧТО ИСПОЛЬЗОВАЛОСЬ ПРИ НАПИСАНИИ ЭТОЙ КНИГИ

При написании этой книги были использованы следующие материалы. Во-первых, был тщательно проработан и проанализирован стандарт BPMN 2.0. Спецификация дает сухое описание синтаксиса и семантики моделей, поэтому для улучшения понимания конструкций моделирования было необходимо привести достаточное число примеров и иллюстраций.

Разработчики стандарта выпустили несколько книг на английском языке. Хочется выделить «BPMN Method & Style», в которой Б. Сильвер делится своим опытом моделирования бизнес-процессов, а также «BPMN Modeling and Reference Guide» авторы которой С. Уайт и Д. Майерс дают подробный обзор возможностей нотации.

Автор внимательно следит за публикациями в интернете, посвященным вопросам моделирования BPMN. Хочется выделить англоязычный блог «WPMS Watch», где его ведущий Брюс Сильвер дает рекомендации по использованию моделей BPMN. Так же заслуживают внимания блоги «Главное процесс» и

«Изучаем BPMN», их автор Анатолий Белайчук разбирает многочисленные примеры использования BPMN для бизнес-моделирования. Анализ обоих блогов оказался очень полезен при подготовке практических примеров, использованных в книге.

И. БЛАГОДАРНОСТИ

В написании этой книги автору помогали аспиранты и студенты МЭСИ, которые практически проверяли примеры, приведенные в этой книге, оказали неоценимую помощь при подготовке рукописи и ее оформлении. Главы 7 и 10 были написаны совместно с К.С. Курышевым. В оформлении рукописи приняли активное участие Н.А. Морозов, Е.А. Рыбак и другие.

Автор благодарит сотрудников факультета ИКТ и кафедры ПИЭ МЭСИ за ценные обсуждения и советы, которые помогли довести этот труд до конца. Особая благодарность заведующему кафедрой ПИЭ профессору д.э.н. Ю.Ф. Тельнову за содействие в подготовке материала и помощь в организации данной публикации.

1. БИЗНЕС-ПРОЦЕССЫ И ПРОЦЕСНОЕ УПРАВЛЕНИЕ

Гуру процессного управления М.Хаммер и Д.Чампи заметили: не продукты, а эффективные процессы их создания и развития приносят компаниям долгосрочный и устойчивый успех. Большинство производственных компаний уделяют пристальное внимание своим технологическим процессам и повышению их эффективности. Напротив, непроизводственный сектор не слишком решительно движется по пути процессного управления, не занимается регулярно своими бизнес-процессами. Компании постоянно ищут пути сокращения расходов и, оперируя экономическими критериями, оптимизируют свою организационную структуру, добавляя или сокращая персонал, при этом, они не замечают, что переход на процессное управление позволит не только добиться повышения эффективности и сокращения расходов на персонал, но и проведет к повышению качества обслуживания клиентов, сокращению рисков. А ведь именно процессное управление оценивается как основной резерв повышения эффективности.

В чем причина низкой эффективности и качества обслуживания клиентов? Даже сфере банковских услуг характерно отсутствие жесткой регламентации действий персонала особенно фронт-офисных операций. В противоположность реальному производству, где менеджмент зорко следит за соблюдением требований производственного процесса и строго карает любые отклонения от технологии, в непроизводственной сфере считается допустимым только самое общее описание действий персонала, что приводит к большой вариативности операций. Сотрудники склонны модифицировать свои действия в соответствии с личным опытом, знаниями и квалификацией, в результате, снижаются эффективность работы и качество обслуживания клиентов.

Вариативность действий может иметь ряд негативных последствий: возрастают убытки, связанные с неудовлетворенно-

стю клиента уровнем обслуживания, уменьшается возможность контроля со стороны менеджмента, что в свою очередь является питательной средой для повышения банковских рисков. Если со снижением эффективности процессов банки еще могут мириться, то увеличения рисков банки категорически допустить не могут. Поэтому уменьшение вариативности действий персонала есть первоочередная задача любой организации, стремящейся к повышению своей эффективности.

1.1. ПРОИЗВОДИТЕЛЬНОСТЬ ТРУДА В НЕПРОИЗВОДСТВЕННОЙ СФЕРЕ

В проведенном компанией McKinsey Global Institute исследовании «Эффективная Россия: Производительность труда как фундамент роста»¹ отмечается, что производительность труда во всех отраслях, включая финансовый сектор, остается недопустимо низкой. В России низким уровнем производительности отличаются все виды услуг – от кредитования (4% от уровня США) до платежных транзакций (13% от уровня США). Даже в лучших российских банках самые простые операции занимают в 2 – 6 раз больше времени: снятие денег со счета – больше в 6 раз, пополнение счета – в 2,2 раза, а платеж по счету – в 3,8 раза. Количество сотрудников, приходящихся на одну банковскую операцию выше в 2-2,5 раза, а среднее число бумажных документов в 1,5 - 3 раза больше. И это несмотря на то, что финансовый сектор является одним из самых автоматизированных и обеспеченных ресурсами.

Можно сделать вывод - ошибочно думать, что высокая степень автоматизации является гарантией высокой производительности. Последняя есть результат правильной организации труда.

¹ Отчет: «Эффективная Россия: Производительность как фундамент роста», McKinsey Global Institute, 2009.

1.2. ПРИЧИНЫ НИЗКОЙ ПРОИЗВОДИТЕЛЬНОСТИ

Как отмечено в исследовании, всем исследуемым сферам, включая сектор финансовых услуг, характерно отсутствие жесткой регламентации действий персонала, что приводит к большой вариативности процессов и, как следствие, снижению эффективности и качества обслуживания клиентов.

Сквозная технология, связывающая несколько подразделений в единое целое, остается вне фокуса внимания менеджмента. При этом, как показывает практика, нескоординированная деятельность высококлассных специалистов, оказывается менее эффективной, чем хорошо организованный труд сотрудников обычной квалификации.

1.3. НЕДОПУСТИМО ПОДМЕНЯТЬ ПРОЦЕССНОЕ УПРАВЛЕНИЕ АВТОМАТИЗАЦИЕЙ

Большинство методологий моделирования бизнес-процессов, используемых при реинжиниринге, относятся к классу функциональных, отвечают на вопрос «что делает компания», но не раскрывают деталей о том «как она это делает». Сложно представить себе руководителя, который бы не знал, что делает его компания. Получается, модель рассказывает то, что уже хорошо известно, но умалчивает о действительно важных вопросах. Т.о., первое противоречие заключается в попытке перейти к процессному управлению через функциональное моделирование.

Но даже в тех случаях, когда существуют качественные модели, описывающие процессы компании, разработчики ИС проектируют функционально-ориентированные средства автоматизации. При этом происходит чудовищная подмена понятий – вместо «ломки границ между функциональными подразделениями», к чему нас призывают Хаммер и Чампи, функциональная автоматизация возводит крепостные стены вокруг

этих подразделений. Таким образом, на лицо второе противоречие между целью и средствами ее достижения.

1.4. ПРОЦЕСС КАК КОНВЕЙЕР

Есть изобретения, которые радикально изменили существующий мир. Например, конвейер позволил существенно повысить производительность труда в области материального производства. Конвейер – это тщательно продуманная система организации труда, где изготовление некоторого изделия разбито на операции - такие единицы работ, которые могут быть выполнены одним участником.

Реальный конвейер выполняет транспортную и синхронизирующую функции. Во-первых, он подвозит детали к месту работы, исключая излишние перемещения исполнителя. Во-вторых, он координирует работу, если кто-то не успеет выполнить производственную операцию за отведенный тakt времени, конвейер д.б. приостановлен. Конвейер означает стандартизацию по выпуску, по работам и по нормативам.

Аналогичным образом, бизнес-процесс – это организация труда, регламентирующая порядок выполнения операций с целью добиться наивысшей возможной производительности труда и качества результата. Бизнес-процесс реализует виртуальный конвейер, это организация труда, а не автоматизация отдельных операций, что доказывается многочисленными японскими методиками управления, возникшими еще в «докомпьютерную» эру.

1.5. БИЗНЕС-ПРОЦЕСС

Известно много определений понятия бизнес-процесс, все они оперируют базовым понятием «работа». Целенаправленная работа, совершаемая субъектом над объектом труда, приводит к изменению свойств этого объекта, в результате получается продукт или услуга. Под термином бизнес-процесс мы будем понимать совокупность работ, направленную на получение воспроизводимого, повторяемого результата. Этим процесс отличается от проекта, который направлен на достижение уни-

кального результата. В этом определении делается упор на воспроизводимость, повторяемость продукта процесса, это одновременно означает, что каждый экземпляр результата м.б. однозначно идентифицирован, а количество выходов процесса, полученное за определенный интервал времени м.б. посчитано. Если организация выпускает товар, то она хочет, что бы все произведенные экземпляры были идентичны. Для этого она требует от рабочих выполнять свою работу определенным стандартом способом. Если организация оказывает услуги, она ожидает, что произведенный продукт будет удовлетворять нормативным требованиям. Для этого она должна требовать от сотрудников выполнять работу в соответствии с определенными стандартами.

Бизнес-процесс есть технология достижения запланированного результата. Технологией называют совокупность методов и инструментов для изготовления продукта с номинальным качеством и оптимальными затратами. Для этого технология предлагает разложить работу на элементарные составляющие, зафиксировать и стандартизовать их. Т.о. требование воспроизводимости результата обуславливает повторяемость выполняемых работниками действий, а не наоборот. Бизнес-процесс есть «технология производства» услуг, которые, в отличие от материального производства, не всегда имеют материальное воплощение.

1.6. ОПЕРАЦИИ И ДЕЙСТВИЯ

Бизнес-процесс состоит из операций и действий. Дадим определение этим понятиям.

Операция - это единица работы, выполняемая непрерывно, на одном рабочем месте, над одним обрабатываемым объектом. До начала выполнения операции объект имеет определенное начальное состояние. В результате выполнения операции состояние предсказуемо изменяется. Таким образом, операция описывает работу, приводящую к требуемому изменению состояния обрабатываемого изделия. Операция состоит из действий или набора действий, производимых над обрабаты-

ваемым объектом. **Действие** есть акт взаимодействия оператора с обрабатываемым изделием, в котором достигается определенная, заранее определенная, цель

Выполнение *операции* приводит к качественным изменениям обрабатываемого изделия, а *действия* – к количественным. Например, операция «проверить платежеспособность клиента» приводит к принятию решения, важного с точки зрения дальнейшего исполнения процесса. Эта проверка включает ряд действий, но их индивидуальные результаты в дальнейшем по отдельности учитываться не будут, только итоговое решение, принятое в результате всей операции.

1.7. ИМЯ БИЗНЕС-ПРОЦЕССА

Каждый процесс имеет уникальное название. Хорошее имя процесса состоит из глагола, который указывает на работу, выполняемую в процессе и существительного, указывающего на обрабатываемое изделие. Например: «выдать кредит», «предоставить услугу», «урегулировать убыток»- состоят из глагола и существительного. Иногда в название дополнительно вводится уточнение, позволяющее точнее специфицировать изделие, например, «выдать кредит физическому лицу», «предоставить телекоммуникационную услугу», «урегулировать убыток автострахования». Для процессов, которые именуются парой глагол плюс существительное можно идентифицировать результат и посчитать их число за определенный период времени. Например, указать, сколько кредитов было выдано за день, какое количество услуг или убытков было обработано за отчетный период.

Неудачными следует считать названия процессов, которые содержат вместо глагола отглагольное существительное, например, «маркетинговая деятельность», «возмещение убытка», «кредитование населения». В этих названиях нет глагола, нет действия, поэтому невозможно идентифицировать отдельный результат. Как следствие, невозможно определить, сколько «деятельности» было выполнено за отчетный период. Правильно именуя процессы, мы ограждаем себя от последующих

ошибок.

Эти же правила следует применять для именования операций, образующих процесс. Имя должно включать глагол, обозначающий работу и существительное, указывающее на объект, подвергаемый преобразованию, дополнительные слова могут уточнять либо вид работы, либо тип объекта. Например: «дополнительно проверить заявку», «рассмотреть кредитную историю».

1.8. УПРАВЛЕНИЕ БИЗНЕС-ПРОЦЕССАМИ

Управление - это целенаправленное информационное воздействие на людей и экономические объекты, осуществляющееся с целью направить их действия и получить желаемые результаты. Термин «управление бизнес-процессами» имеет несколько значений. Во-первых, он означает управление предприятием с использованием бизнес-процессов. Вся деятельность предприятия разделяется на отдельные бизнес-процессы, которые координируют работу сотрудников из разных структурных подразделений. Таким образом, бизнес-процесс пересекает границы подразделений, сформированные по функционально-иерархическому принципу. Каждый бизнес-процесс имеет своего владельца, который отвечает за конкретный результат деятельности данного процесса и руководит всеми сотрудниками, участвующими в процессе. Процессное управление подразумевает, что предприятие переходит от иерархической к матричной организационной структуре.

Во-вторых, этот термин означает управление собственно бизнес-процессами, которые как живые организмы, требуют постоянного контроля и вмешательства со стороны менеджера процесса. Рассмотрим уровни управления бизнес-процессами.

1.8.1. ТРИ УРОВНЯ УПРАВЛЕНИЯ БИЗНЕС-ПРОЦЕССАМИ

Говоря об управлении бизнес-процессами, будем выделять три уровня. Каждый из них имеет характерные особенности.

Первый уровень - **оперативное управление** экземпляра-

ми процесса, предполагает контроль параметров исполнения каждого экземпляра с целью выявить те из них, которые выполняются с отклонениями. Управляющее воздействие, в этом случае, направленно на отдельный экземпляр и имеет целью вернуть параметры исполнения в норму. Например, наивно предполагать, что все экземпляры одного бизнес-процесса исполняются с постоянной скоростью, как детали на конвейере. Отдельные экземпляры могут отставать от расписания по самым разным причинам. В этом случае следует принять меры, что бы отстающий процесс мог нагнать расписание. При оперативном управлении шаблон или схема процесса не изменяется. Эффект от оперативного управления проявляется в уменьшении брака и повышении качества исполнения процесса.

Второй уровень - **тактическое управление** на уровне группы бизнес-процессов, предполагает контроль показателей, характеризующих работу совокупности экземпляров на краткосрочном временном интервале. В этом случае, управляющее воздействие, направленно на некоторое множество экземпляров процесса, оно связано с изменением параметров процесса или перераспределением ресурсов и не предполагает изменения схемы исполнения процесса. Этот уровень управления помогает адаптировать процесс к небольшим временным изменениям условий рынка.

Третий уровень - **стратегическое управление** на уровне схемы процесса предполагает контроль параметров на долгосрочном временном интервале. Оно осуществляется в том случае, когда первый и второй уровни управления уже не в состоянии обеспечить достижения поставленных целей или произошло радикальное изменение условий ведения бизнеса в целом. Управление заключается в изменении логики (схемы) процесса

Т.о. неправильно сводить управление бизнес-процессами исключительно к изменению его модели.

1.9. ТИПЫ ИНФОРМАЦИОННЫХ СИСТЕМ

Организации, которые хотят получить наибольшую выгоду от процессного подхода, применяют специальные технические средства управления бизнес-процессом, называемые системами для управления бизнес-процессами (Business Process Management Systems). Этот термин обозначает, во-первых, информационную технологию, обеспечивающую генерацию исполняемой модели процесса непосредственно из графической модели бизнес-процесса, во-вторых, управлеченческую концепцию непрерывного совершенствования бизнес-процессов, предлагающую эволюционный подход к проведению изменений, направленных на повышение эффективности бизнеса. Системы управления бизнес-процессами относятся к классу *процессно-ориентированных систем*.

1.9.1. ПРОЦЕССНО-ОРИЕНТИРОВАННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Определим информационную систему, **базирующуюся на процессах** (process aware), как систему, при проектировании которой использовалась модель бизнес-процессов. Так, в основе большинства информационных систем управления ресурсами (ERP-систем) лежит процессная модель, но можно ли говорить, что существующие ERP-системы создавались для реализации процессного подхода? Хотя внедрение ERP начинается с описания бизнес-процессов, эта модель, как правило, используется однократно для целей написания технического задания и быстро перестает быть актуальной.

Системы **ориентированные на процессы** (process oriented), организуют взаимодействие участников друг с другом и с информационными системами таким образом, что задания, включающие информацию и документы, передаются между участниками (людьми и системами) в соответствии с формализованными процедурными правилами. К этому классу следует отнести системы управления потока работ (workflow).

Практика показывает, что несогласованная деятельность высококлассных специалистов менее эффективна, чем хорошо организованная работа работников обычной квалификации. Эффект от координации взаимодействия проявляется в том, что процедуры становятся короче, их выполнение менее затратным, качество повышается. Процессно-ориентированные системы нацелены на получение синергетического эффекта от скоординированных действий организационных единиц компании. Многие процессно-ориентированные информационные системы (ИС) содержат жестко запрограммированное описание процессов. В этом случае, изменение процедурных правил взаимодействия может оказаться трудоемким, так как потребует перепрограммирования. Можно говорить, что процессно-ориентированные системы создаются для реализации процессного подхода.

1.9.2. СИСТЕМЫ УПРАВЛЯЕМЫЕ МОДЕЛЬЮ

Отличительная особенность **системы управляемой моделью** (model driven) заключается в том, что их разработка ведется в терминах предметной области, а не используемой компьютерной среды, так что разработчик оказывается защищенным от сложностей программирования.

В процессно-ориентированных системах под моделью понимают графическое описание последовательности работ, выполняемых участниками. Созданием модели может заниматься бизнес-аналитик без знания программирования.

Разработанная модель без изменений и с минимальным программированием преобразуется в исполняемый машинный формат. Если в дальнейшем понадобится внести изменение в логику работы системы, модификации подвергается непосредственно исходная модель. Таким образом, в системе, управляемой моделью, именно визуальная модель является «исходным текстом программы», она определяет логику работы проектируемой информационной системы.

1.10. СИСТЕМЫ УПРАВЛЕНИЯ БИЗНЕС-ПРОЦЕССАМИ

Системы управления бизнес-процессами (BPMS – Business Process Management System) являются одновременно процессно-ориентированными и управляемыми моделью. Они предназначены, чтобы организовать эффективное взаимодействие всех участников процесса, помогают менеджеру процесса преодолеть результаты отклонений, возникающих в ходе исполнения процесса, позволяют контролировать выполнение заданий, как по времени, так и по качеству в соответствии с заранее определенными критериями. Логика работы таких систем полностью основывается на исполняемой визуальной модели бизнес-процесса, программирование требуется для описания нестандартных ситуаций, изменение логики работы осуществляется через модель процесса.

Современная технология управления бизнес-процессами на базе BPM – это совокупность управленческих методологий и информационных технологий.

Под управленческой методологией мы будем понимать управление предприятием с помощью бизнес-процессов. При этом вся деятельность предприятия рассматривается как совокупность взаимодействующих бизнес-процессов. Обычно, процессное управление сводится к их идентификации, стандартизации, реинжинирингу и регламентации бизнес-процессов компании. В нашем случае реинжиниринг, который изначально ориентирован на однократное радикальное преобразование бизнес-процессов компании, заменяется на постоянную целенаправленную деятельность по контролю и управлению бизнес-процессами.

Суть технологии заключается в том, что непосредственно из модели бизнес-процесса, созданной с использованием стандартной для отрасли нотации моделирования бизнес-процессов BPMN, генерируется исполняемая на компьютере модель, которая позволяет выполнять операции бизнес-процесса по всей цепочке в автоматизированном режиме, от-

слеживать отклонения, предлагать решения по их устраниению.

Благодаря тому, что за основу берется графическая, интуитивно понятная пользователю модель бизнес-процесса, центр тяжести в разработке перемещается с программиста на бизнес-аналитика. При необходимости внести изменения в порядок исполнения бизнес-процесса, соответствующие коррекции проводятся аналитиком непосредственно в модели процесса. Таким образом, бизнес-аналитик получает средство разрабатывать средства автоматизации бизнес-процессов с минимальным привлечением программистов.

Среда BPMS позволяет собрать полную статистику периода выполнения бизнес-процесса, а так же измерить реальные значения ключевых показателей эффективности. Поэтому улучшение существующей модели процесса осуществляется на основании полных, достоверных и актуальных данных.

1.11. КЛАССИФИКАЦИЯ БИЗНЕС-ПРОЦЕССОВ

Процессы компании можно систематизировать по различным классифицирующим признакам. Далее мы рассмотрим:

- основные,
- вспомогательные
- обеспечивающие процессы
- сквозные процессы
- внутри- и межорганизационные процессы
- автоматизированные и автоматические процессы

1.11.1. ОСНОВНЫЕ, ВСПОМОГАТЕЛЬНЫЕ И ОБЕСПЕЧИВАЮЩИЕ

Деятельность любой организации направлена, в первую очередь, на создание продуктов или предоставление услуг, имеющих реальную ценность для ее внешнего окружения. **Основной процесс** направлен на достижение главной цели компании.

Вследствие разделения труда участников операционной

деятельности, возрастаёт потребность в координации их действий. Организационная и управленческая деятельность служит задачам управления операционной подсистемой и ее координации. **Вспомогательные процессы** - совокупность отдельных видов деятельности, направленных на координацию функционирования и развитие организационной системы в интересах достижения стоящих перед ней целей.

Наконец, организация должна закупать изделия и товары, необходимые для основной деятельности, нанимать персонал, осуществлять хозяйственные операции. **Обеспечивающие процессы** не создают добавленной стоимости продукта, предлагаемого предприятием. Они снабжают ресурсами всю деятельность организации и обеспечивают работу основных процессов.

1.11.2. СКВОЗНЫЕ ПРОЦЕССЫ

Сквозные бизнес-процессы, пронизывают всю организацию насквозь, пересекая границы многих функциональных подразделений. Например, клиент обратился в организацию за товаром или услугой. Он разметил заказ в клиентском отделе. Далее, заказ проходит различные внутренние подразделения. Экономическая служба определяет его стоимость. Производственный отдел исполняет заказ. Бухгалтерия фиксирует оплату. Экспедицию отвечает за доставку. Наконец, опять клиентский отдел, который подписывает с заказчиком документ об успешности предоставления товара или услуги. В организации, где реализовано процессное управление, заказчик видит только клиентский отдел, а вся внутренняя кухня исполнения заказа скрыта от него в «черном ящике».

Сквозные процессы кажутся нам монолитными, но на практике они оказываются фрагментированными, распадаются на сеть взаимодействующих подпроцессов. Можно назвать несколько причин разделения сквозного процесса на подпроцессы. Во-первых, хотелось бы выделить повторно используемые компоненты, что упростит разработку и сопровождение процессной системы. Во-вторых, модель процесса, которая не

умещается на стандартном листе, кажется не понятной и сложной, поэтому аналитики группируют операции в подпроцессы.

1.11.3. СТРУКТУРИРОВАННЫЕ ПРОЦЕССЫ

По степени формализации процессы делятся на хорошо и плохо структурированные. Рассмотрим три критерия классификации степени структурированности процесса:

- 1) Выбор следующего исполнителя. В формализованном процессе следующий участник может быть известен заранее или его выбор формализован. Например, при стоимости продукта до 1000 рублей проверяет экономист, а если его цена выше, то старший экономист с более высокими полномочиями. Т.о. выбор исполнителя четко формализован. Напротив, если для каждого конкретного случая следующий исполнитель назначается на основании неформализованного выбора, то можно говорить, что исполнитель заранее неизвестен.
- 2) Выбор следующей операции. В формализованном процессе следующий следующая операция известна заранее. Например, при продаже изделия после приемки заказа идет проверка наличия продукта на складе. Обычно выбор следующей операции формализован, например, если товар есть на складе, можно переходить к организации доставки, а если его нет, то нужно заказать товар, ожидать его поступления и лишь затем организовывать доставку. Но бывают ситуации, когда следующее действие можно определить только после завершения предыдущего. Например, в юридической деятельности выбор варианта продолжения возможен только после завершения текущего судебного разбирательства, причем существует несколько вариантов продолжения, выбор между которыми неоднозначен и не м.б. формализован, поэтому осуществляется экспертом предметной области, в данном случае адвокатом.
- 3) Выбор объекта управления. Обычно с каждым процессом связан объект управления, какой либо документ, движение которого вдоль процесса определяет ход его исполнения. Например, выдача кредита начинается с заполнения документа

под названием заявка, ее движение между участниками определяет ход рассмотрения, на заявке ставят свою визу участники. Информационный объект заявка формализован, разбит по полям, каждое из них имеет уникальное имя идентификатор, позволяет хранить данные определенного вида. Формализованные документы удобны для машинной обработки. Но бывают неформализованные документы, например письма, где та же заявка может быть описана в простой письменной форме без шаблона и специальных правил, поэтому понять, где в заявке заказчик, а где адрес доставки машина не может.

Таблица 1-1 рекомендует выбор средства автоматизации в зависимости от сочетания описанных критериев. Для хорошо формализованных процессов, где следующий участник и его действие заранее известны, а документ формализован, в наибольшей степени подходят средства BPMS. В другой крайней ситуации, когда участник и его действие неизвестно, а документ не формализован, в большей степени подходят средства коллективной работы. Использование систем BPMS для слабо и плохо формализованных процессов возможно, однако в этом случае они не в полной мере раскроют свой потенциал и возможности.

Таблица 1-1. Классификация процессов по степени формализации

Критерий	Степень формализации процесса			
	Хорошо	Слабо	Плохо	Не формализован
Следующий участник	Известен	Неизвестен	Известен	Неизвестен
Следующая операция	Известен	Известен	Неизвестен	Неизвестен
Объект управления	Формализован	Формализован	Не формализован	Не формализован
Инструмент	BPMS	Case Management	DocFlow	Collaboration

1.11.4. ВНУТРИ И МЕЖОРГАНИЗАЦИОННЫЕ ПРОЦЕССЫ

Внутриорганизационные процессы описывают взаимодействие, осуществляемое в пределах одной организации. Они делятся на основные, вспомогательные и управления.

1.11.4.1. МЕЖОРГАНИЗАЦИОННОЕ ВЗАИМОДЕЙСТВИЕ

Межорганизационные процессы отвечают за взаимодействие нескольких предприятий, связанных задачами общего бизнеса. Это может быть взаимодействие компании со своим клиентом или партнером по кооперации. Такие процессы обладают повышенной сложностью, ведь вначале надо стандартизировать выполнение работ внутри каждого из предприятий, а затем переходить к стандартизации их взаимодействия.

Реализация процессов межорганизационного взаимодействия происходит, обычно, в два этапа. На первом этапе организации «договариваются» о формате документов, используемых при обмене. На втором этапе они договариваются о последовательности операций по пересылке этих документов.

Для решения первой задачи организации устанавливают единую терминологию для определения бизнес понятий, входящих в документы, пересылаемые между торговыми партнерами. Например, Консорциум RosettaNet, объединяющий более 500 ведущих ИТ компаний создал одноименный индустриальный стандарт для B2B коммуникаций. Словари RosettaNet устанавливают единую терминологию для определения бизнес транзакций между торговыми партнерами, товаров и услуг. На базе общих словарей строятся электронные документы, которые «понятны» всем участникам обмена и могут «электронно» стыковаться с корпоративными системами каждого из участников.

Для решения второй задачи организации стандартизуют последовательности действий по обмену электронными документами. Диаграммы взаимодействия, хореографии и диалогов, рассматриваемые ниже, решают данную задачу.

1.11.5. ПРОЦЕССЫ АВТОМАТИЗИРОВАННЫЕ И АВТОМАТИЧЕСКИЕ

Бизнес-процессы можно различать по степени вовлечения человека в производственную деятельность. **Автоматизированные процессы** происходят с участием человека в контуре управления. **Автоматические процессы** осуществляются без его участия. Первые являются интерактивными, они основаны на двусторонней диалоговой связи между человеком исполнителем и центральным исполняющим узлом, который воспринимает ввод команд и данных от пользователя во время работы. Вторые исполняются без участия человека, всю работу выполняет машина.

Различия в следующем: работа машины возможна, когда задание полностью формализовано, точно известны все критерии принятия решения. При этом каждую операцию принято рассматривать как сервис, подав на вход требуемые данные, мы вправе ожидать на выходе предполагаемый результат. В автоматизированном процессе решения менее формализованы. Предполагается, что человек может принять решение, пользуясь личным опытом и навыками. Но это вносит в исполнение процесса определенный субъективизм, человек часто выполняет работу в соответствии с личными взглядами, интересами или вкусами, как результат возникает вариативность исполнения процесса.

1.12. ПРОЦЕСС И ДОКУМЕНТООБОРОТ

Часто бизнес-процесс смешивают с движением документа. Безусловно, любой процесс связан с некоторым документом, движение которого определяет ход исполнения процесса. Например, при выдаче кредита основным документом является заявка клиента, ее обработка участниками образует операции бизнес-процесса. Различие между процессом и документооборотом в следующем: жизненный цикл процесса шире, чем у одного документа, обычно он покрывает несколько документов. Например, процесс «от заказа до отгрузки» подразумевает ра-

боту с несколькими документами: «заявкой», содержащей пожелания клиента; «сметой», описывающей экономическую сущность заказа; «заданием», являющимся инструкцией по изготовлению заказа; «счетом», который является платежным документом; наконец «накладной», которая сопровождает отгрузку товара или услуги. На разных этапах исполнения процесса соответствующий документ становится объектом управления, определяя статус исполнения всего процесса.

В рамках т.н. распорядительного документооборота, все эти документы рассматриваются по отдельности. Для контроля исполнения отдельных поручений, возникающих в рамках работы с документом, компании ведут разнообразные журналы регистрации документов, фиксируют резолюции, связанные с документами, что, в конечном счете, позволяет определить текущее состояния обработки, оперативно получать агрегированную информацию о состоянии исполнения документов в компании. В рамках бизнес-процесса эти документы рассматриваются во взаимосвязи, большое внимание уделяется трансформации и переносу информации из одного документа в следующий.

1.13. ОБЪЕКТ УПРАВЛЕНИЯ ПРОЦЕССА

В программировании графиком потока управления принято называть множество всех возможных путей исполнения некоторой программы, представленное в виде графа, а потоком управления — траекторию, по которой исполняется соответствующая программа. В управлении бизнес-процессами роль графа играет модель процесса, которая должна показывать все возможные сценарии исполнения, а поток управления есть траектория, по которой в данный момент времени исполняется отдельный экземпляр.

Чтобы понять поток управления, введем понятие объект управления бизнес-процесса. Дело в том, что каждый процесс связан с обработкой одного или нескольких документов (информационных объектов). Среди этих информационных объектов есть такой, который сохраняет в себе результат выполне-

ния текущей операции, этапа исполнения или всего процесса целиком и тем самым связывает соответствующие входы и выходы. Выполнение операций процесса приводит к последовательному изменению состояний объекта. Назовем информационный объект, связывающий входы и выходы процесса или подпроцесса - **объектом управления**. Вспомогательными будем называть остальные информационные объекты, которые фиксируют изменения данных бизнес-процесса, но не результат выполнения операции. Например, в кредитном процессе к заявлению прилагаются различные справки, последние являются информационными объектами, но результаты рассмотрения отмечаются в заявке на получение кредита.

В каждый момент времени в BPMS системе рассматривается только один объект управления. Жизненный цикл объекта управления м.б. короче, чем весь процесс, в этом случае мы наблюдаем смену объекта управления. Движение объекта управления по траектории процесса определяет порядок исполнения операций процесса. Прибытие объекта управления в соответствующую операцию, делает ее активной и подготавливает ее к исполнению. Траектория движения объекта управления образует поток управления процесса.

1.13.1. МАРКЕР ПОТОКА УПРАВЛЕНИЯ ПРОЦЕССА

В теории бизнес-процессов для объяснения потока управления принято использовать понятие маркер процесса, который определяется как «теоретическое понятие», он используется, чтобы отобразить ход исполнения бизнес-процесса. Какая сущность скрывается за этим понятием, не раскрывается.

Мы будем ассоциировать **маркер потока управления процесса** с объектом управления процесса, поскольку траектория движения последнего определяет порядок исполнения процесса. В программировании сходную функцию исполняет переменная состояния, ее движение вдоль процесса образует поток управления. В аналитическом моделировании роль маркера играет некоторый документ, например, «заявка», т.ч. выполнение очередной операции начинается, после поступления

«заявки». При моделировании исполняемых процессов объект управления может быть шире, чем обычный документ, включая служебные поля, например, статус обработки, которого в исходном бумажном документе нет.

1.13.2. ПАРАЛЛЕЛЬНЫЕ ПОТОКИ УПРАВЛЕНИЯ ПРОЦЕССА

Один поток управления может вначале разделяться на несколько ветвей, при этом образуются **параллельные потоки управления процесса**. Возникновение параллельных потоков управления обычно связано со сменой объекта управления. Например, заказчик разместил заказ, содержащий номенклатуру требуемых деталей, по каждому изделию из списка мы посылаем отдельный запрос на склад, с целью проверить наличие соответствующего продукта. Основной процесс делится на параллельные потоки, в каждом из них есть свой объект управления, данном примере им является **запрос**.

При этом не регламентируется обязательная синхронизация всех потоков при их слиянии. Т.о. один поток на узле ветвления может породить несколько потоков на узле слияния, т.ч. по схеме будут двигаться несколько маркеров, порожденных из одного объекта управления. На практике такая ситуация может привести к коллизии. Следовательно, при проектировании схем процессов надо следить за ветвлением и слиянием параллельных потоков, их синхронизацией. Мы подробно рассмотрим эту ситуацию в разделе 10 «Процессные паттерны».

1.13.3. МНОГОПОТОЧНЫЕ ОПЕРАЦИИ

Процесс может включать **многопоточные операции**, которые на этапе выполнения порождают много экземпляров исполнения одной задачи, причем их количество может быть не известно в момент моделирования. Например, операция «Отправить приглашение гостям» д.б. выполнена многократно, столько раз, сколько гостей в списке приглашенных. Различают последовательное и параллельное исполнение. Если все экземпляры работают с одним общим набором данных, то исполнение

ние, обычно, последовательное, т.к. каждая следующая итерация перезаписывает данные, внесенные на предыдущем шаге.

Если каждый экземпляр использует индивидуальный набор данных, то исполнение м.б. одновременным. Например, при рассылке приглашений, наборы данных разных экземпляров не зависят друг от друга, т.ч. могут исполняться параллельно и одновременно друг с другом. В этом случае, основной поток управления разделяется на несколько параллельных нитей. С точки зрения выполнения, каждый поток является дочерним экземпляром основного процесса, в котором он был создан. Соответственно, в каждом экземпляре процесса одновременно может выполняться некоторое число экземпляров потоков, порожденных одной операцией.

При этом не регламентируется обязательная синхронизация потоков операции. Это означает, что каждый поток при слиянии может породить отдельные маркеры потока управления. Так же как в предыдущем случае, надо следить за разделением процесса на потоки и их дальнейшей синхронизацией, поскольку это может привести к различным коллизиям при выполнении всего процесса.

1.14. МОДЕЛЬ БИЗНЕС-ПРОЦЕССА

Моделью принято называть некоторый материальный или мысленно представляемый объект или явление, являющийся упрощённой версией моделируемого прототипа. Модель в достаточной степени повторяет одни свойства, существенные для целей конкретного моделирования, и опускает другие, несущественные свойства, в которых модель может отличаться от прототипа.

Набор свойств модели бизнес-процесса определяется целями моделирования. Если поставить целью понимание того, как исполняется процесс, то для этого достаточно ограничиться описанием укрупненных операций. Если же цель - использовать модель как программу, определяющую регламент взаимодействия пользователей, то она должна быть точной, полной и исполняемой.

1.14.1. ПРОЦЕСС ОПИСЫВАЕТ РАБОТУ

Из приведенного выше определения можно понять, что процесс описывает работу, которую необходимо выполнить, чтобы получить запланированный продукт или услугу. Что следует делать, чтобы спланировать работу? Для этого следует создать список работ, которые необходимо выполнить, а затем, составить расписание исполнения указанных работ. В проектном менеджменте список работ получил название **структурированная декомпозиция работ** (WBS). Разработано множество техник составления такого списка, обычно используется декомпозиция работ. Однако никто не задумывается о влиянии выбора стратегии декомпозиции на результат.

1.14.2. СПОСОБЫ ДЕКОМПОЗИЦИИ ПРОЦЕССА

Методология структурного анализа и проектирования (SADT) предполагает рассматривать сложный объект как иерархическую многоуровневую модульную систему. Концепция декомпозиции предполагает разделять исходный объект на составляющие, но таким образом, чтобы из компонентов можно было восстановить, синтезировать исходный объект. Таким образом, система рассматривается как фиксированное упорядоченное множество объектов и отношений между ними. Авторы SADT рассматривают четыре способа декомпозиции системы.

1. Функциональная декомпозиция позволяет получить перечень всех тех действий, которые необходимо выполнить, чтобы добиться запланированного результата. В результате получается функциональная модель, которая не привязана к конкретной организационной структуре предприятия.
2. Декомпозиция в соответствии с известными стабильными подсистемами приводит к созданию иерархии, привязанной к существующей структуре организации. Как следствие, модель окажется неприменима к организации с иной структурой. Вместе с тем, она помогает понять распределение работ между подразделениями, например, какие работы выполняет бухгалтерия, а какие экономический отдел.
3. Декомпозиция, основанная на отслеживании жизненного

цикла какого-либо объекта, позволяет увидеть очередность основных этапов процесса. Однако реальные процессы имеют более сложные сценарии исполнения.

4. Декомпозиция по физическому процессу показывает последовательность действий, направленную на получение конкретного результата, отвечает на вопрос «Как выполняется работа?» и обеспечивает получение процессной модели. Она включает расписание исполнения указанных работ.

Авторы SADT рекомендуют функциональную декомпозицию, но не отвергают остальные стратегии. Но последователи, забыв рекомендации, используют только функциональную декомпозицию. Интересно, что за моделями IDEF0 прочно закрепилось название функциональная диаграмма. Последнее верно лишь в случае использования функциональной декомпозиции. Но если использовать декомпозицию по этапам жизненного цикла, с помощью диаграммы IDEF0 можно описывать процесс.

1.14.3. ФУНКЦИОНАЛЬНЫЕ И ПРОЦЕССНЫЕ МОДЕЛИ

Итак, можно предположить, что именно способ декомпозиции определяет тип результирующей модели. Если использовалась функциональная декомпозиция, результат можно классифицировать как **функциональную модель**, она отвечает на вопрос: «что делать?» и содержит перечень операций и действий, которые необходимо выполнить, мы ранее договорились называть такой список *структурной декомпозицией работ*. Если же применялась декомпозиция по процессу, то результирующую модель можно назвать **процессной**, она отвечает на вопрос: «как выполнять работу?». Она располагает операции и действия во временном порядке их выполнения.

1.15. ИСПОЛНЯЕМАЯ МОДЕЛЬ БИЗНЕС-ПРОЦЕССА

Системы ВРМ имеют дело с так называемыми **исполняемыми**

моделями. **Исполняемая модель бизнес-процесса** – это описание участников процесса: людей и машин, а также порядка и времени выполняемых ими операций и действий, которое может быть использовано для автоматизации взаимодействия участников друг с другом и машинами без дополнительного кодирования и программирования.

Исполняемая модель описывает динамику поведения организационной системы. Можно предположить, что она существенно сложнее, объемнее, чем диаграмма бизнес-процесса, используемая с целью аналитического описания бизнес-процесса. Исполняемая модель процесса включает следующие слои, называемые перспективами:

1. Функциональная - определяет состав выполняемых работ;
2. Поведенческая - описывает порядок исполнения операций и действий процесса;
3. Информационная - определяет бизнес сущности предметной области процесса;
4. Организационная – изображает совокупность способов, посредством которых процесс разделяется на отдельные операции, стратегию распределения работ между исполнителями, методы координации действий участников.

Функциональная модель описывает статику системы, помогает ответить на вопрос «что надо делать, что бы достичь поставленной цели?». Она представляет собой справочник, структурную декомпозицию работ, перечисляет все действия, выполняемые субъектами, но не указывает порядок их исполнения. Модель описывает «идеальный» взгляд на деятельность организации - две компании, работающие в одной сфере бизнеса, выполняют одинаковые работы, однако последовательность операций может отличаться, поскольку предприятия обладают различной организационной структурой, производственной культурой и т.д.

Поведенческая перспектива описывает динамику системы и отвечает на вопрос «Как исполняется процесс?». Она включает: (1) аспект бизнес логики, процедурное описание порядка выполнения операций; (2) временной аспект – определяющий

хронологию событий, время, когда стартуют операции, их длительность; (3) аспект бизнес-правил - декларативное описание ограничений, накладываемых на выполнение процесса.

Информационная перспектива определяет связи между документами и элементами данных. Документы процесса делятся на структурированные и неструктурированные, хранимые в виде образа. Для описания информационной перспективы используется иерархическая объектная модель данных, включающая методы, которые определяют способы работы с соответствующими бизнес объектами. Эта модель не описывает способы хранения информации, но показывает связи между отдельными элементами. Разные структурированные документы могут содержать общую информацию, так что данные, введенные в один документ, становятся доступны в других документах. Это означает, что информационная модель должна описывать связи между документами по данным.

Организационная перспектива модели бизнес-процесса описывает динамику распределения работ между сотрудниками в отличие от организационной структуры компании, которая описывает статику распределения сотрудников по структурным подразделениям. Организационная перспектива должна дать ответы на вопросы:

1. Как отобрать кандидатов на выполнение каждой операции?
2. Кого из кандидатов следует назначить исполнителем?
3. Каковы привилегии исполнителя, назначенного на выполнение задачи?

Интегрированная модель бизнес-процесса суть взаимоувязанная совокупность перечисленных частных моделей, каждая из которых описывает отдельные аспекты его структуры, а все вместе они образуют полное представление характеризующее динамику его исполнения. Частные перспективы, образующие интегрированную модель, должны быть взаимоувязаны и хорошо интегрированы между собой. В противном случае, потребуется дополнительное кодирование и программирование, модель станет сложно сопровождать, она потеряет

свойство гибкости и адаптивности.

1.16. ЭКЗЕМПЛЯР ПРОЦЕССА

Как известно, для каждого шаблона процесса одновременно может исполняться неограниченное число экземпляров процессов (ЭП). Под экземпляром мы будем понимать частный случай исполнения шаблона процесса. В системе могут одновременно существовать много экземпляров процессов, каждый из которых находится на определенной стадии обработки.

1.17. ВЕРСИЯ ПРОЦЕССА

В результате изменений шаблона процесса появляются версии модели. Договоримся различать версионность на уровне схемы процесса и на уровне исполняемой модели процесса. Первая является инструментом разработчика, отслеживающим изменения, внесенные в схему процесса, вторая – инструментом администратора системы, который контролирует экземпляры процесса, исполняемые по данной версии схемы процесса.

1.17.1. ВЕРСИЯ МОДЕЛИ ПРОЦЕССА

В ходе постоянных улучшений бизнес-процесса принято фиксировать версии исполняемых моделей. Это позволяет, в случае неудачных изменений в модели, вернуться к предыдущему варианту. Версионирование исполняемых экземпляров процесса осуществляется с помощью централизованной системы управления версиями – специализированного программного обеспечения для облегчения коллективной работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одной и той же модели, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение.

1.17.2. ВЕРСИОНИРОВНИЕ ЭКЗЕМПЛЯРОВ ПРОЦЕССА

Версионность исполняемой модели процесса означает возможность одновременного исполнения в одной ИТ системе

сразу нескольких экземпляров одного процесса, соответствующих разным версиям его шаблона. Например, мы планируем ввести в производство исполнение процесса по новым правилам. Однако процессы, стартовавшие ранее, могут быть закончены в соответствии со старыми правилами, существовавшими на момент их запуска. Может возникнуть и обратная задача: перевести ранее запущенные процессы на вновь вводимый в эксплуатацию шаблон.

При этом разработчик должен помнить, что версии шаблона могут быть совместимыми, допуская простой перевод ранее запущенного на исполнение процесса на новую схему, и несовместимыми, когда перевод процесса возможен только при определенных условиях. Т.о., стратегическое управление не ограничивается заменой одного шаблона на другой, но охватывает еще управление версиями процесса, включая миграцию процесса со «старого» шаблона на «новый».

1.18. СТАНДАРТЫ ОПИСАНИЯ БИЗНЕС-ПРОЦЕССОВ

Существует много способов описания бизнес-процессов, для этих целей используются различные языки и нотации, но, к сожалению, большинство языков и нотаций не позволяют описывать одновременно внутри- и меж- организационные процессы. Рассмотрим некоторые из них:

BPEL Язык исполнения бизнес-процессов (Business Process Execution Language, BPEL) - индустриальный стандарт, описания выполняемых ориентированных на интеграцию моделей процессов. Выполнение бизнес-функций осуществляется путем вызова соответствующих веб-служб. Оригинальное название спецификации Business Process Execution Language for Web Services, BPEL4WS. Нотация не поддерживает визуальное моделирование бизнес-процессов.

- Первая версия стандарта, опубликованная в 2002, являлась совместной разработкой IBM, BEA, SAP, Siebel и Microsoft .
- Текущая версия BPEL 2.0, опубликована в 2007 г.

- Поддерживается организацией OASIS (Organization for the Advancement of Structured Information Standards, www.oasis-open.org).

BPMN Графическая нотация и модель бизнес-процессов (Business Process Modeling Notation) – индустриальный стандарт визуального описания исполняемых моделей процессов, ориентированных на интерактивное взаимодействие с участниками. Используется в большинстве систем BPMS в качестве основного средства для графического моделирования, имеет техническую реализацию, т.ч. модель м.б. интерпретирована в исполняемый программный код.

- Права принадлежат рабочей группе (консорциуму) OMG (Object Management Group, www.omg.org), занимающемуся разработкой и продвижением объектно-ориентированных технологий и стандартов.
- Текущая версия BPMN 2.0 опубликована в 2011 г.

EPC Ориентированная на событие цепочка процессов (Event-driven Process Chains, EPC) – проприетарная нотация описания бизнес-процессов. Широко используется для документирования рабочих процессов.

- Правообладателем является Software AG (www.softwareag.com).

UML Унифицированный язык моделирования (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля. Это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация программного кода.

- Права принадлежат рабочей группе (консорциуму) OMG (Object Management Group, www.omg.org).
- Формальная спецификация последней версии UML 2.0 опубликована в 2005 г.

- Последняя версия UML 2.4.1 опубликована в 2011 г.
- UML 1.4.2 принят в качестве международного стандарта ISO/IEC 1950

WSDL Язык описания веб-служб и доступа к ним, основанный на языке XML (Web Services Description Language). Служит для описания интерфейсов веб-служб, используется для моделирования доступных операций, включая адреса их вызова.

- Права принадлежат всемирному интернет консорциуму W3C (World Wide Web Consortium, www.w3.org/Consortium).

- Последняя официальная версия 2.0 опубликована в 2007 г.

XPDL Формат описания процесса на основе XML (XML Process Definition Language). Является техническим стандартом, используемым для хранения, исполнения и переноса моделей бизнес-процесса между различными BPMS средствами.

- Права принадлежат коалиции WfMC (Workflow Management Coalition, www.wfmc.org).
- Текущая версия XPDL 2.2 опубликована в 2012.

XSD Язык описания структуры XML-документа (XML Schema Definition). Это стандарт описания данных, которыми пользуются и обмениваются бизнес-процессы и веб-службы. Такая система состоит из спецификации новых элементов XML, их атрибутов, а также их производных элементов.

- Спецификация XML Schema является рекомендацией всемирного интернет консорциума W3C.
- Последняя версия 1.0 была одобрена в качестве рекомендации в 2001 г.

Предметом дальнейшего рассмотрения станет нотация BPMN 2.0.

2. СПЕЦИФИКАЦИЯ BPMN 2.0

Нотация BPMN 2.0 является одновременно средством графического визуального моделирования бизнес-процессов и языком создания исполняемой модели бизнес-процесса. Основным преимуществом данной нотации является то, что она ориентирована на широкий круг специалистов, вовлеченных в описание и автоматизацию бизнес-процессов: начиная с аналитиков и разработчиков и заканчивая экспертами предметной области. Нотация позволяет создавать диаграммы процессов на различных уровнях абстракции: от концептуальных моделей взаимодействия участников процессов (в том числе контрагентов) до технических схем, которые содержат всю необходимую информацию о том, как выполняется процесс.

Нотация BPMN относится к классу индустриальных стандартов. Она разработана общественной организацией Object Management Group и утверждена общим решением лидеров ИТ рынка. Статус индустриального стандарта означает, что она носит рекомендательный, а не обязательный характер, однако высокий статус организации разработчика, его независимость от конкретного производителя и поддержка лидеров компьютерной индустрии обеспечивают BPMN широкое распространение. В настоящее время актуальной является вторая версия стандарта, которая реализована в большинстве продуктов для моделирования и управления бизнес-процессами, присутствующими на рынке.

Нотация BPMN одновременно конкурирует и взаимно дополняет язык описания бизнес-процессов WS-BPEL. Первоначально два стандарта предназначались для решения одной задачи описать работу участников процесса. Язык WS-BPEL также имеет статус индустриального стандарта. Он рассматривает операцию, выполняемую сотрудником, как сервис, в его крайнем проявлении как веб-сервис, а весь процесс, как последовательность вызовов сервисов (веб-сервисов). Такой способ описания удобен для реализации хорошо formalизованных процессов, в которых не участвуют люди. Например, при выстав-

лении счета клиенту телекоммуникационной компанией необходимо наладить взаимодействие между коммутатором, который хранит реальную длительность разговоров клиента и биллинговой системой, которая должна подсчитать общую стоимость звонков. Взаимодействие двух систем происходит с использованием веб-сервисов, интерактивное взаимодействие с участниками не требуется, поэтому этот процесс может быть легко описан на WS-BPEL. Но, если в ходе процесса необходимо решение человека, такой способ описания становится менее удобен для моделирования. В последнее время наметилась тенденция интеграции обеих спецификаций, в частности модель, созданная в нотации BPMN, может быть преобразована для последующего исполнения в формат языка WS-BPEL. Такое преобразование стало частью спецификации BPMN 2.0 подготовленной OMG, о которой пойдет речь далее.

Язык моделирования имеет свой **синтаксис** (условные обозначения различных элементов и правила их сочетания) и **семантику** (правила толкования моделей и их элементов). **Семантика нотации BPMN** очень важна, поскольку именно она определяет то, как будут исполняться в реальности те графические элементы, которые указаны на схеме процесса.

2.1. ИСТОРИЯ РАЗВИТИЯ НОТАЦИИ BPMN

Развитием нотации BPMN (Business Process Model and Notation) в 2001-2004 годах занималась организация Business Process Management Initiative (BPMI). При этом нотация определяла только внешний вид графических элементов, но не определяла семантику их исполнения. Первоначально аббревиатура расшифровалась как Business Process Management Notation. Позднее расшифровка изменилась - *Business Process Model and Notation* означает: *нотация и модель бизнес-процессов*. Это название подчеркивает, что спецификация охватывает как графическую нотацию, так и исполняемую модель. В 2004 разработка спецификации перешла к организации Object Management Group, а в феврале 2006 года OMG опубликовала эту спецификацию

уже как свою собственную.

В 2011 году OMG выпустила последнюю на сегодняшний день спецификацию BPMN 2.0, которая является стандартом де-факто для организаций, которые занимаются моделированием и созданием исполняемых моделей бизнес-процессов. Начиная со второй версии, спецификация BPMN расшифровывается как: Business Process Modeling and Notation, подчеркивая, что она описывает графическое представление и модель процесса.

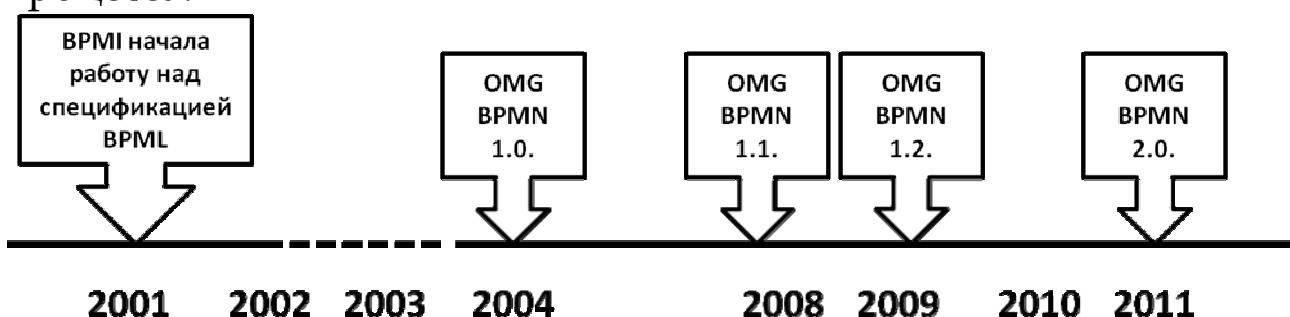


Рисунок 2-1. История развития нотации BPMN

Спецификация BPMN 2.0 состоит из трех компонентов. Во-первых, она определяет набор графических элементов и правил их применения для моделирования последовательности работ, из которых состоит бизнес-процесс. Во-вторых, спецификация опирается на формализованное описание графических элементов и отношений между ними, что позволяет переносить диаграммы между различными средствами моделирования, поддерживающими нотацию BPMN 2.0, без потери какой-либо смысловой и графической информации. В-третьих, спецификация BPMN 2.0 описывает семантику выполнения графических элементов, что позволяет генерировать в автоматическом режиме программный код на языках исполнения бизнес-процессов, таких как WS-BPEL, YAWL и др. Нотация BPMN 2.0 разрешает моделировать как бизнес-процессы, которые выполняются в рамках одной организации, так и те, в исполнении которых участвуют несколько контрагентов. Таким образом, нотация позволяет описывать все категории процессов электронной коммерции: B2B, C2C, B2C и т.д.

Вместе с тем, неправильно сводить исполнение бизнес-

процесса исключительно к преобразованию в формат языка WS-BEPL. По-прежнему декларируется возможность преобразования модели в исполняемый формат XPDL (XML Process Definition Language). Язык XPDL предназначенный для описания определений и реализаций **рабочих процессов** и используется для переноса исполняемой модели бизнес-процессов между разными средами BPMS. Он предложен и развивается Workflow Management Coalition, в настоящее время текущей версией является XPDL 2.2.

2.2. ОБЛАСТЬ ПРИМЕНЕНИЯ НОТАЦИИ BPMN

Нотация BPMN предназначена для описания:

- Порядка исполнения работ образующих бизнес-процесс;
- Потоков данных между операциями процесса;
- Потоков сообщений между процессами;
- Ассоциации обрабатываемых объектов данных с операциями процесса.

Моделирование осуществляется с помощью визуальных диаграмм, что позволяет участникам быстрее понять логику исполнения.

Нотация BPMN не позволяет моделировать другие аспекты модели бизнес-процесса, например:

- Функциональную (структурную) декомпозицию работ;
- Организационную структуру предприятия;
- Модель данных;
- Бизнес правила,
- Бизнес стратегию компании

Поскольку интегрированная модель бизнес-процесса включает не только поведенческую перспективу, но также другие аспекты, описываемые перечисленными моделями, спецификация BPMN уделяет повышенное внимание вопросам интеграции моделей. Интеграция осуществляется на уровне метамодели BPMN.

2.3. ОБЗОР ОСНОВНЫХ ЭЛЕМЕНТОВ НОТАЦИИ

В нотации BPMN 2.0 можно выделить пять основных категорий графических элементов, которые используются для создания схем оркестровки бизнес-процессов (см. Рисунок 2-2):

1. Элементы управления;
2. Соединительные элементы;
3. Артефакты;
4. Данные.
5. Зоны ответственности;

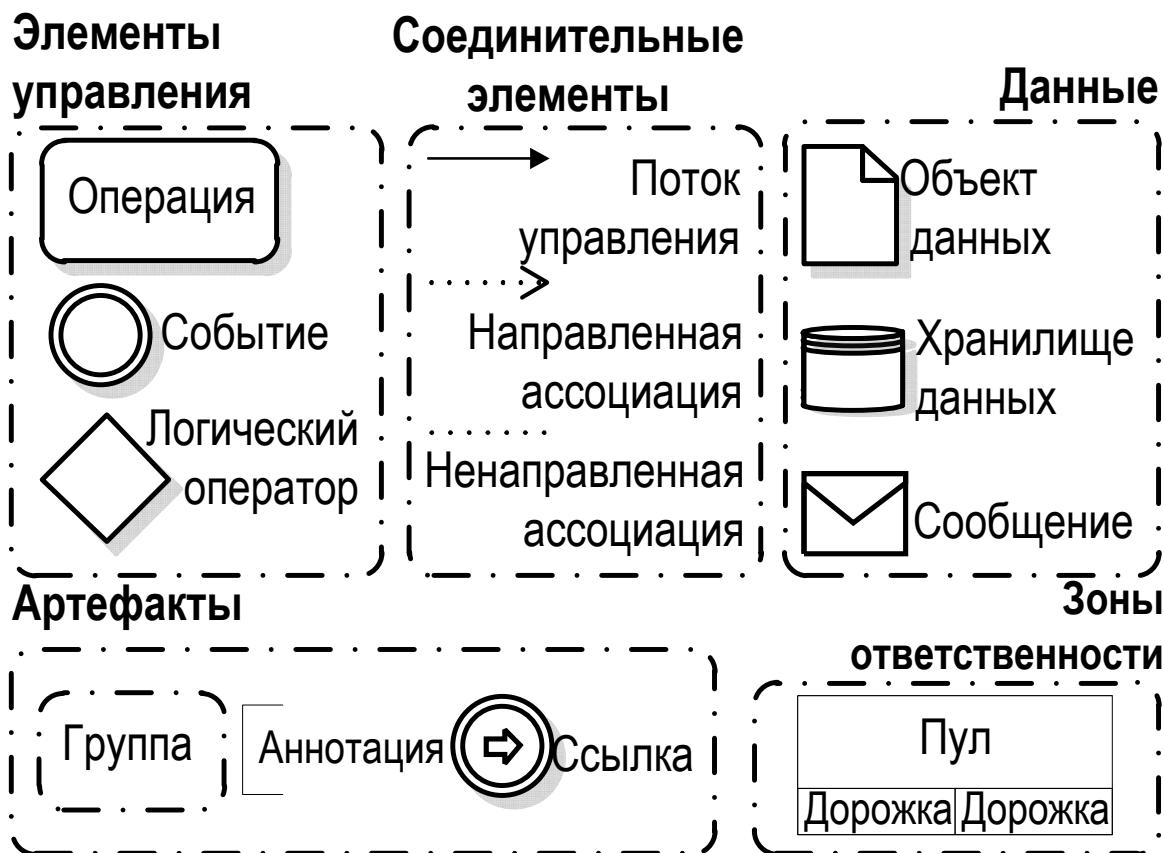


Рисунок 2-2. Основные графические элементы в нотации BPMN

2.3.1. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Элементы управления включают: *операции* (см. п. 3), *логические операторы* (см. п. 5) и *события* (см. п. 6.3).

Операция обозначает единицу работы, в результате которой изменяется состояние объекта управления, например, «Согласовать заявку», «Вынести решение» и т.д.

Логический оператор изображают работу, которая не изменяет объект, но маршрутизирует его в соответствии с некоторым правилом. Например, если величина запрошенного кредита превышает 1000 рублей, то его согласует старший менеджер.

Событие используется для нескольких целей. Во-первых, что бы указать моменты времени, когда выполняется работа. Например, начать выполнение очередной операции через 1 час, после завершения предыдущей. Во-вторых, что бы ограничить длительность операций. Например, прервать исполнение операции через 30 минут после начала. В-третьих, они описывают реакцию на изменение состояния внешних, по отношению к процессу объектов. Например, продолжить исполнение, после получения сигнала.

2.3.2. СОЕДИНИТЕЛЬНЫЕ ЭЛЕМЕНТЫ

Соединительные элементы предназначены для связывания остальных элементов нотации. Они включают: потоки управления (см. п. 4), ассоциации, потоки сообщений (см. п. 6.5.1).

Потоки управления связывают отдельные *операции, логические операторы и события* в логическую цепочку и устанавливают порядок их выполнения.

Ненаправленные ассоциации используется для соединения артефактов с элементами управления и потоками управления, они не изображают последовательность выполнения работ.

Направленные ассоциации служат для указания направления передачи объектов данных и для указания операции, которая выполнит компенсацию в случае отката транзакции (см. п. 3.2.6).

Потоки сообщений (см. п. 6.5.1) позволяют отобразить обмен информационными посылками между участниками процесса. Следует обратить внимание, что модель, изображает факт пересылки информации, но не позволяет отобразить структуру информационного сообщения.

Сообщения помогают описать структуру информационного обмена между процессами. Если один процесс иницииру-

ет исполнение другого процесса, то необходимо изобразить, какая информация передается от вызывающего процесса в вызываемый. Однако следует учитывать, что информационная структура объектов отображается, но не моделируется.

2.3.3. ЭЛЕМЕНТЫ ДАННЫХ

Элементы данных используются для изображения информационных потоков на диаграмме процесса. Они включают: объекты данных (см. п. 8), хранилища данных (см. п. 8.1.5), сообщения.

Объекты данных позволяют описать внутреннюю структуру информационных объектов, которые подвергаются обработке в ходе исполнения операций.

Хранилища данных изображают внешние по отношению к процессу системы хранения, например, СУБД.

Сообщения изображают на схеме процесса информационные посылки, которыми обмениваются между собой процессы.

2.3.4. ЗОНЫ ОТВЕТСТВЕННОСТИ

Зоны ответственности – *пулы* и *дорожки* есть графические элементы, служащие для логической группировки операций процесса.

Пул это «контейнер», который очерчивает границы процесса. Название пула может указывать владельца процесса. В некоторых случаях пул не рисуется, но предполагается. Так или иначе, пул всегда явно или неявно присутствует на диаграмме. Поток управления не может пересекать его границу. Напротив, поток сообщений может изображаться между пулами, но не может соединять операции внутри одного пула. Если пул показывает детали процесса: *операции, логические операторы, события и потоки управления*, его называют «белый» ящик. Напротив, пул, который не показывает деталей процесса, называют «черный» ящик.

Пул разделен на **дорожки**, которые служат для группировки операций диаграммы. Дорожки имеют имя. Первонач-

чально дорожки предназначались, чтобы визуализировать исполнителя задания, так что имя дорожки указывало на роль участника. Однако в настоящее время *ролевая модель доступа* используется редко, дорожки чаще используются для аналитического моделирования, т.ч. именование дорожек не оказывает влияния на выбор исполнителя. Дорожки могут быть иерархически вложенными, отражая иерархию группировок операций, выбранную аналитиком. Один процесс может включать несколько иерархий дорожек. Поток управления может пересекать границы дорожек.

2.3.5. АРТЕФАКТЫ

Артефакты - это графические элементы, для которых спецификация BPMN не определяет семантики исполнения, они используются для комментирования процесса, например, для аннотирования отдельных операций на схеме. Т.о. на схеме они изображаются, а при исполнении не учитываются. К этой категории элементов относятся *группы (операций)*, *ассоциации* и *аннотации*. В целом, артефакты никак не влияют на выполнение процесса.

Группы – это способ логически объединить на схеме несколько операций процесса. Чаще всего группирование операций используется для того, чтобы скрыть излишние детали процесса. Группы могут пересекать несколько дорожек и даже пулов.

Аннотации есть способ добавить на схему необходимые комментарии. **Ассоциация** логически связывает комментарий и некоторую операцию.

2.4. СУБКЛАССЫ НОТАЦИИ BPMN

Если сравнить нотацию BPMN 2.0 с предыдущей версией BPMN1.2, можно обнаружить, что число графических элементов нотации увеличилось с 55 до 116, что делает работу с нотацией достаточно сложной и кропотливой. Что бы упростить работу бизнес-аналитиков предлагается разделить все множество элементов на несколько групп (см. Рисунок 2-3).

Полный набор + 50 элементов

Аналитический набор + 29 элементов

Описательный набор

Пул
Дорожка
Поток сообщений
Пользовательское задание
Сервисное задание
Повторно используемый подпроцесс
Объект данных
Входной объект данных
Выходной объект данных
Хранилище данных
Аннотация
Ассоциация данных
Стартовое событие получение сообщения
Завершающее событие отправка сообщения
Стартовое событие таймер
Завершающее событие таймер

Основной набор

Операция
Подпроцесс
Поток управления
Логические элементы:
XOR
AND
Начальное событие
Завершающее
событие

Рисунок 2-3. Субклассы нотации BPMN

- Основной набор включает 7 элементов, достаточен для разработки концептуальной (не исполняемой) модели процесса.
- Подмножество описательных элементов (+17) достаточно для построения исполняемой модели;
- Подмножество аналитических элементов (+29)
- Наконец, полный набор (+50) позволяет создавать любые типы диаграмм.

Следует отметить, что существующие сегодня системы BPMS поддерживают не все 100% элементов нотации, а только какое-то подмножество. При этом вендоры не придерживаются описанных субклассов, а руководствуются своими индивидуальными пристрастиями. В случае возникновения вопросов, следует внимательно изучить справочное руководство соответ-

ствующего продукта.

2.4.1. ПРИМЕР ИСПОЛЬЗОВАНИЯ ЭЛЕМЕНТОВ НОТАЦИИ BPMN

Рассмотрим пример (см. Рисунок 2-4), иллюстрирующий использование основного набора элементов нотации BPMN. Начало процесса обозначено *стартовым событием*, а окончание – *завершающим*. Все элементы *управления* между началом и стартом связаны в единую цепочку управления посредством *потока управления*. Ветвление и слияние потоков управления осуществляется логическими операторами «И» и «ИЛИ».

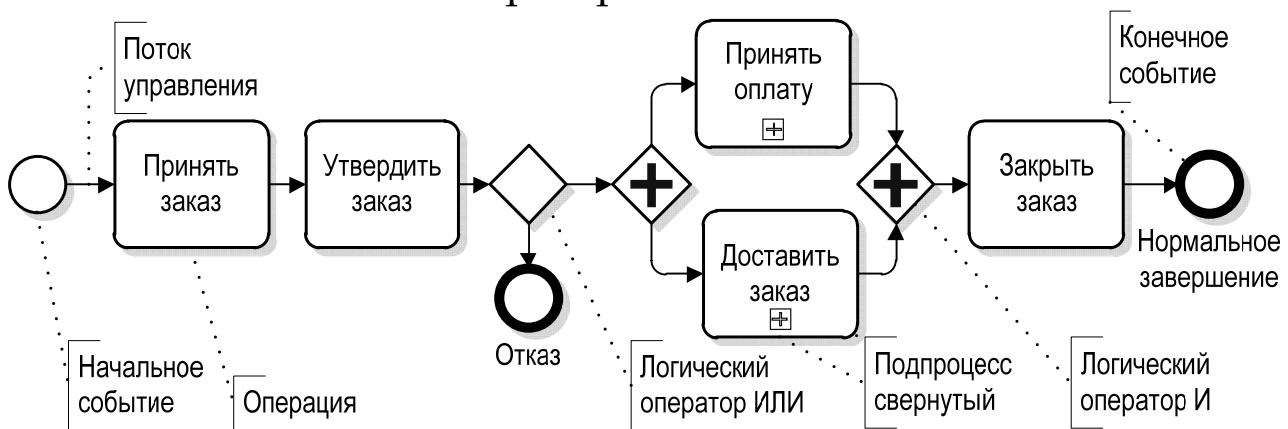


Рисунок 2-4. Использование основного набора элементов нотации

Пример (см. Рисунок 2-5) иллюстрирует использование описательного набора элементов нотации. Процесс ограничен пулом, он имеет название, которое обозначает название компании, в нашем случае это «автомагазин». Пул разделен на отдельные дорожки. Названия дорожек символизируют роли участников. Дорожки являются не обязательным элементом нотации, поскольку назначение исполнителей определяется для каждой задачи в отдельности. Однако они позволяют сделать схему процесса более наглядной и читаемой. Начальное и конечное события могут быть типизированными, в нашем примере это событие-таймер и отправка сообщения.

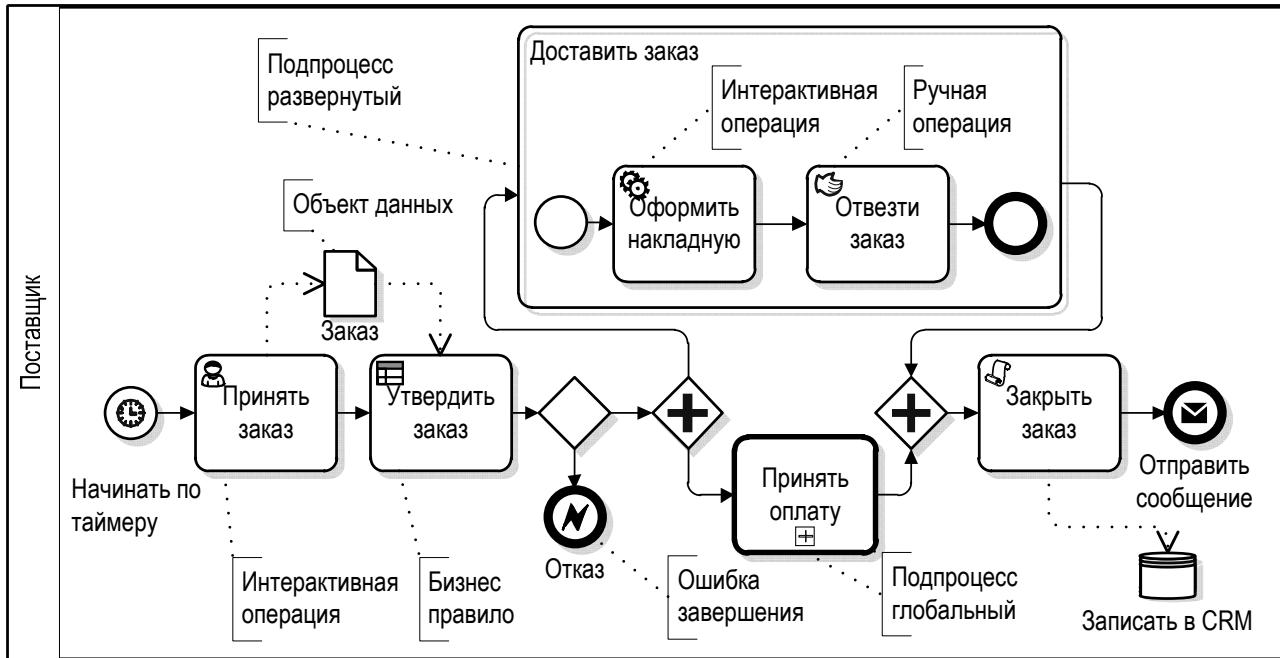


Рисунок 2-5. Основной набор элементов

2.5. КАТЕГОРИИ ДИАГРАММ БИЗНЕС-ПРОЦЕССОВ

Спецификация BPMN 2.0 регламентирует следующие типы диаграмм бизнес-процессов:

1. Диаграммы **оркестровки** (схемы потока работ), включая:
 - a. Модели **приватных** (внутренних) бизнес-процессов:
 - концептуальные схемы внутренних процессов;
 - исполняемые схемы внутренних процессов;
 - b. Модели **публичных** (внешних) процессов;
2. Диаграммы **взаимодействия** участников одного или нескольких бизнес-процессов (*Collaboration*);
3. Диаграммы **диалогов** (*Conversation*), которая помогает сгруппировать отдельные сообщения, которыми обмениваются участники по темам обсуждения;
4. Диаграммы **хореографии** (*Choreography*), которая описывает регламенты обмена сообщениями между участниками;

Все типы диаграмм могут быть объединены в рамках единой модели процесса. Рассмотрим эти категории бизнес-процессов.

2.5.1. ДИАГРАММЫ ОРКЕСТРОВКИ

Диаграмма оркестровки (orchestration) - это схема, показывающая очередность выполнения операций процесса. С позиции сервис-ориентированных технологий, такой бизнес-процесс может рассматриваться как набор вызовов веб-сервисов. Поэтому моделирование потоков работ иногда называют оркестровкой процесса, то есть координацией вызовов необходимых веб-служб. Схемы процессов можно разделить на закрытые и открытые, приватные (внутренние) и публичные (внешние).

2.5.1.1. ДИАГРАММЫ ЗАКРЫТОГО ПРОЦЕССА

Диаграмма закрытого процесса есть схема процесса, моделируемого внутри некоторого контейнера, называемого пулом (см. п. 9). Этот контейнер можно изображать на схеме явно или только подразумевать. Если пул явно указывается на диаграмме процесса, то схема называется *закрытой*. Пример (см. Рисунок 2-6) иллюстрирует закрытый процесс.

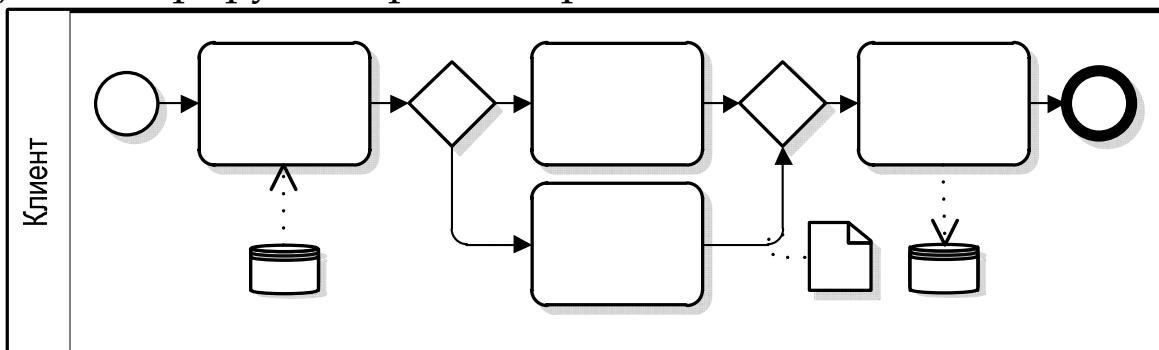


Рисунок 2-6. Закрытый процесс

2.5.1.2. ДИАГРАММЫ ОТКРЫТОГО ПРОЦЕССА

Диаграмма открытого процесса есть схема процесса, на которой пул явно не указывается, а только подразумевается. Пример (см. Рисунок 2-7) иллюстрирует схему открытого процесса.

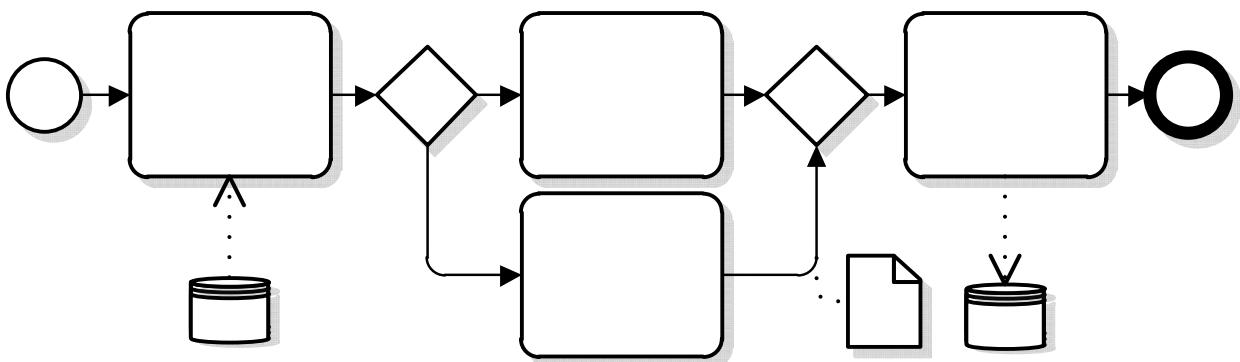


Рисунок 2-7. Открытый процесс (раскрытый пул)

В любом случае, в закрытом или открытом процессе поток управления (см. п. 1.13) не может пересекать границу пула, явно очерченную или только подразумеваемую.

2.5.1.3. ДИАГРАММА ПРИВАТНОГО ВЗАИМОДЕЙСТВИЯ

Диаграмма приватного взаимодействия (private) показывает процесс, который исполняется в пределах какой-либо организации или координируется из единого центра. Для неисполнимой аналитической модели это означает, что у процесса единый владелец. Например, схема процесса (см. Рисунок 2-6) является приватной.

2.5.1.4. АНАЛИТИЧЕСКАЯ МОДЕЛЬ ПРОЦЕССА

Аналитическая модель процесса (см. Рисунок 2-8) создается для документирования и формализации различных областей деятельности организации. На таких схемах, как правило, отсутствуют многие детали, которые не позволяют выполнить процесс в соответствии с описанием без дополнительного уточнения. На схеме изображены не все переходы, соединяющие операции процесса, прописаны не все условия логических операций, не определены исполнители задач и т.д.

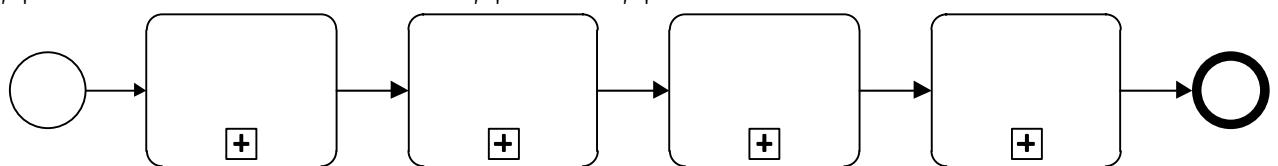


Рисунок 2-8. Пример аналитической схемы процесса

2.5.1.5. ИСПОЛНЯЕМАЯ МОДЕЛЬ ПРОЦЕССА

Исполняемая модель (см. п. 1.15) должна содержать всю информацию, необходимую для последующего выполнения процесса в специализированной программной среде BPMS. Она обязана показывать все возможные маршруты исполнения, иметь детализацию уровня элементарных действий, включать спецификацию всех правил, используемых для маршрутизации потоков, определять расписание исполнения процесса, определять исполнителей операций Пример (см. Рисунок 2-9) иллюстрирует исполняемую схему бизнес-процесса.

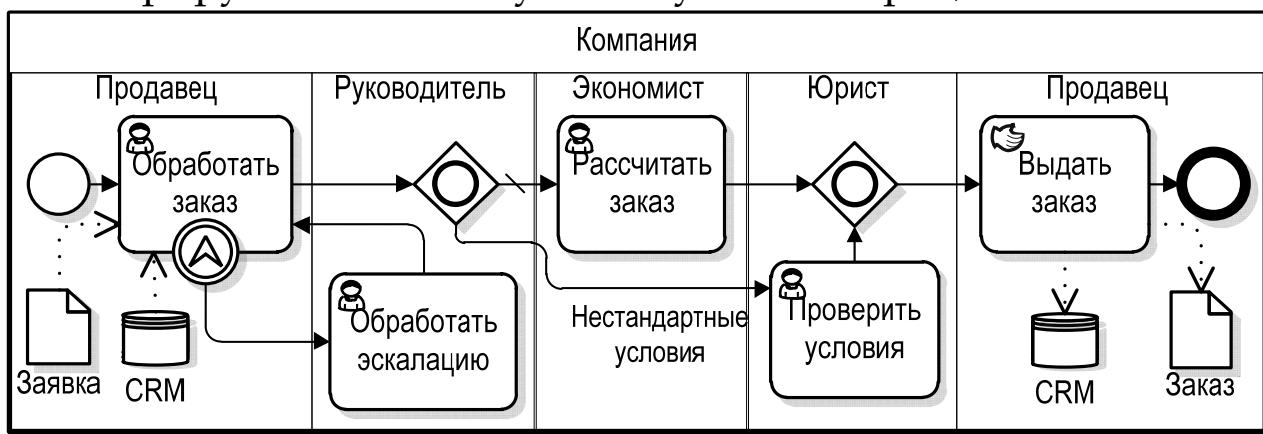


Рисунок 2-9. Пример исполняемой схемы процесса

2.5.1.6. ДИАГРАММА ПУБЛИЧНОГО ВЗАИМОДЕЙСТВИЯ

Диаграмма публичного взаимодействия (public) показывает коммуникацию между двумя приватными процессами, у каждого свой центр управления, каждый из них имеет своего владельца. Диаграмма публичного взаимодействия может иметь разную степень детализации.

Во-первых, процессы обоих участников взаимодействия могут моделироваться как «черный ящик», т.е. опускать все внутренние операции участников (см. Рисунок 2-10).

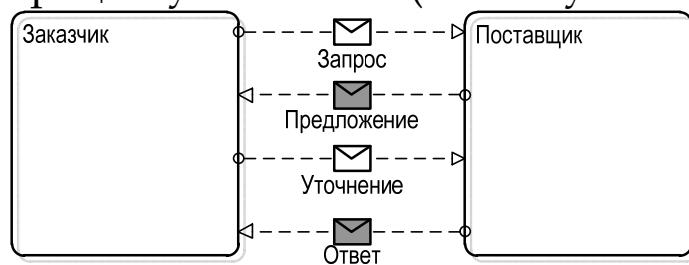


Рисунок 2-10. Публичное взаимодействие, закрытые процессы

Во-вторых, процесс одного из участников может моделировать операции, отвечающие за коммуникацию между участниками. Пример (см. Рисунок 2-11) показывает взаимодействие компании и ее корпоративного клиента. Действия компании расписаны не полностью, показаны только те задачи, которые видны второму участнику взаимодействия, а действия клиента не показаны вообще. Модель позволяет показать потоки сообщений, которыми обмениваются оба участника.

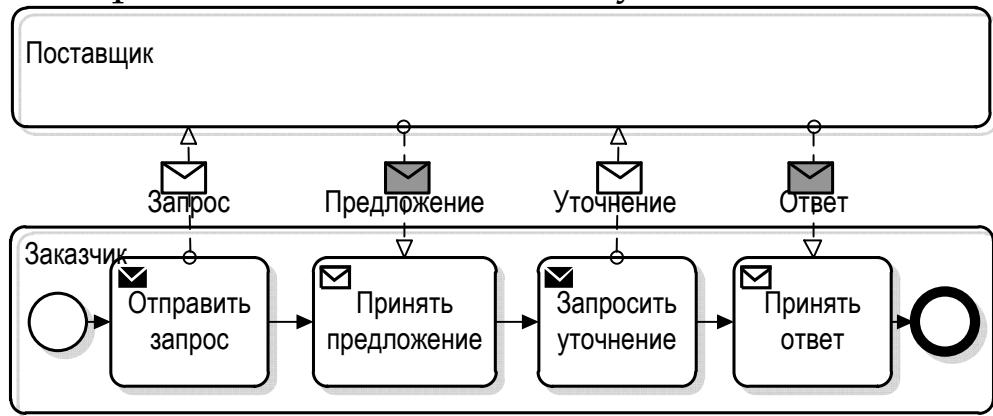


Рисунок 2-11. Публичное взаимодействие, один открытый процесс

Наконец, действия обоих участников м.б. изображены на диаграмме. Пример (см. Рисунок 2-12) показывает действия Заказчика и Поставщика, потоки сообщений, которыми обмениваются оба участника. Примечательно, что схема показывает только операции, видимые участникам информационного обмена. Остальные операции не изображаются.

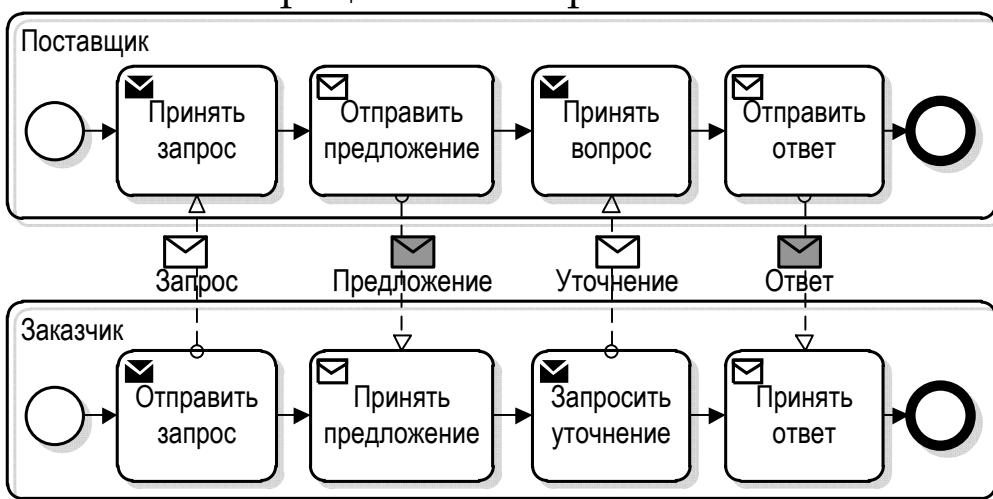


Рисунок 2-12. Публичное взаимодействие, открытые процессы

Следует особо подчеркнуть две особенности диаграммы публичного взаимодействия:

- Во-первых, оба участника взаимодействия являются юридическими лицами, которые осуществляют свою работу в соответствии со своими собственными внутренними бизнес-процессами. Неверно представлять одного из участников физическим лицом, которое не имеет своего собственного внутреннего бизнес-процесса или не обязано пользоваться BPMS системой Заказчика.
- Во-вторых, обмен информацией между участниками осуществляется через поток сообщений (см. 6.5), а не через абстрактный информационный поток, например, электронную почту и т.п.

2.5.1.7. МОДЕЛИРУЕМЫЕ И НЕМОДЕЛИРУЕМЫЕ ДЕЙСТВИЯ НА ДИАГРАММЕ ВЗАИМОДЕЙСТВИЯ

Возможна ситуация, когда в рамках единой модели один лист диаграммы изображает процесс публичного взаимодействия, а на другом листе этот же процесс описывается как приватный. Возникает коллизия, обе модели изображают один процесс, но с разной степенью детализации: публичный процесс не до конца определен, в нем отсутствуют некоторые детали, которые не важны с точки зрения обмена сообщениями, тогда как соответствующий ему приватный процесс определен полностью и во всех деталях.

Более того, модели могут отличаться и по существу. Пример (см. Рисунок 2-13) показывает две схемы: диаграмму публичного взаимодействия (А) и эквивалентную ей приватную (Б). Публичный процесс содержит две операции отправить и принять сообщение. Приватный процесс для обмена сообщениями использует сигналы, он включает операции, которых в публичном процессе нет. Принято говорить, что приватный процесс поддерживает публичный, если он способен воспроизвести требуемое поведение.

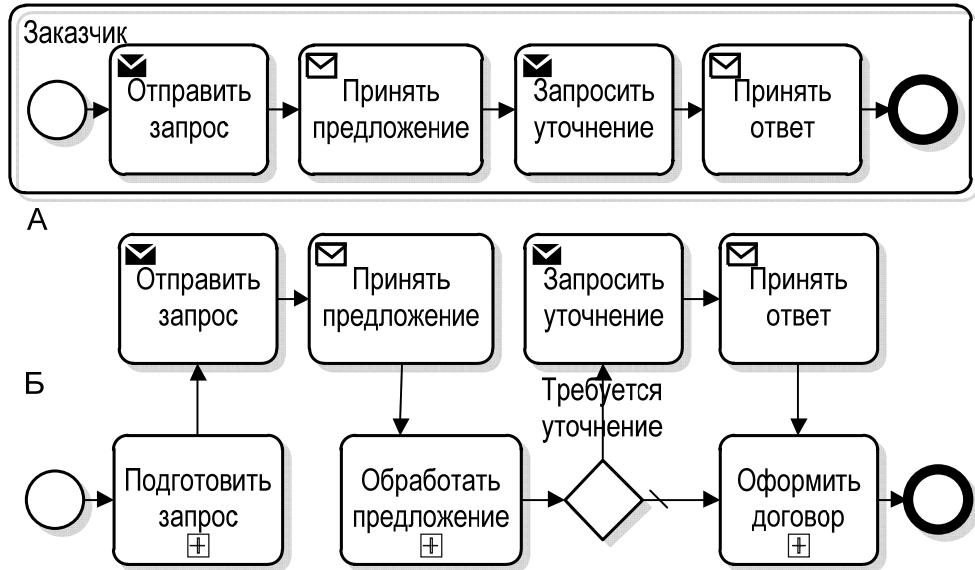


Рисунок 2-13. Эквивалентные публичный (А) и приватный (Б) процессы

Приватный процесс, поддерживающий публичный, не обязан полностью его копировать. Необходимо лишь, чтобы во время исполнения экземпляры одного процесса вели себя также, как если бы они были экземплярами другого. На диаграмме публичного процесса допускается появление немоделируемых действий, которые специфицированы в поддерживающем его приватном процессе. Т.о. модель публичного процесса выглядит как неполностью детализированная схема соответствующего ей приватного.

2.6. СХЕМЫ ВЗАИМОДЕЙСТВИЯ

Представим себе, что два процесса исполняются асинхронно и независимо друг от друга, но иногда они должны взаимодействовать и обмениваться информацией. **Схемы взаимодействия** (Collaboration) показывают обмен сообщениями между двумя и более участниками. Участники по отдельности управляют своими бизнес-процессами, у каждого есть свой владелец.

Для изображения участников на схеме взаимодействия обычно используют графический элемент *пул* (см. Рисунок 2-14). Взаимодействие между участниками осуществляется при помощи потоков сообщений. Следует учитывать, что схема определяет лишь наличие потока сообщений, но не детализирует его содержимое. На схеме показываются только операции, ко-

торые участвуют в обмене сообщениями. Мы помним, что соответствующие схемы внутренних процессов содержат, как правило, намного больше информации. Пул, изображающий участника, может быть пустым, не иметь внутри себя ни одной операции. В этом случае, такой процесс называется «черный ящик». Потоки сообщений могут изображаться только между пулами.

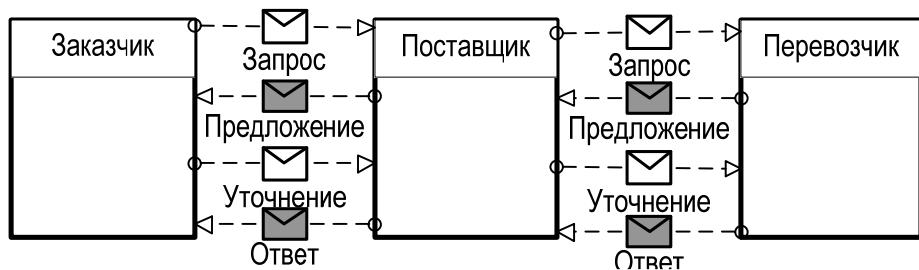


Рисунок 2-14 Моделирование обмена сообщениями между участниками

2.7. ХОРЕОГРАФИЯ ПРОЦЕССОВ

Схема хореографии (без пулов, без оркестровки) показывает последовательность процедур обмена сообщениями между двумя и более участниками (см. Рисунок 2-15). В отличие от обычного процесса, который обязательно изображается внутри пула, хореография обычно описывается без пулов.

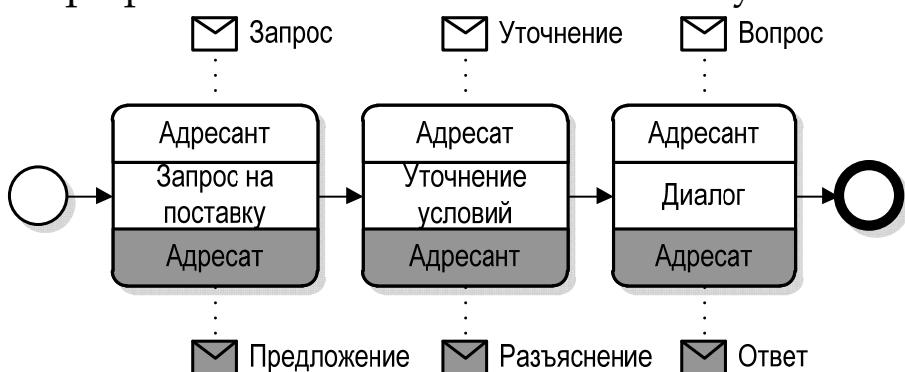


Рисунок 2-15. Схема хореографии.

В целом, хореография похожа на схему приватного процесса, поскольку она так же состоит из цепочки задач хореографии, событий и логических операций. Тем не менее, хореография отличается от схемы оркестровки тем, что операции на ней отображают факт передачи сообщения между двумя и более участниками.

2.7.1. СХЕМА ДИАЛОГОВ

Схема диалога позволяет отобразить информационный обмен сообщениями, сгруппированными по темам обсуждения. Например, заказ товаров, перевозка грузов, выписка счета м.б. темой диалога между поставщиком и покупателем. По сути, такие схемы изображают различные сценарии общения участников между собой. Рассмотрим пример из сферы логистики. Процесс «пополнение запасов» может быть описан следующим сценарием: создается заказ, назначается перевозчик товара, транспортировка, оплата, доставка. Таким образом, на схеме диалога (см. Рисунок 2-16) шестиугольники отображают обмен сообщениями между пулами – участниками процесса. Мы видим процесс с «высоты птичьего полета», однако все необходимые детали на нем присутствуют.

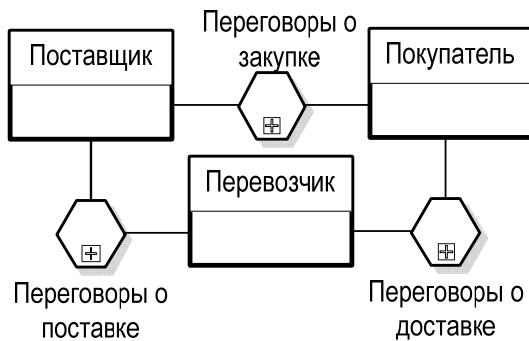


Рисунок 2-16. Схема диалога участников процесса

2.8. СХЕМАМЫ ОРКЕСТРОВКИ, ДИАЛОГОВ ХОРЕОГРАФИИ

Рисунок 2-17 иллюстрирует соотношение между схемами оркестровки, диалогов и хореографии

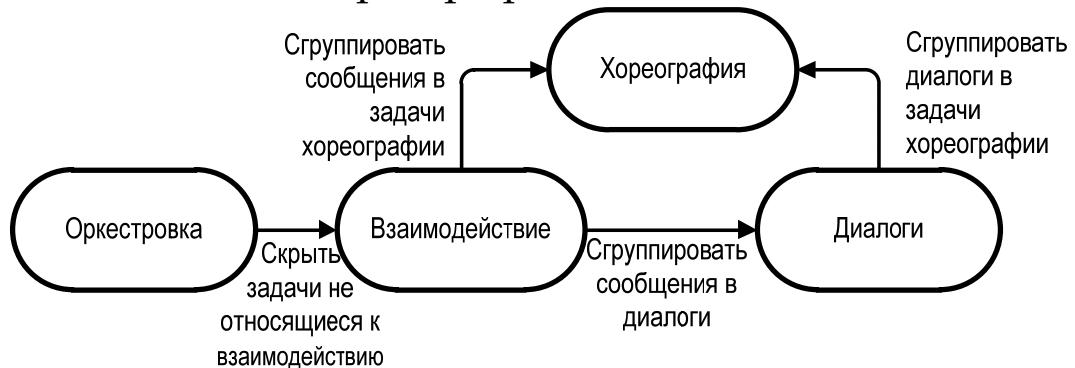


Рисунок 2-17. Схемы оркестровки, диалогов и хореографии

3.ОПЕРАЦИИ

Операции процесса бывают элементарными и составными. Элементарные операции м.б. поручены одному исполнителю. Составные операции подразумевают декомпозицию на составляющие их действия низшего порядка. Как правило, детализация составной операции изображается на отдельной диаграмме процесса. Этот тип элементов называется **подпроцесс**.

Для изображения операций в нотации BPMN используются различные графические элементы. Выделяют несколько типов: операция, подпроцесс, глобальная (вызывающая) операция, транзакция (см. Рисунок 3-1).



Рисунок 3-1. Типы операций в нотации BPMN 2.0

Под **операцией** будем понимать единицу работы, изображенную на схеме процесса. Операции - главная составляющая любого процесса.

Подпроцесс это группа логически связанных операций. Подпроцесс может включать одну или несколько связанных потоками управления операций (см. 3.5)

Глобальная (вызывающая) операция обозначает вызов повторно используемого, глобально известного подпроцесса (см. 3.5.2).

Транзакцией называется подпроцесс, который может быть выполнен либо целиком, либо не выполнен вообще. В случае, если транзакция не завершена, возникает необходимость вернуть все измененные данные в состояние, предшествовавшее началу выполнения транзакции.

3.1. ВИДЫ ОПЕРАЦИЙ

В нотации BPMN знак операции процесса может относиться к одной из семи видов задач, для каждой предусмотрен отдель-

ный маркер, позволяющий визуально различать операции. Таблица 3-1 показывает маркеры различных видов операций.

Таблица 3-1. Виды операций

Тип операции	Обозначение
Интерактивная (Пользовательская)	
Ручная	
Автоматическая (Сервис)	
Сценарий	
Бизнес-правило	
Отправка сообщения	
Получение сообщения	
Нетипизированная, абстрактная	нет

3.1.1. ИНТЕРАКТИВНАЯ ОПЕРАЦИЯ

Интерактивная операция исполняется в среде BPMS, она создает *пользовательское задание* для участников бизнес-процесса, которое появляется в рабочей области соответствующего исполнителя на BPMS портале. С интерактивной операцией связана соответствующая экранная форма, которую видит участник.

Когда стартует выполнение интерактивной операции, для нее необходимо назначить исполнителя. Процедура выполняется в два этапа, отбор потенциальных исполнителей, выбор актуального исполнителя (см. п. 9). После того как исполнитель завершает работу, управление передается следующей по порядку операции.

3.1.2. РУЧНАЯ ОПЕРАЦИЯ

Ручная операция отличается от пользовательской тем, что сотрудники выполняют работу вне рамок BPMS или иной ИС. Например, получают задание «выкопать яму», «перенести коробки», «приготовить обед» и т.д. В этом случае, в рабочей области портала соответствующий исполнитель только проставляет отметку о факте выполнения ручной операции.

3.1.3. АВТОМАТИЧЕСКАЯ ОПЕРАЦИЯ

Автоматическая операция выполняются без участия людей. Как правило, в ней реализуется вызов внешних, по отношению к BPMS-среде служб, информационных систем, например веб-сервисов.

Вначале входные данные операции сопоставляются с аргументами соответствующего сервиса. Затем происходит вызов сервиса. В конце выходные аргументы сервиса отображаются на выходной набор данных операции.

3.1.4. ОПЕРАЦИЯ СЦЕНАРИЙ

Операция сценарий содержит в себе простейший программный код, исполнение которого не требует внешних информационных ресурсов. Язык написания сценария зависит от используемой BPMS среды и никак не специфицируется BPMN.

3.1.5. ОПЕРАЦИЯ БИЗНЕС-ПРАВИЛО

Операция бизнес-правило помечает задачу, в которой проверяется значения параметров, определяющих или ограничивающих какие-либо стороны бизнеса. *Бизнес-правила* можно менять в ходе исполнения и, таким образом, в режиме реального времени управлять логикой выполнения текущих и вновь запускаемых экземпляров процесса. Например, банк определил, что если сумма кредита не превышает 100 условных единиц, то проверка заемщика службой безопасности не требуется. В любой момент банк вправе поменять пороговое значение на 50, и тогда все экземпляры процессов, выполнение которых еще не дошло до точки принятия решения, у которых сумма кредита более порогового значения, попадут в службу безопасности. Изменение схемы процесса при этом не потребуется. *Операция бизнес-правило* часто используется совместно с *логическими операторами*. При этом бизнес-правило проверяет истинность некоторого условия, а логический оператор осуществляет маршрутизацию потока управления.

3.1.6. ОПЕРАЦИЯ ОТПРАВКА И ПОЛУЧЕНИЕ СООБЩЕНИЯ

Операции отправки и получения сообщений предназначены для осуществления межпроцессного взаимодействия. Более подробное они будут рассмотрены ниже (см. п. 6.4.). Отправка сообщения осуществляется следующим образом: когда поток управления достигает операции отправить сообщение (см. Рисунок 3-2А) последняя генерирует информационное оповещение. Что бы указать, кто является получателем, используется поток сообщений – пунктирная линия, соединяющая отправителя и получателя. После этого управление передается на следующую операцию.

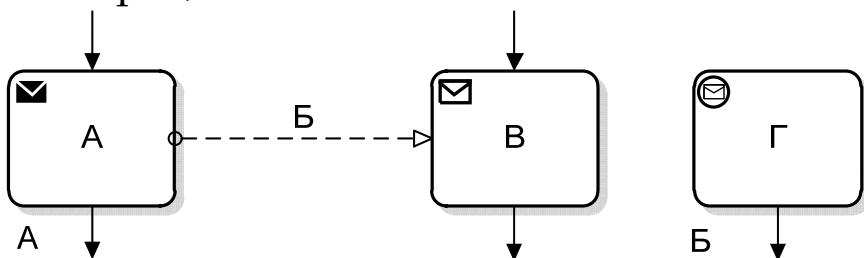


Рисунок 3-2. Операции отправки и получение сообщения

Когда поток управления достигает операции принять сообщение (см. Рисунок БВ), последняя переходит в режим ожидания. Только после получения сообщения, выполнение этой операции завершается, а управление передается на следующую операцию. Следует отметить, что сообщение может переносить полезное информационное наполнение.

Операция получения сообщения может использоваться для того, что бы стартовать исполнения процесса. Для этого используется специальная операция получить сообщение и стартовать процесс, она не может иметь входящий поток управления, но только исходящий.

3.1.7. АБСТРАКТНАЯ ОПЕРАЦИЯ

Абстрактная (нетипизированная) операция, не имеет реального наполнения. Сразу по получению управления она считается исполненной, управление передается следующему заданию. Она может использоваться в аналитическом моделировании.

нии, когда не ясен окончательно тип моделируемой операции. В дальнейшем, при переходе к исполняемой модели, тип операции уточняется и указывается на схеме процесса.

3.2. МАРКЕРЫ ОПЕРАЦИИ

Операция является базовым графическим элементом нотации, она допускает несколько способов исполнения. Для того чтобы указать способ исполнения операций используются **маркеры**. Они размещаются поверх графического элемента, добавляя тем самым определенную семантику. Нотация BPMN 2.0 регламентирует шесть видов маркеров, в т.ч. три маркера для изображения разных видов циклов: обычный цикл; параллельное выполнение, параллельное исполнение; последовательное выполнение, компенсация, операция «по случаю» (Ad-Hoc).

Таблица 3-2. Маркеры операций

Маркер	Тип операции
	Подпроцесс
	Цикл
	Параллельное исполнение
	Последовательное исполнение
	Компенсация
	По-случаю (Ad-Hoc)

3.2.1. МАРКЕР ПОДПРОЦЕССА

Маркер подпроцесса сигнализирует аналитику, что тот имеет дело с составной операцией – подпроцессом, который может быть декомпозирован на отдельные операции. Последние, в свою очередь, могут быть составными. Глубина вложности составных операций не ограничивается. Составная операция мо-

жет быть свернутой или раскрыта. Развёрнутый подпроцесс детализуется на отдельной странице диаграммы.

3.2.2. МАРКЕР ЦИКЛА

Маркер цикла означает последовательное, циклическое исполнение операции. Для цикла можно задать условие повторения: «while...do», «repeat...until». Число повторений и способ исполнения определяются встроенными атрибутами операции и могут гибко перестраиваться. Таблица 3-3 объясняет значения атрибутов:

Таблица 3-3. Атрибуты циклического исполнения операции

Атрибут	Семантика исполнения
loopCounter	Счетчик итераций, целое число, определяющее количество циклов повторения.
testBefore	Флаг, который определяет, когда проводить проверку: Если testBefore = ИСТИННО, в начале, Если testBefore = ЛОЖНО, то в конце итерации
loopMaximum:	Максимальное количество итераций (повторений)
completionCondition	Условие завершения, результат типа Boolean: Если условие ИСТИННО, все еще не завершенные экземпляры будут завершены;

3.2.3. МАРКЕР ПАРАЛЛЕЛЬНОГО ВЫПОЛНЕНИЯ

Маркер параллельного выполнения означает одновременный, параллельный запуск нужного числа экземпляров операции. Он применяется, когда аналитик на этапе моделирования не знает точное количество экземпляров в момент исполнения. Например, описывая электронное взаимодействие с контрагентами, мы заранее не знаем их числа, более того, разные экземпляры одного процесса могут вызывать разное число контрагентов. Число контрагентов, участвующих в процессе станет известно на этапе исполнения. Вычисление числа экземпляров выполняется только один раз, будет запущено нужное число экземпляров операции. Число повторений и семантика исполнения определяются встроенными атрибутами опе-

рации. Аналитик может явно указать число экземпляров или указать условие, при которых будет создан еще один экземпляр процесса. Соответствующие атрибуты определяют условия координации и завершения запущенных множественных экземпляров процессов. Таблица 3-4 объясняет назначение атрибутов:

Таблица 3-4. Семантика параллельного исполнения

Атрибут	Семантика исполнения
isSequential	Если ИСТИННО, то последовательное исполнение; Если ЛОЖНО, то параллельное исполнение.
loopCardinality	Количество экземпляров (повторений)
loopDataInputRef	Указатель на массив данных
completionCondition	Условие завершения, результат типа Boolean: Если условие ИСТИННО, все еще не завершенные экземпляры будут завершены;
MultinstanceBehavior: {None One All Complex}	Определяет способ координации завершения параллельных экземпляров: <ul style="list-style-type: none"> None: Маркеры потока управления создаются после завершения каждого экземпляра. One: Один маркер потока управления создается после завершения первого экземпляра. All: Один маркер потока управления создается после завершения всех экземпляров. Complex: Комплексное условие определяет порядок создания маркеров потока управления.

Как видно из таблицы, маркер параллельного исполнения, если установлено соответствующее значение атрибута **isSequential**, допускает последовательное исполнение (см. п. 3.2.4).

Если выбран вариант параллельного исполнения, то нужное число экземпляров процесса запускается единовременно. Если выбран вариант последовательного исполнения, то следующий экземпляр стартует только после завершения предыдущего.

Часто операция с параллельным исполнением обрабатывает коллекцию, интерпретируемую как массив данных. Каждый стартуемый экземпляр работает со своим элементом этого

массива. В этом случае, число запущенных экземпляров определяется количеством элементов в массиве.

3.2.4. МАРКЕР ПОСЛЕДОВАТЕЛЬНОГО ВЫПОЛНЕНИЯ

Маркер последовательного выполнения отображает циклическое выполнение множества экземпляров. Работа организуется точно так же, как в обычном цикле. Для цикла можно также задать условие вида: «while...do», «repeat...until».

Число повторений и способ исполнения определяются встроенными атрибутами операции, так же как для параллельного исполнения. Обратим внимание, что маркер последовательного исполнения, если установлено соответствующее значение атрибута **isSequential**, означает параллельное исполнение (см. 3.2.3), соответственно изменится маркер активности..

3.2.5. МАРКЕР AD-HOC ОПЕРАЦИИ

Операция для случая (ad-hoc) обозначает решение, предназначенное для конкретной проблемы или ситуации. Маркер «*ad-hoc*» используется для обозначения сложного действия, которое состоит из набора вложенных операций, каждая предназначено для решения своей задачи, причем они не обязательно связаны между собой потоками передачи управления. Пользователь может выбрать подходящую операцию на этапе исполнения. Последовательность исполнения операций и количество их повторений не регламентируются и определяются пользователем в ходе работы. Данный маркер полезен при моделировании неструктурированных процессов. Пример (см. Рисунок 3-3) иллюстрирует «*ad-hoc*» операцию «подготовка командировки». Пользователь сам выбирает нужные ему действия. Он может вначале получить командировочные, а затем забронировать гостиницу или может выполнить действия в другом порядке.



Рисунок 3-3. Ad-hoc операции

3.2.6. ОПЕРАЦИЯ КОМПЕНСАЦИЯ

Операция компенсация используется для описания логики отмены действий, выполненных в некоторой ранее завершенной операции. Пусть несколько операций образуют транзакцию. Часть операций, составляющих транзакцию, уже успешно завершились, однако какая-то из оставшихся завершилась отказом. Это означает, что транзакция не выполнена, т.ч. ранее завершенные операции надо отменить, для этого следует откатить измененные в них данные в состояние, предшествовавшее началу выполнения транзакции. Например, мы начали бронировать билеты, заблокировали средства на счете клиента, однако выяснилось, что бронирование отменяется, средства надо разблокировать. Разумеется, компенсацию можно определить не для всех типов операций.

Пример (см. Рисунок 3-4) иллюстрирует *операцию компенсации*. Операция А есть составная часть некоторой транзакции. Операция А имеет прикрепленное *событие компенсация*, которое с помощью *направленной ассоциации* указывает на операцию, выполняющую компенсацию. Если транзакция не выполнена, то будет выработан сигнал, который будет перехвачен прикрепленным событием, последнее запустит операцию компенсация, которая откатит данные в исходное состояние.

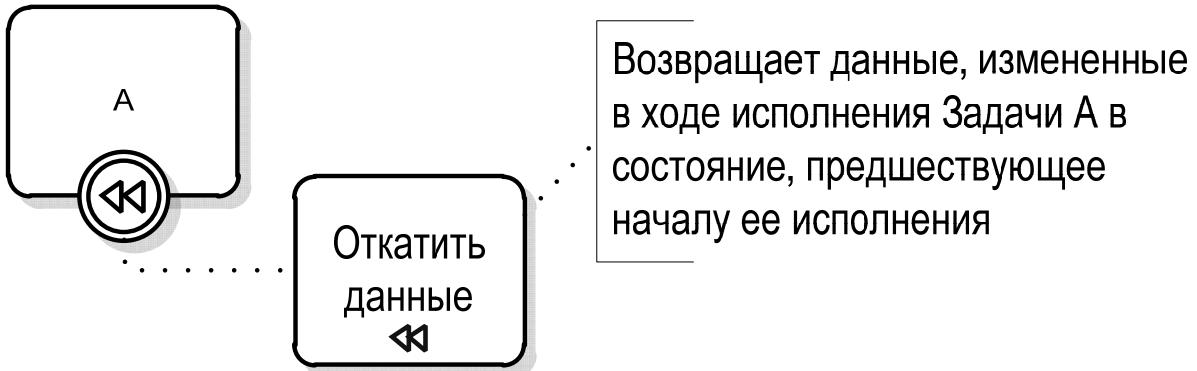


Рисунок 3-4. Операция компенсация

Следует обратить внимание, стрелка *направленной ассоциации* не обозначает переход управления, она лишь указывает операцию, выполняющую откат данных.

3.2.7. КОМБИНАЦИИ МАРКЕРОВ ПОДПРОЦЕССА

Спецификация BPMN определяет одиннадцать различных вариантов комбинаций маркеров операции, которые могут использоваться с обычными подпроцессами.

1. Свернутый подпроцесс;
2. Свернутый подпроцесс, реализующий компенсацию;
3. Свернутый Ad-Hoc подпроцесс, содержащий набор действий, выполняемых по случаю.
4. Циклическое выполнение подпроцесса;
5. Циклическое выполнение подпроцесса, каждая итерация выполняет компенсацию;
6. Циклическое выполнение Ad-Hoc подпроцесса;
7. Циклическое выполнение Ad-Hoc подпроцесса с компенсацией. Подпроцесс Ad-Hoc итеративно повторяется;
8. Параллельное выполнение подпроцессов;
9. Параллельное выполнение подпроцессов, выполняющих компенсацию;
10. Параллельное выполнение Ad-Hoc подпроцессов;
11. Параллельное выполнение Ad-Hoc подпроцессов, каждый экземпляр выполняет компенсацию

Таблица 3-5 показывает допустимые варианты комбинации маркеров (маркеры параллельного и последовательного исполнения не различаются).

Таблица 3-5. Комбинации маркеров подпроцесса

	Подпроцесс	Компенса- ция	Ad-Hoc	Компенсация + Ad- Hoc
Подпроцесс	1	2	3	Нет
Циклическое исполнение	4	5	6	7
Параллельное исполнение	8	9	10	11

3.3. ГРУППА ОПЕРАЦИЙ

Группа операций позволяет визуально объединить несколько задач. Группа никоим образом не оказывает влияния на исполнение процесса. На схеме процесса группа отображается прямоугольником, ограниченным штрихпунктирной линией. Группа может пересекать границы пула, объединяя операции, принадлежащие разным процессам (см. Рисунок 3-5). Объединяя несколько задач в группу, мы можем, к примеру, комментировать их.

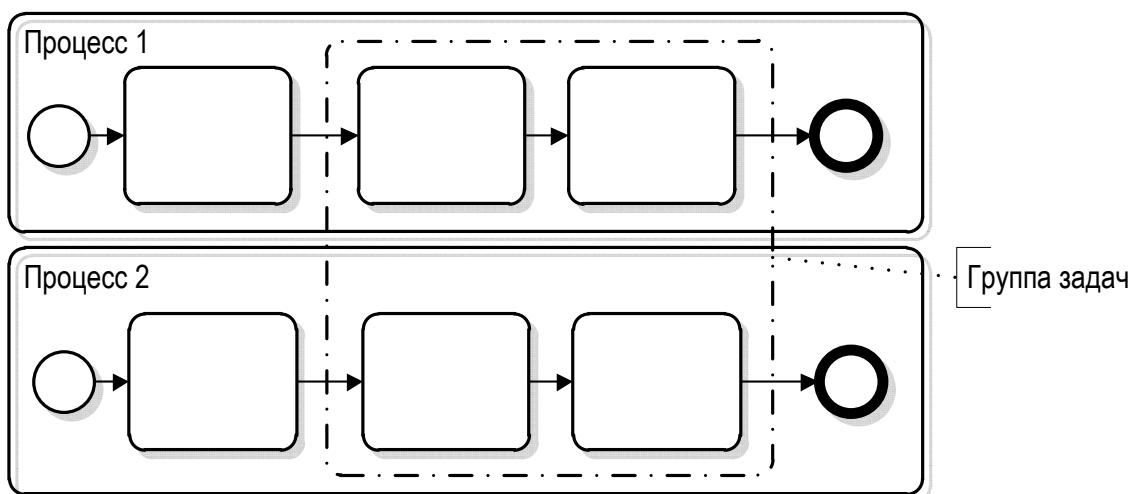


Рисунок 3-5. Группа операций

3.4. МОДЕЛЬ ПРОЦЕССА

Модель процесса есть совокупность логически связанных работ, изображаемых на схеме операциями, инициируемая начальным событием, и завершающаяся конечным событием. Процесс всегда располагается внутри пула. В некоторых случаях пул может не отображаться, в последнем случае он присутствует неявно.

Исполняемые и аналитические модели процессов могут иметь разную степень детализации и использовать разные подмножества элементов нотации (см. п. 2.4). Спецификация различает следующие классы моделей:

- Концептуальные модели - определяют структуру моделируемого процесса, основные этапы его выполнения, их причинно-следственные связи. Это есть предварительное, приближенное представление о рассматриваемом процессе, оно включает абстракции высокого уровня, используя для этого ограниченное множество элементов нотации.
- Аналитические модели служат средством для выявления бизнес проблем исследуемого процесса. Они не предназначены для последующего исполнения, поэтому могут опускать некоторые детали, например, показывать только наиболее вероятные сценарии исполнения, опуская редко используемые, не включать детализацию действий низкого уровня. Эти модели фокусируются на визуальном представлении бизнес-процесса. Они используют расширенный набор элементов, который наиболее часто используют бизнес аналитики, но используют элементы, в которых программируется семантика исполнения.
- Исполняемые модели предназначены для запуска в среде BPMS. Они покрывают все допустимые сценарии исполнения, даже редко используемые, без которых работа системы окажется невозможной, содержат детализацию элементарных действий, определяют участников, исполнителей и их права доступа к объектам системы. Исполняемые модели включают полный набор элементов нотации, в том

числе такие, где явно задается семантика исполнения.

3.5. ПОДПРОЦЕСС

Часто *сквозные* процессы представляются как цепочка взаимодействующих подпроцессов. **Подпроцесс** это последовательность логически связанных работ, имеющая начало и конец, выполняющая законченное действие. Подпроцессы м.б. двух видов: вложенными (*embedded*) и повторно используемыми, глобально известными (*global*).

Подпроцесс запускается, если произошло событие, указанное в качестве стартового. Он завершается после того, как завершились все существующие в нем потоки управления.

3.5.1. ВЛОЖЕННЫЙ ПОДПРОЦЕСС

Вложенный подпроцесс есть часть родительского процесса, он определен и известен только локально и не может использоваться в других проектах. Вложенный подпроцесс изображается на одном листе модели с родительским процессом верхнего уровня (см. Рисунок 3-6). Он может изображаться как в свернутом, так и в развернутом виде. В первом случае, он выглядит как операция, у которой добавлен маркер подпроцесса (Рис. А), во втором случае, она представляет отдельный процесс, составленный из операций, логических операторов, событий и знаков перехода управления (Рис. Б). Вложенный подпроцесс в качестве стартового события, запускающего исполнение, использует только одно обычное нетипизированное событие. Поскольку процесс изображается в рамках схемы родительского, он не может изображать пулы и дорожки.

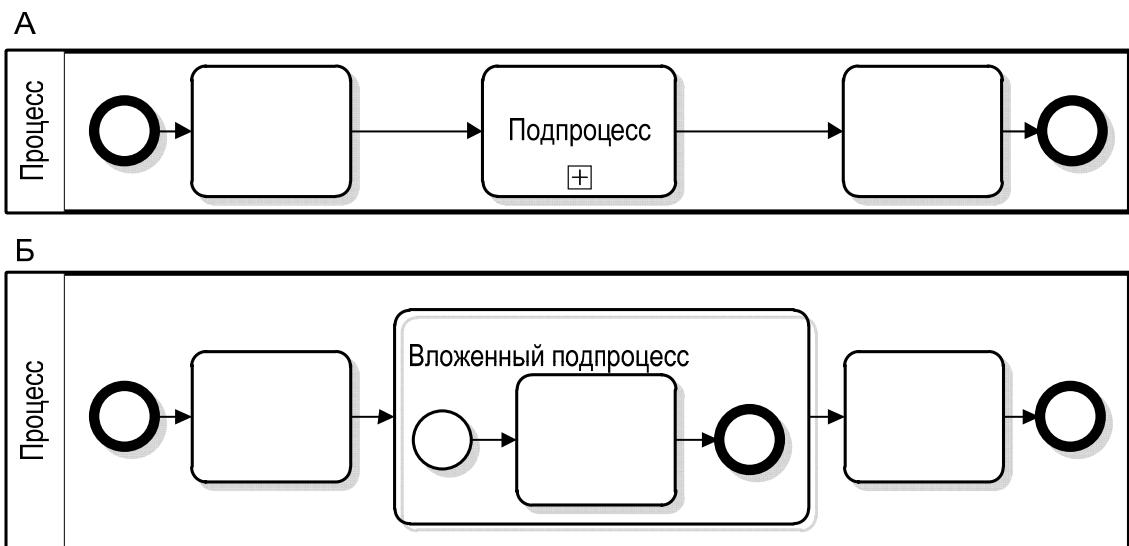


Рисунок 3-6. Развёрнутый вложенный подпроцессы

Объекты данных, которые определены в родительском процессе, автоматически доступны операциям вложенного подпроцесса. А после его завершения, все сделанные изменения в данных, будут видны в родительском процессе. Пример (см. Рисунок 3-7) иллюстрирует область действия переменных вложенного процесса.

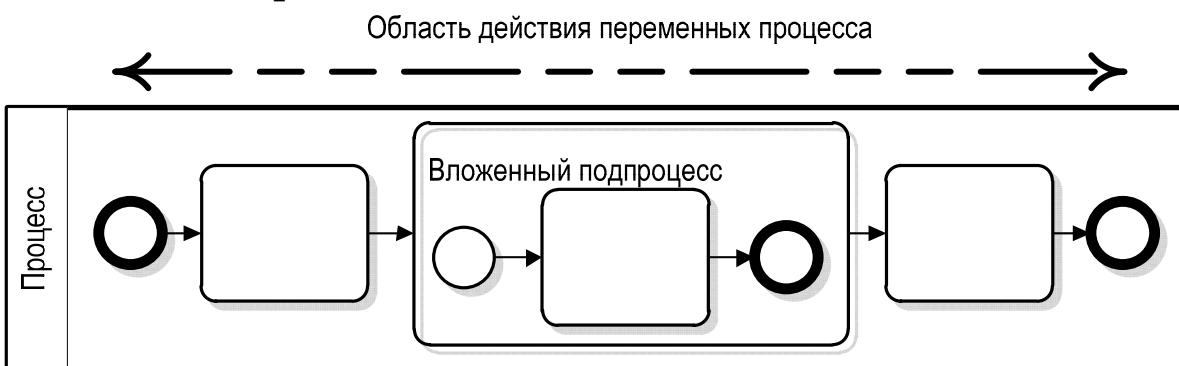


Рисунок 3-7. Область действия переменных вложенного процесса

Развёрнутый подпроцесс изображает группу из нескольких, графически не связанных потоком управления задач. Эта форма записи показывает, что задачи, входящие в подпроцесс исполняются параллельно и независимо друг от друга. Пример (см. Рисунок 3-8) показывает две операции Б и В, помещенные внутри вложенного подпроцесса. Когда операция А будет выполнена, стартует подпроцесс, обе операции Б и В будут запущены одновременно и выполняются параллельно.

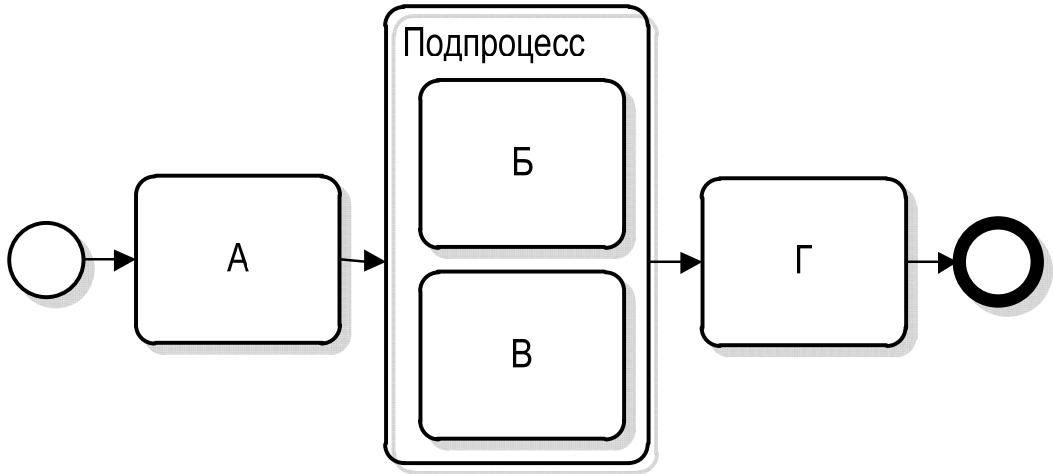


Рисунок 3-8. Подпроцесс с параллельно исполняемыми заданиями

Эта форма записи не предполагает указания стартового или конечного событий подпроцесса, не показывает последовательность потоков управления.

3.5.2. ПОВТОРНО ИСПОЛЬЗУЕМЫЙ ПОДПРОЦЕСС

Повторно используемый (внешний) подпроцесс это отдельный модуль, который известен глобально и может многократно использоваться путем вызова из разных процессов. В спецификации BPMN 1.0 этот тип подпроцессов назывался *независимый* подпроцесс.

Повторно используемый подпроцесс разрабатывается полностью независимо от вызывающего его процесса. Он представляет некоторый сервис, который доступен для многократного повторного использования, путем вызова из разных процессов. Повторно используемый подпроцесс может казаться визуально не отличим от обычного вложенного, отличия заключаются в способе его запуска.

Повторно используемый подпроцесс может начинаться и заканчиваться только простыми, нетипизированными событиями старта и завершения. Он не может начинаться со стартового события *сообщение* и *таймер*. Однако следует учесть, что подпроцесс может использоваться двояко: в одной ситуации как глобальный, а в другой ситуации выступать в качестве процесса верхнего уровня, при этом он может иметь дополнительные стартовые и конечные события.

тельные возможности старта, в т.ч. с использованием *событий* сообщений и *таймеров*. Пример (см. Рисунок 3-9) показывает повторно используемый подпроцесс, который имеет два набора стартовых событий. Старт по сообщению используется, когда подпроцесс используется как встроенный, а простое нетипизированное событие старт, когда подпроцесс используется как повторно используемый.

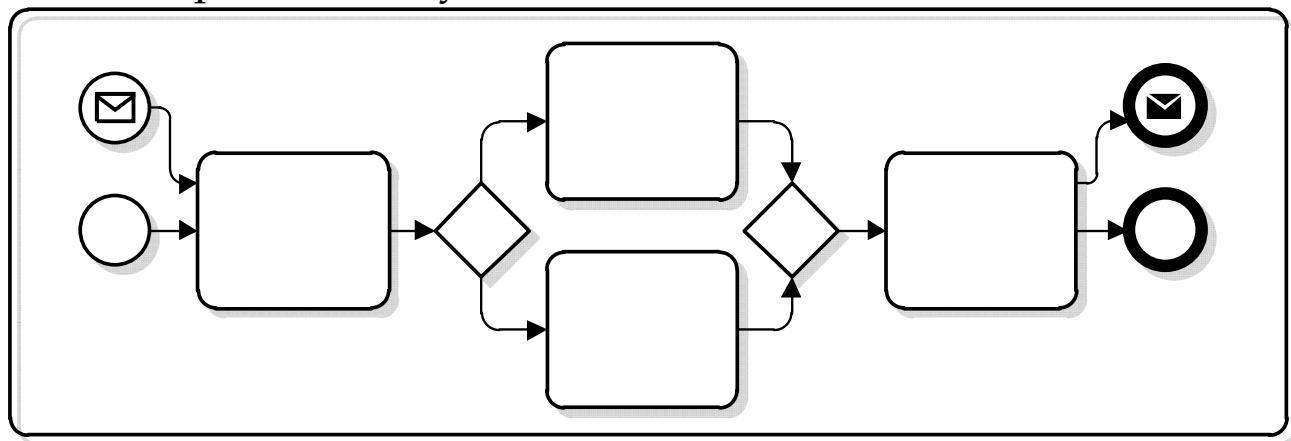


Рисунок 3-9. Подпроцесс «двойного» назначения

Когда возникает необходимость вызвать внешний повторно используемый компонент, следует использовать *вызывающую операцию* (Call Activity). В результате исполнения этой операции управление передается в вызываемый подпроцесс, а после его окончания снова вернется в родительский.

Поскольку модель внешнего подпроцесса разрабатывается на отдельном листе схемы, она может включать пулы и дорожки.

3.5.3. ВЫЗЫВАЮЩАЯ ОПЕРАЦИЯ

Вызывающая операция (Call Activity) используется, когда возникает необходимость вызвать внешний по отношению к данной модели повторно используемый процесс. Повторно используемый компонент д.б. глобально известен. Спецификация определяет, четыре типа операций, которые могут выступать в качестве *глобальных*:

- Пользовательская операция
- Ручная операция
- Операция сценарий

- Операция бизнес-правило

Глобальная операция визуально отличима благодаря границе, изображаемой более толстой линией. Пример (см. Рисунок 3-10) иллюстрирует вызов глобальной пользовательской операции.

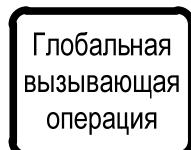


Рисунок 3-10. Глобальная пользовательская операция

Глобально известный, повторно используемый подпроцесс моделируется на отдельной схеме вне рамок данного процесса. Вызывающая операция передает управление в вызываемый подпроцесс, а после его окончания управление снова вернется в родительский процесс (см. Рисунок 3-11).

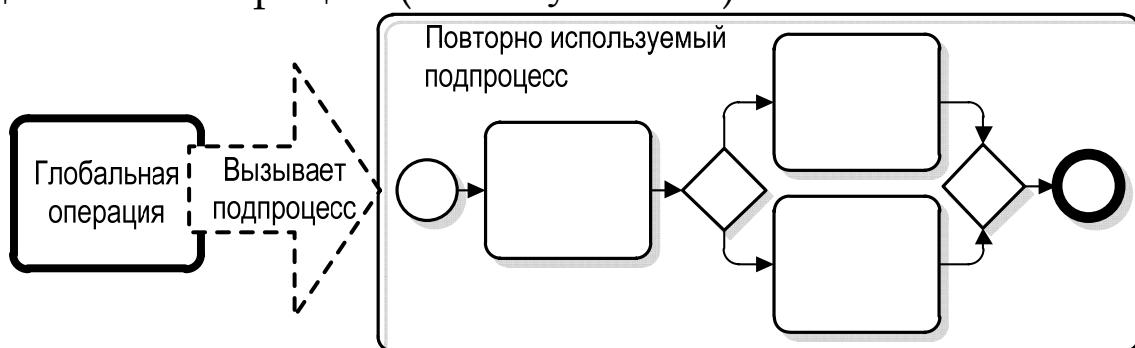


Рисунок 3-11. Вызывающая операция и глобальный процесс.

3.5.4. ОБЛАСТЬ ДЕЙСТВИЯ ДАННЫХ ПРИ ВЫЗОВЕ ГЛОБАЛЬНО ИЗВЕСТНОГО ПОДПРОЦЕССА

В рамках повторно используемого подпроцесса неизвестны данные, определенные в родительском процессе. Поскольку данные не будут автоматически доступны в вызываемом подпроцессе, потребуется явно передавать их, возможно определить их взаимное отображение или правило трансформации. Придется явно описать набор информационных объектов, передаваемых ему на вход, а затем организовать прием возвращаемой информации из подпроцесса в вызывающий процесс.

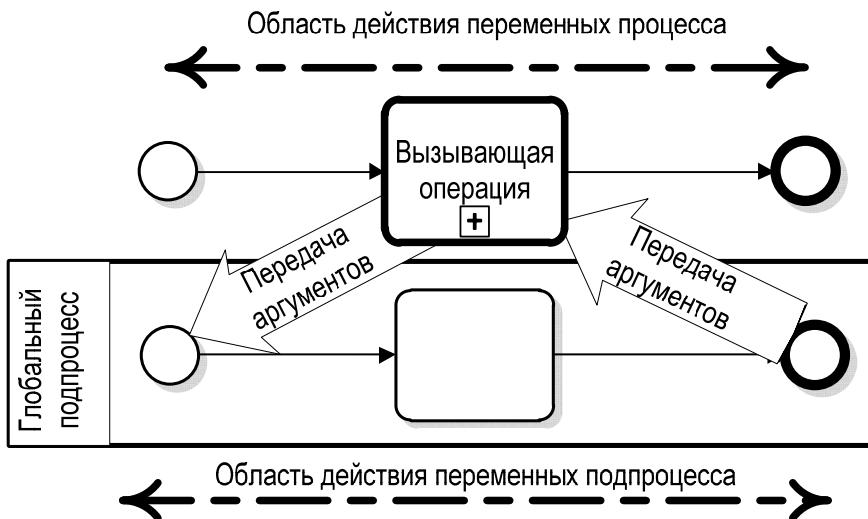


Рисунок 3-12. Повторно используемый подпроцесс

3.5.5. АД-НОС ПОДПРОЦЕСС ДЛЯ КОНКРЕТНОГО СЛУЧАЯ

Подпроцесс для конкретного случая (Ad-Hoc) предназначен для решения слабо формализованной задачи, когда заранее известен возможный спектр всех действий, но нельзя предсказать последовательность их исполнения. Подпроцесс Ad-Hoc состоит из операций и подпроцессов, которые, не связаны общими сценариями исполнения, так что пользователь в ходе исполнения сам определяет порядок, время запуска, длительность исполнения и число повторений операций и подпроцессов, перечисленных в Ad-Hoc подпроцессе. Пользователь может выбрать разные способы исполнения операций, например последовательный и параллельный. Во втором случае возможна конкуренция разных операций за доступ к общему ресурсу. Пример (см. Рисунок 3-13) изображает свернутый и развернутый Ad-Hoc подпроцесс, последний включает две операции. Бронировать гостиницу и билеты. Пользователь сам решает, в каком порядке выполнять операции.

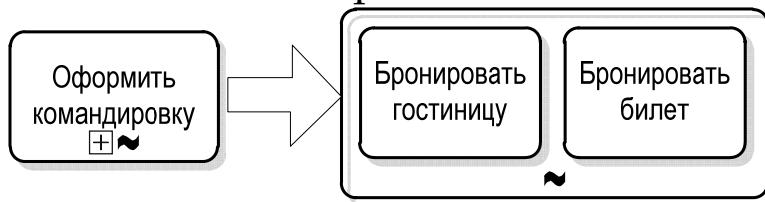


Рисунок 3-13. Свернутый и развернутый Ad-Hoc подпроцессы

Подпроцесс Ad-Hoc может включать только ограниченное количество элементов нотации BPMN: операции и группы операций, подпроцессы, объекты данных, потоки управления, ассоциации и ассоциации данных, логические элементы, потоки сообщений, промежуточные события. Подпроцесс Ad-Hoc не может включать следующие элементы нотации BPMN: начальные и конечные события, диалоги, элементы процессной хореографии

Операции, включенные в Ad-Hoc подпроцесс, могут быть частично логически взаимосвязаны, например, связь предшествование и последование отображает временную зависимость между работами типа "конец-начало", когда последующая работа может начаться только по завершении предшествующей. Другой тип связей отображает обмен информацией между операциями процесса Ad-Hoc. Кроме того, можно указать операции, осуществляющие откат данных. Пример (см. Рисунок 3-14) изображает подпроцесс, в котором присутствуют операции, связанные потоком управления, что определяет очередьность их запуска, объекты данных отображают передачу информации между операциями.

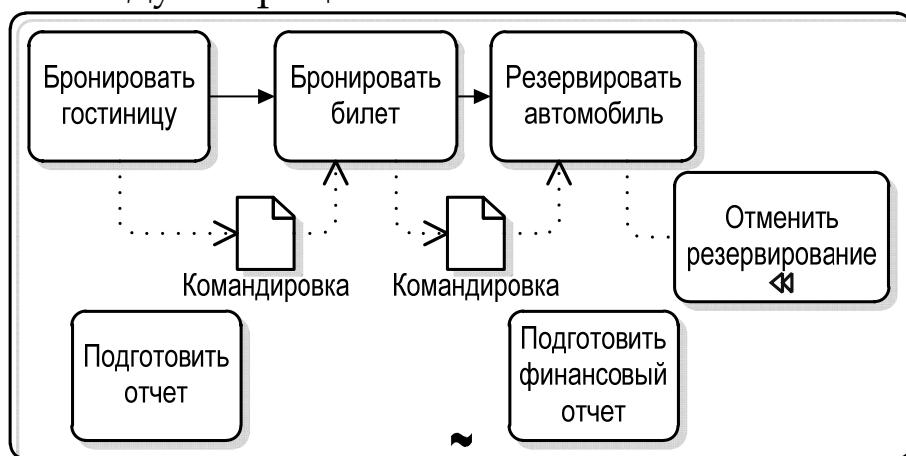


Рисунок 3-14. Подпроцесс «Ad-Hoc» с зависимыми операциями

3.5.6. СОБЫТИЙНЫЙ ПОДПРОЦЕСС

Событийный подпроцесс (стартующий по событию) описывает последовательность действий, которую необходимо выполнить после того, как будет зарегистрировано некоторое ожидаемое событие. Этот тип подпроцессов не является частью

стандартного сценария исполнения процесса, напротив, он показывает действия, выполняемые в особой ситуации. Он не связан непосредственными переходами управления с основным процессом.

Событийный подпроцесс может иметь только одно стартовое событие, причем только определенных типов:

- Сообщение (Message),
- Ошибка (Error),
- Эскалация (Escalation),
- Компенсация (Compensation),
- Условие (Conditional),
- Сигнал (Signal),
- Множественное событие (Multiple).

На схеме процесса этот подпроцесс обозначается с границей, изображаемой прерывистой линией. Пример (см. Рисунок 3-15) иллюстрирует развернутый (А) и свернутый (Б) событийные подпроцессы

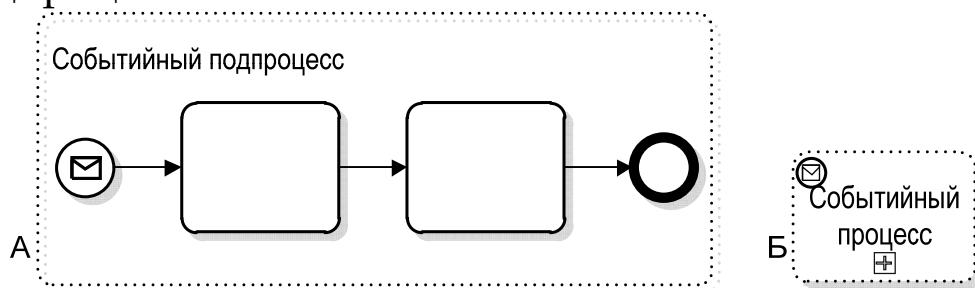


Рисунок 3-15. Развернутый и свернутый событийный подпроцессы

Подпроцесс, стартующий по событию может прерывать основной процесс или выполняться одновременно, параллельно с ним. Способ исполнения зависит от типа события, которое инициирует старт событийного подпроцесса (см. 6.9). Прерывающее стартовое событие приостанавливает основной процесс до окончания событийного. А непрерывающие стартовое событие инициирует исполнение событийного подпроцесса одновременно и параллельно с основным. Если установлен маркер множественного исполнения, событийный подпроцесс исполняется многократно.

Пример (см. Рисунок 3-16) изображает родительский процесс, который вызывает несколько событийных подпроцессов,

но разными способами. Во-первых, он содержит операцию отправки сообщения. Первый из вложенных подпроцессов стартует после получения сообщения, он не прерывает работу основного процесса и может исполняться несколько раз. Во-вторых, в ходе исполнения основного процесса могла произойти ошибка, которая прерывает его исполнение. Второй из вложенных подпроцессов обрабатывает ошибки, возникающие в ходе исполнения основного процесса.

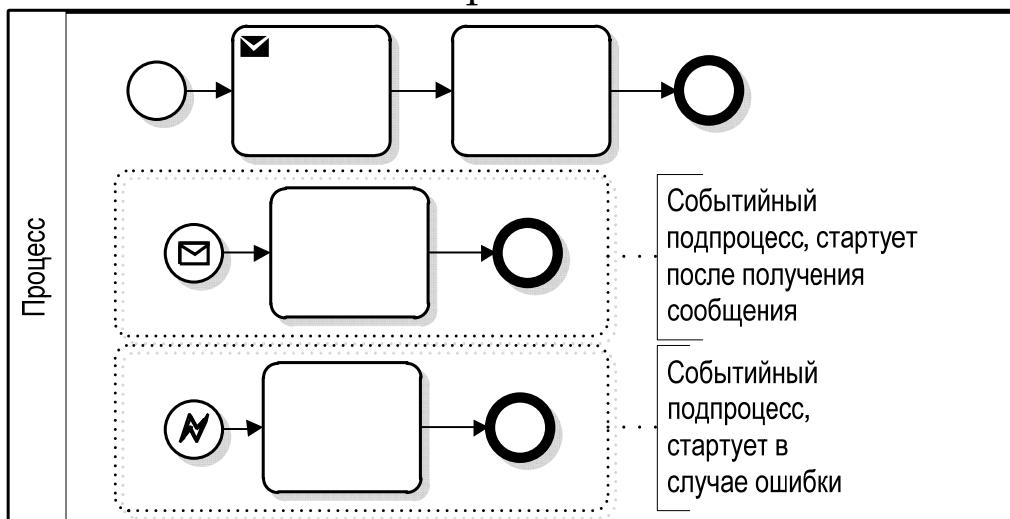


Рисунок 3-16. Событийный подпроцесс

Событийный подпроцесс является вложенным в основной процесс, поэтому все переменные, определенные в родительском процессе, будут автоматически доступны в потомке. И наоборот, все изменения данных, выполняемые в событийном процессе, станут автоматически известны в вызывающем процессе. Пример (см. Рисунок 3-17) изображает область действия переменных событийного подпроцесса.

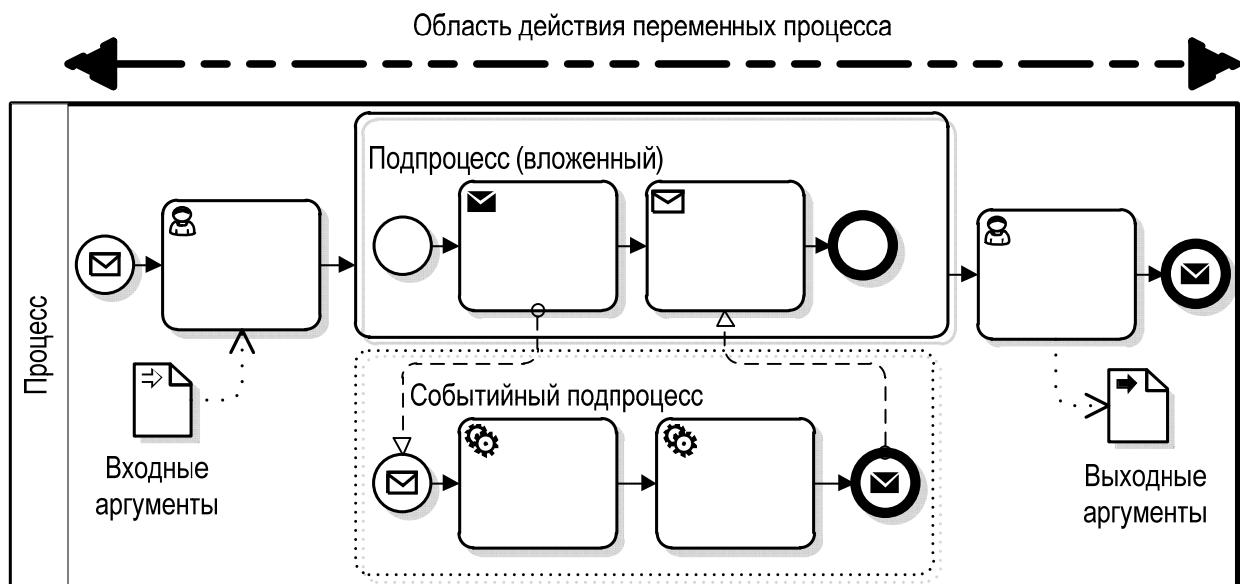


Рисунок 3-17. Область действия переменных событийного подпроцесса

3.5.7. ТРАНЗАКЦИОННЫЙ ПРОЦЕСС

Транзакция - это механизм, позволяющий объединять несколько действия в логический блок, чтобы принимать решение об успешности выполнения всего блока в целом, а не отдельных операций порознь. **Транзакционный подпроцесс** есть последовательность логически связанных операций, которая либо выполняется целиком и успешно, соблюдая целостность данных, либо не выполняется вообще и тогда все изменения данных отменяются. В последнем случае выполняется компенсация данных, которая заключается в том, что в обратном порядке, последовательно отменяются все сделанные ранее изменения в объектах данных.

Важно отметить, что транзакции связаны с изменением данных во внешних, по отношению к BPMN системах и включают межорганизационное взаимодействие, например, билеты и гостиницу мы бронируем через агентство или международную систему заказов. При работе с транзакциями BPMN 2.0 опирается на спецификацию Web Services Atomic Transaction, которая описывает способы координации для атомарных транзакций и для долго живущих бизнес транзакций. Именно последние, объединяющие совокупность атомарных транзакций, являются предметом рассмотрения данного вида процессов.

Согласно этим спецификациям, требуется взаимодействие на двух уровнях: между приложениями и между диспетчерами транзакций. В спецификациях подробно описываются форматы сообщений и обмен сообщениями для обоих уровней взаимодействия.

Межорганизационное взаимодействие обычно осуществляется с помощью Web Services, которое на схеме процесса изображаются как автоматическая операция. Если сервисная операция не завершается успешно, то в вызывающий процесс возвращается код завершения, содержащий информацию об ошибке. Благодаря этому, становится возможно оповестить транзакцию, что одно из действий не выполнено, т.ч. потребуется отменить все предшествующие изменения.

На схеме транзакционный процесс показывается с границей, изображаемой двойной линией. Он имеет один вход и три выхода (см. Рисунок 3-18). Первый описывает движение потока управления, когда транзакция завершилась успешно. Второй, соответствует случаю, когда транзакция была завершена, но возникла ошибка, т.ч. был выполнен откат данных. Третий выход описывает ситуацию, когда транзакция не была завершена из-за катастрофической внутренней ошибки, потребуется внешнее вмешательство.

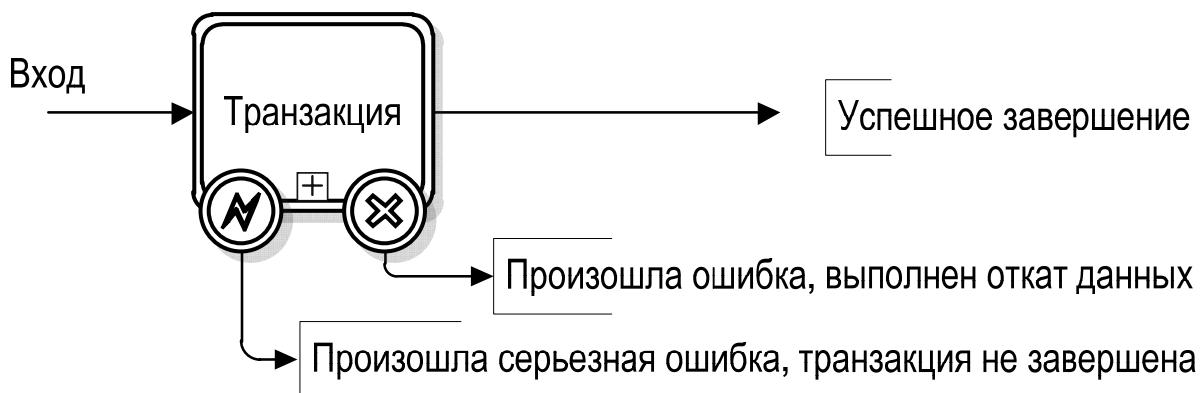


Рисунок 3-18. Свернутый транзакционный подпроцесс

1. Успешное завершение транзакционного процесса отличается от поведения обычного процесса. Когда поток управления достигает завершающего события, управления не возвращается в вызывающий процесс немедленно, как произошло бы в обычном процессе. Вначале транзакционный

протокол проверяет, что все участники успешно завершили свои задания, только после получения подтверждения управление передается в вызывающий процесс.

2. Если одна из операций транзакции завершилась ошибкой, произойдет откат транзакции в исходное состояние. Для этого для всех ранее завершенных операций будет выполнена компенсация. Операции обрабатываются в порядке обратном их исполнению. После того как все компенсации выполнены, поток управления продолжит движение по маршруту, отмеченному знаком события-отмена (Cancel), прикрепленного к границе транзакции. Существует два механизма оповещения об изменении исполнения.

- Во-первых, в ходе выполнения транзакции достигнуто конечное событие Отмена (Cancel). Этот тип конечного события можно использовать только в транзакционных процессах.
- Во-вторых, может поступить сообщение об Отмене от монитора транзакций.

Следует иметь в виду, что события типа Ошибка и Таймер не оказывают влияния на транзакцию и не вызывают исполнения компенсации.

3. Наконец, может произойти неисправимая ошибка, так что нормальное завершение операции и всей транзакции в целом становится невозможным. В этом случае используется прикрепленное событие Ошибка (Error). В случае возникновения такой ошибки, происходит прерывание исполнения текущей операции, компенсация не производится, начинается исполнение обработчика прерывания, на который указывает событие ошибки.

Рассмотрим пример (см. Рисунок 3-19) иллюстрирующий обработку транзакции. Производится перевод денег с одного счета на другой, транзакция включает две последовательные операции списание средства с первого счета и зачисление средств на второй. Если транзакция завершается успешно, мы можем быть уверены, что обе операции выполнены. Если же транзакция не завершается успешно, то могут возникнуть сле-

дующие ошибочные ситуации: деньги уже ушли с первого счета, но на другой не пришли, или деньги зачислены на второй счет, но не списаны с первого и т.д. Следует отменить операции с данными.

Если все операции, образующие транзакцию успешно завершены, то поток управления достигает простого конечного события. В этот момент происходит обмен сообщениями с монитором транзакций, после получения подтверждения об успешном завершении, исполнение продолжается.

Если первая операция успешно завершена, но вторая сообщила об ошибке, например, указан неправильный номер счета, то управление направляется на конечное событие Отмена (Cancel). Вначале будет выполнен откат первой операции, так что списанные средства вернутся на счет, а затем начнется выполнение обработчика прерывания. Можно попытаться повторить транзакцию.

Наконец, если любая из операций не была завершена, так что мы не знаем реального состояния данных, откат средствами BPMS станет невозможен. Придется откатывать данные либо средствами монитора транзакций, либо вручную. Прикрепленное событие-ошибка указывает сценарий продолжения для этого случая.

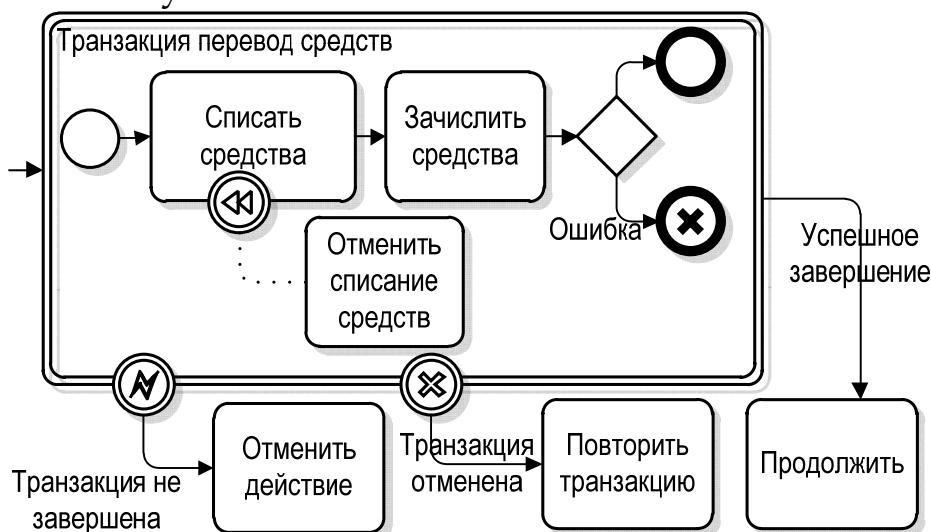


Рисунок 3-19.Обработка транзакции

4. ПОТОКИ УПРАВЛЕНИЯ.

Для моделирования порядка и маршрутов выполнения операций процесса используются потоки управления (ПУ), они изображаются на схеме в виде толстой, непрерывающейся стрелки в направлении от предшествующей операции к очередной. Поток управления всегда располагается внутри пула. Стрелка может иметь надпись, которая обозначает условие, соответствующее данному переходу. Таблица 4-1 показывает типы ПУ.

Для моделирования предпочтительного маршрута выполнения процесса обычно используется *безусловный* последовательный поток, который не содержит в себе никаких условий. Он показывает порядок, в котором будут выполняться операции процесса, если очередная операция завершилась успешно и ее цель была достигнута.

Для моделирования разветвок в процессе используется *условный* поток передачи управления, который может содержать некоторый критерий, определяющий, будет ли выполнен данный поток. Условие проверяется перед выполнением перехода, т.ч. он будет выполнен только в случае, когда условие истинно.

Поток управления по умолчанию используется в случае, если для всех других потоков проверяемое условие не верно.

Таблица 4-1. Потоки управления

Знак	Вид перехода
	Безусловный последовательный поток управления, показывает порядок, в котором выполняются операции процесса.
	Условный поток, помогает выбрать то направление, куда надо передать управление. Если условный поток выходит из операции, то необходим знак ромбик. Если условный поток выходит из логического оператора, то знак ромбик не ставится.
	Поток управления по умолчанию, определяет маршрут, который будет выполнен, если условие для всех остальных потоков ложно.

5.ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

В нотации BPMN логические операторы (ЛО) используются для маршрутизации потоков управления (ПУ) по разным направлениям и для синхронизации нескольких потоков. Они бывают двух типов: **ветвления** и **слияния**, первые разделяют и маршрутизируют поток управления по нескольким направлениям, вторые объединяют и синхронизируют несколько потоков в один. Внешне операторы обоих типов не различимы. У ЛО *ветвления* есть один входящий поток и, по крайней мере, два исходящих потока, а у ЛО *слияния* есть несколько входящих и один исходящий. Входящие и исходящие ПУ могут присоединяться к любой точке границы ЛО, не только к углам.

5.1. ЛОГИЧЕСКИЕ ОПЕРАТОРЫ И БИЗНЕС ПРАВИЛА

Логические операторы часто путают с бизнес правилами. ЛО изображает некоторую работу процесса, которая не изменяет входящий информационный поток, но маршрутизирует его в соответствии с условием, которое определяет направление движения выходного потока. ЛО есть элемент бизнес логики процесса, а условие, по которому осуществляется ветвление, есть бизнес- правило.

Под бизнес-правилом принято понимать утверждение, определяющее или ограничивающее некоторые аспекты бизнеса. Правила постулируют ограничения на исполнение процесса, но не определяют, как предполагается достичь результата. Принято различать следующие виды бизнес-правил:

1. *Правила поведения*: определяют необходимость выполнить соответствующее действие или управляющее воздействие.
2. *Правила определения*: устанавливают критерий применимости какого-либо бизнес понятия, называемого фактом, подразделяются на:
 - *Правила вычисления*, помогают узнать значения искомых величин, называемых фактами. Например, торговая скидка

определяется общим объемом закупок за определенный период и числом закупок по определенной категории товаров, платежеспособностью клиента и т.д.

- *Правила классификации*, помогают проверить истинность фактов. Например, клиент классифицируется как VIP, если на его счете имеется определенная сумма денег и он не имел задолжности по платежам.

Рассмотрим пример, ветвление процесса осуществляется на основе *правила поведения*, которое принимает значения Истинно и Ложно. Но что есть Истинно, а что есть Ложно, определяется *правилом классификации*. В свою очередь последнее должно получить на вход некоторое значение, которое, можно предположить, получено с использованием *правила вычисления*. Например, представим следующую последовательность действий: вычислить величину скидки как функцию от размера текущего заказа (*правило вычисления*), классифицировать размер скидки: большая, средняя, низкая (*правило классификации*) и, наконец, отправить сделку на одобрение руководителю с соответствующим уровнем полномочий (*правило поведения*).

Однако, распространенная практика моделирования - фиксировать на схеме правило ветвления, забывая про правила определения. Отсутствие в модели части бизнес-правил делает диаграмму потоков работ не полной.

Из сказанного следует практическая рекомендация – следует явно выделять *правила определения* в отдельные конструкции на диаграмме потоков работ. Это поможет аналитику четко локализовать соответствующую логику. Другая рекомендация заключается в том, что бы явно подписывать потоки управления, выходящие из ЛО, причем так, что бы отражать условие, на основании которого осуществлена маршрутизация.

5.2. ТИПЫ ЛОГИЧЕСКИХ ОПЕРАТОРОВ

В зависимости от условий, определяющих правила ветвления и слияния, ЛО делятся на виды.

Таблица 5-1 перечисляет все виды ЛО, отдельно описана семантика ветвления и слияния.

Таблица 5-1. Виды логических операторов

Знак	Название	Семантика	
		Ветвление	Слияние
	«И»	Параллельные потоки активируются одновременно и выполняются параллельно.	Исходящий поток активируется после завершения всех входящих ветвей
	«ИЛИ»	При разделении активируется одна или более ветвей.	Ждет завершения всех созданных ветвей и затем активирует исходящий поток.
	«Исключающее ИЛИ», управляемое данными	Исходящий поток направляется лишь по одной из исходящих ветвей.	Ожидает завершения любой входящей ветви, активирует исходящий поток.
	Комплексное условие	Моделирует комплексные условия ветвления.	Моделирует комплексные условия слияния.
	«Исключающее ИЛИ» событийное	Создает новый экземпляр процесса после наступления любого из указанных событий.	Не предусмотрено
	«Исключающее ИЛИ», событийный, создающий	Создает новый экземпляр процесса после наступления одного из указанных событий.	Слияние не предусмотрено
	Событийное, создающее процесс «И»,	Создает новый экземпляр процесса после наступления всех ожидаемых событий	Слияние не предусмотрено

5.2.1. ЛОГИЧЕСКИЙ ОПЕРАТОР «И»

Логический оператор «И» разветвляет входной ПУ на несколько параллельных ветвей, которые активируются одновременно (см. Рисунок 5-1). Число выходных ветвей д.б. заранее известно в момент моделирования. Число созданных выходных потоков равно числу ветвей.

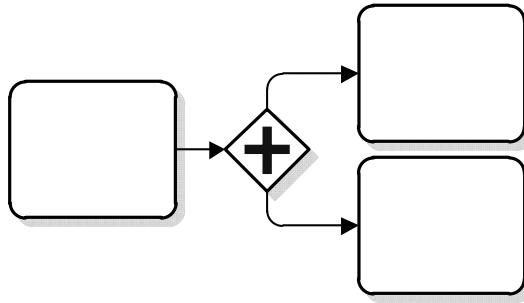


Рисунок 5-1. Логический оператор ветвления «И»

При слиянии параллельных ветвей оператор ждет завершения всех входящих ветвей и только затем активирует исходящий поток (см. Рисунок 5-2). Такое поведение называется **синхронизация** выходных потоков. Число входящих ветвей д.б. заранее известно в момент моделирования.

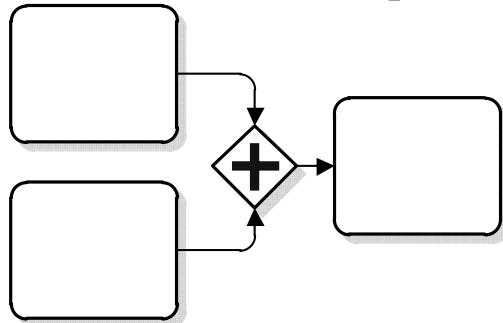


Рисунок 5-2. Логический оператор слияния «И»

Пример (см. Рисунок 5-3) иллюстрирует работу пары ЛО «И» ветвления и слияния. Произошло некоторое событие, о котором надо известить три службы. Вначале оператор регистрирует событие в журнале, затем передает информацию о событии в соответствующие службы. При этом, поток, поступивший на ЛО «И», разветвляется, активируя нужное количество выходных ветвей. Каждая ветвь обрабатывается независимо от остальных. Только после завершения всех параллельных ветвей, потоки соединяются и формируют один исходящий поток. Если одна из ветвей не завершится, то выходной поток не будет сформирован. Таким образом, три ветви между ЛО ветвления и слияния «И» выполняются параллельно, но асинхронно друг от друга. В данном примере реакция служб нас не интересует, важен факт оповещения. Синхронизируя три потока, на выходе, мы подтверждаем, что все три поручения выполнены.

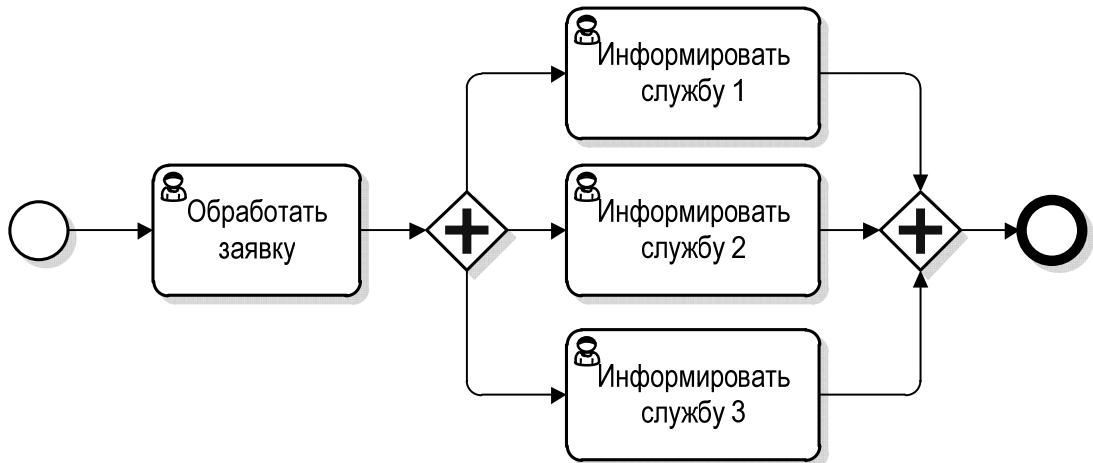


Рисунок 5-3. Парные логические операторы "И"

5.2.2. ЛОГИЧЕСКИЙ ОПЕРАТОР «ИЛИ» УПРАВЛЯЕМЫЙ ДАННЫМИ (OR)

Логический оператор «ИЛИ» управляемый данными (ИЛИ-д), разветвляет входной поток на несколько выходных, в зависимости от условия, которое включает анализ данных экземпляра процесса. Для каждой выходной ветви должно быть определено свое отдельное условие. В момент исполнения происходит анализ каждого из условий и, если для какой-то ветви условие истинно, порождается выходной поток в нужном направлении. На этом проверка не прерывается, остальные условия будут проверены и, если они так же окажутся истинными, будут созданы параллельные ветви. Хорошая практика размещать на схеме ветку безусловного перехода, исполняемую по умолчанию, в том случае, если ни одно из условий не окажется истинно. На схеме процесса безусловный поток управления изображается с косой чертой, пересекающей стрелку, этот знак позволяет отличить его от условного перехода. Если маршрут по-умолчанию в схему не заложен, то на этапе исполнения может возникнуть ситуация, когда ни один из предложенных маршрутов не будет выполнен, при этом возникнет ошибка. Ее можно попытаться перехватить и обработать как событие (см. п. 7). Пример (см. Рисунок 5-4) иллюстрирует ветвление «ИЛИ-д». Если оба Условие1 и Условие2 истинны, то Задачи «А» и «Б» будут выполнены. Если оба условия ложны, то будет активировано задание «Г».

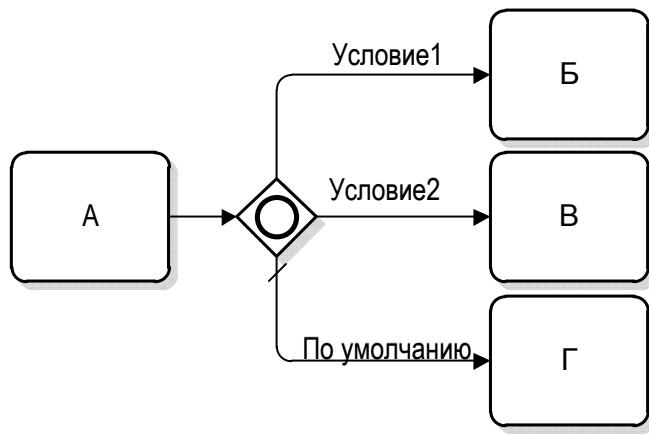


Рисунок 5-4. Оператор ветвление «ИЛИ» управляемый данными

Семантика слияния ПУ на ЛО «ИЛИ-д» является достаточно сложной. Дело в том, что ЛО «ИЛИ-д» ожидает поступления ПУ из некоторых параллельных ветвей и синхронизирует их в единый выходной поток (как ЛО «И»), но при этом он не дожидается ПУ из других ветвей. Как понять, завершения скольких ветвей ожидает ЛО «ИЛИ-д», а какие ветви он игнорирует? Обычно, ЛО *ветвления* и *слияния* используются в паре, так что узел, объединяющий потоки, знает число ветвей, которые были созданы.

Замечание: В реальной схеме ЛО могут употребляться не парно, при этом ЛО «ИЛИ-д» может неверно «рассчитать» число ожидаемых ПУ, что нарушит логику работы процесса.

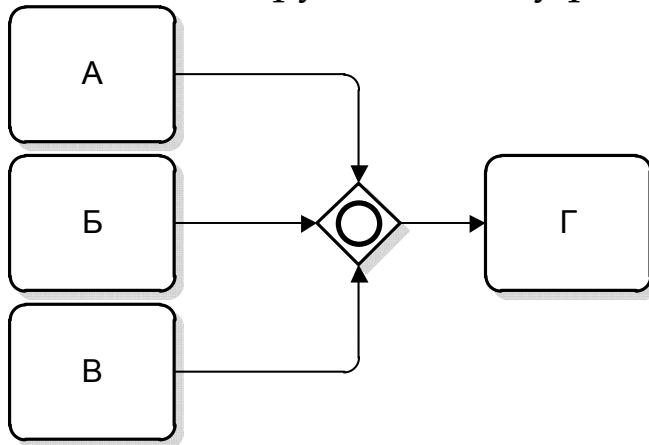


Рисунок 5-5. Оператор слияние «ИЛИ» управляемый данными

Рассмотрим пример (см. Рисунок 5-6), произошло страховое автомобильное событие, клиент позвонил в страховую компанию и сообщил об аварии, оператор зарегистрировал событие и вызывает только те службы, которые необходимы в данном

случае. Все виды служб заранее известны, но конкретное количество вызываемых служб зависит от обстоятельств аварии и определяется сотрудником компании по результатам телефонного разговора с клиентом. После того, как все вызовы служб будут выполнены, оператор слияния объединит потоки, выполнение м.б. продолжено.

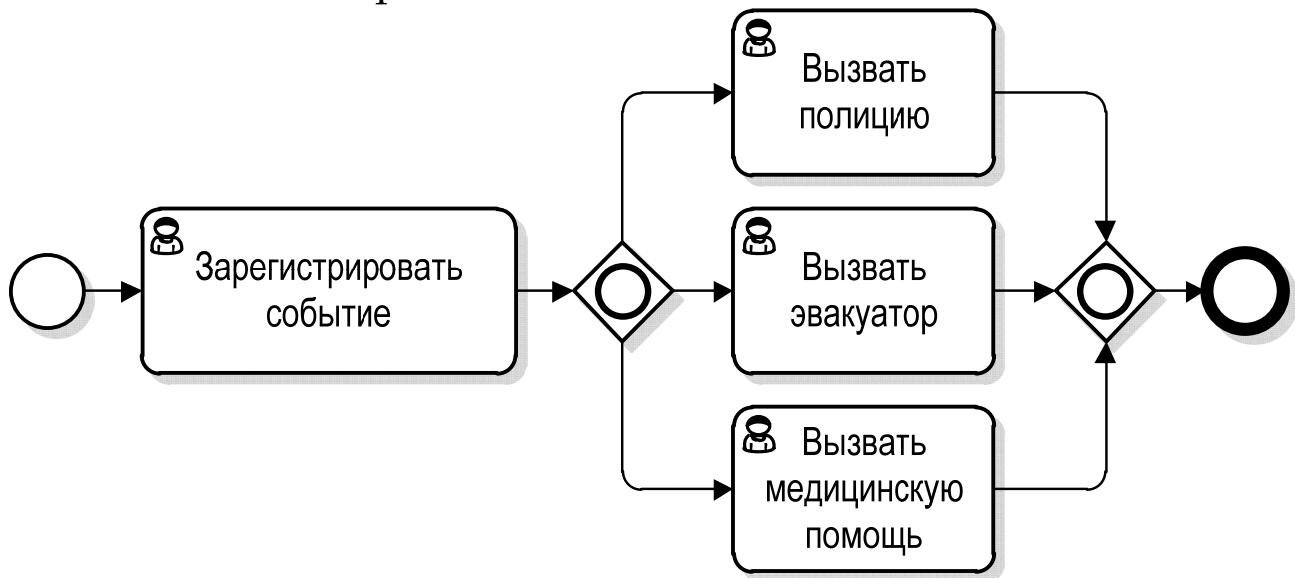


Рисунок 5-6. Парные логические операторы "ИЛИ".

5.2.3. ЛОГИЧЕСКИЙ ОПЕРАТОР «ИСКЛЮЧАЩЕЕ ИЛИ» УПРАВЛЯЕМЫЙ ДАННЫМИ (XOR)

Логический оператор «Исключающее ИЛИ» управляемый данными (Искл-ИЛИ) направляет входной поток строго на одну из нескольких выходных ветвей, в зависимости от условия, которое включает анализ данных экземпляра процесса. В отличие от обычного «Искл-ИЛИ» выходные потоки рассматриваются как альтернативные и взаимоисключающие. Ситуация, при которой выполняются сразу несколько из условий, невозможна. Для предотвращения конфликта для всех условных переходов, устанавливается приоритет, определяющий порядок проверки условий. Обычно, приоритет определяется последовательностью, в которой перечислены условия. Выбирается первый переход по списку, если условие не выполнено, то выбирается следующий переход. Если условие выполнено, то происходит переход управления в указанном направлении, а остальные условия игнорируются и не проверяются.

Для ЛО «Искл-ИЛИ» обязательно должен быть определен один безусловный переход передачи управления. Это необходимо в том случае, если ни одно из перечисленных условий не может быть выполнено. На схеме процесса безусловный ПУ изображается с косой черточкой, пересекающей стрелку, этот знак позволяет отличить его от условного перехода. Если ни одно из условий не выполняется и, при этом, переход по умолчанию не определен, генерируется сигнал об ошибке. Он м.б. перехвачен и обработан с помощью событий (см. п. 6.11.1)

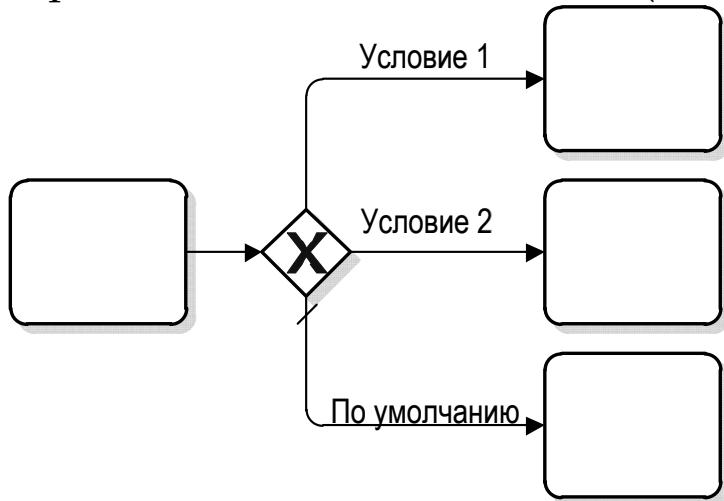


Рисунок 5-7. Ветвление «Исключающее ИЛИ» управляемое данными

При слиянии ПУ, пришедших из входящих ветвей, никакие условия не проверяются, синхронизации не происходит, т.ч. ЛО «Искл-ИЛИ» пропускает на выход потоки, пришедшие из всех входных ветвей. Когда приходит первый поток управления, потоков в остальных ветвях еще нет, т.ч. он проходит на выход без ожидания и синхронизации с другими ветвями. Когда позднее поступит поток управления из другой ветви, он также породит еще один выходной поток, так что на выходе появится несколько экземпляров потоков и т.д. Пример (см. Рисунок 5-8) иллюстрирует слияние трех потоков. Если поступят потоки управления из ветвей «А» и «Б», то операция «Д» будет выполнена два раза.

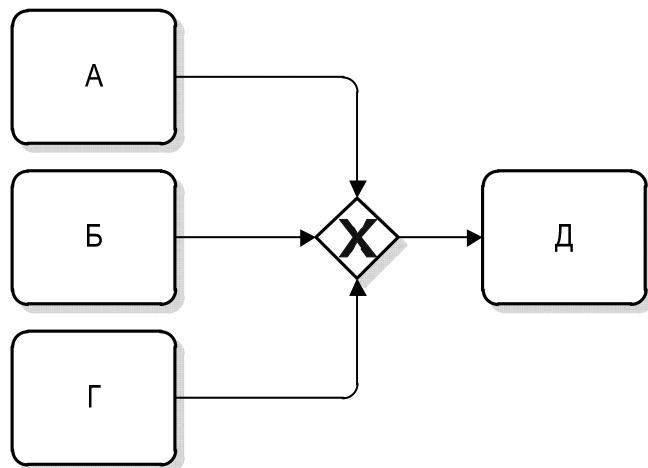


Рисунок 5-8. Слияние «Исключающее ИЛИ»

Если разветвляющий или объединяющий ЛО «Искл-ИЛИ» идут парой, то поток управления возникнет только в одной из ветвей, т.ч. аналитику не надо заботиться, что делать с «лишним» экземпляром ПУ. Но если операторы применяются не парно, порознь, то следует быть внимательным, что бы исключить неконтролируемое размножение потоков управления.

Пример (см. Рисунок 5-9) иллюстрирует парное использование операторов «Искл-ИЛИ». Оператор страховой компании принимает звонки от клиентов, если произошла авария, то инициируется одна последовательность операторов обработки. Если другой тип страхового события, то выполняются иные действия. Важно, что действия оператора при разных событиях являются взаимоисключающими. После завершения любой из ветвей процесс продолжается.

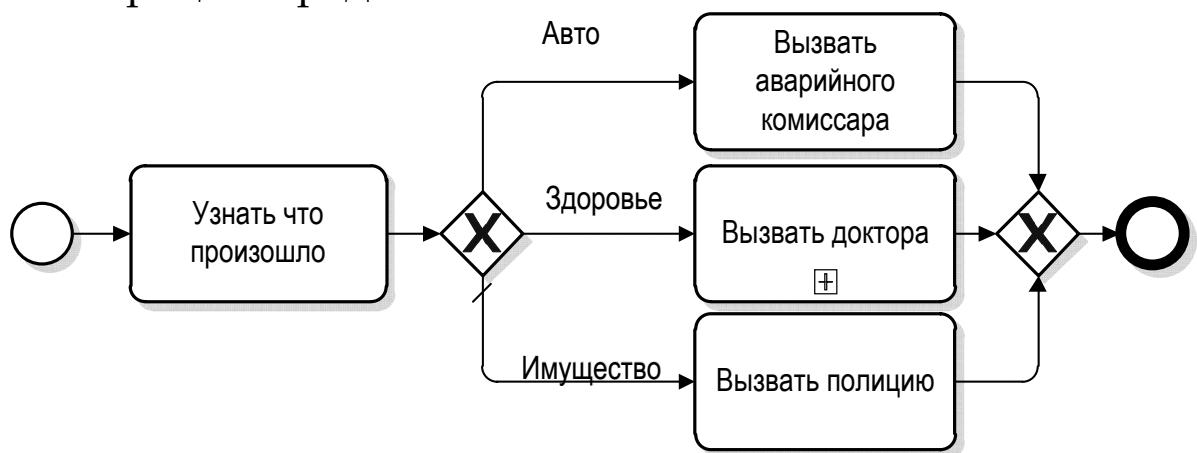


Рисунок 5-9. Парные операторы «Исключающее ИЛИ»

Если аналитик предполагает, что ЛО «Искл-ИЛИ» используется не парно, в результате чего на выходе узла могут появится несколько выходных потоков, он должен принять решение, является ли такое поведение ожидаемым или нет. В последнем случае, следует принять меры, что бы обработать и исключить такую ситуацию, например, путем использования другого ЛО, например, с комплексным условием, которое умеет обрабатывать нестандартную логику объединяемых потоков.

5.3. ЛОГИЧЕСКИЙ ОПЕРАТОР КОМПЛЕКСНОЕ УСЛОВИЕ

Логический оператор комплексное условие моделирует сложные, составные условия ветвления и слияния. Семантика разветвления в целом соответствует операции «ИЛИ» с той разницей, что используется одно комплексное условие, тогда как в «ИЛИ» их несколько, по одному на каждую выходную ветвь. Комплексное условие должно определять переход по умолчанию, в противном случае возможна ситуация, когда ни один переходов не будет выбран, что приведет к ошибке периода исполнения.

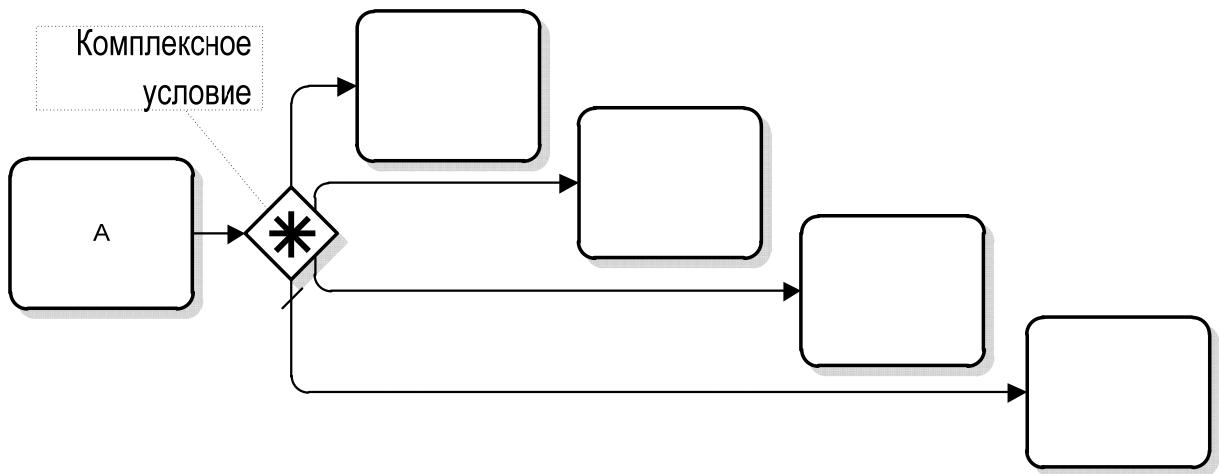


Рисунок 5-10. Оператор ветвления комплексное условие

Семантика соединения потоков по комплексному условию, в отличие от обычного «ИЛИ», позволяет указать способ синхронизации и число входных потоков, поступления которых следует ожидать. Например, можно указать, что выходной

поток появится только после завершения определенного количества входных ветвей. Следует учесть, что стандарт не накладывает специальных ограничений на условие синхронизации, так что может случиться, что число ожидаемых потоков менее или более чем число реально завершаемых. В первом случае, на выходе может быть сгенерировано не один, а несколько потоков управления. Во втором случае, выходной поток не возникнет никогда. Следует быть внимательным, поскольку внешние операторы ветвлениия и объединения неразличимы, так что следует использовать комментарии и аннотации, что бы пояснить смысл использования оператора.

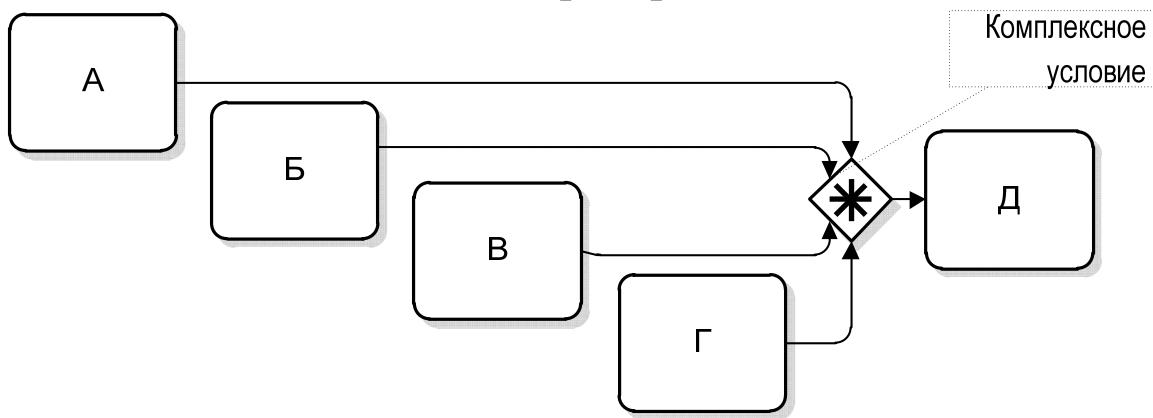


Рисунок 5-11. Оператор слияния комплексное условие

5.4. СОБЫТИЙНЫЙ ОПЕРАТОР «ИСКЛЮЧАЮЩЕЕ ИЛИ»

Событийный оператор «Исключающее ИЛИ» (Искл-ИЛИ-с) моделирует решение, управляемое результатом события, причем с каждым возможным выходным маршрутом связанно отдельное обрабатывающее событие (см. п. 6.2.2). В качестве возможных событий могут выступать:

- Сообщение
- Таймер
- Условие
- Сигнал

Событийный оператор Исключающее ИЛИ имеет один вход и требуемое число выходных ветвей, в которых размещено по обрабатывающему событию. Семантика разветвления тако-

ва: входной поток разветвляется на указанное число параллельных ветвей, каждый из потоков достигает соответствующего перехватывающего события, где останавливается и ждет наступления первого события. Наступившее событие, активирует соответствующую ветвь и одновременно аннулирует все остальные, как предусмотрено логикой событийного оператора исключающее ИЛИ. Рекомендуется принять меры, что бы процесс не «завис» в этом узле, если ни одно из событий не произойдет. Например, таймер, размещаемый в одной из параллельных выходных ветвей, «работает» как переход по умолчанию, выводит процесс из зависания после некоторого периода ожидания.

Пример (см. Рисунок 5-12) иллюстрирует работу оператора «Искл-ИЛИ-с». Операция процесса отправляет оповещение и далее начинается ожидание наступления одного из трех событий: ответа от получателя, сигнала системы или таймера. Ожидание может продолжаться не более одного дня с момента отправки. Если первым будет получен ответ от получателя, то будет выбрана первая ветвь, но если раньше продел сигнал завершить ожидание, то будет выбрана вторая ветвь.

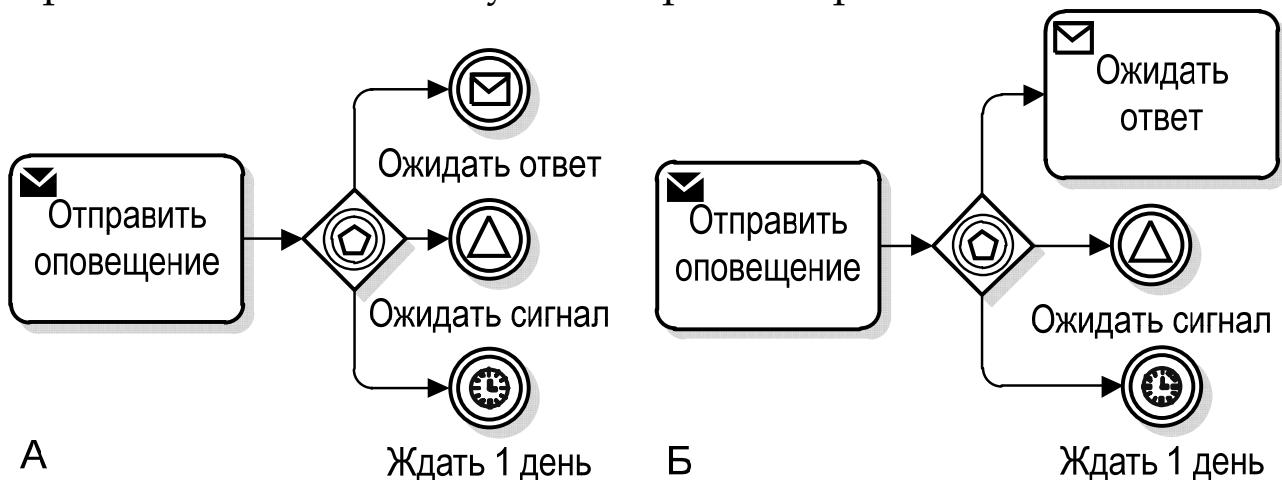


Рисунок 5-12. Событийный оператор «исключающее ИЛИ»

Пример (см. рисунок Б) показывает, что в комплексном узле вместо события-сообщение можно использовать операцию отправить сообщение. Но следует оговориться: операция не может иметь прикрепленных событий, кроме того, нельзя одновременно использовать оба знака на одной диаграмме.

Семантика объединяющего событийного оператора стандартом не специфицирована, но если возникает необходимость объединить потоки, наиболее подходящим будет ЛО «Исключающее ИЛИ».

5.5. СОБЫТИЙНЫЙ ОПЕРАТОР «ИСКЛЮЧАЮЩЕЕ ИЛИ», (**СОЗДАЕТ НОВЫЙ ЭКЗЕМПЛЯР ПРОЦЕССА**)

Событийный оператор Исключающее ИЛИ создающий новый экземпляр процесса (Искл-ИЛИ-проц), позволяет определить сложное условие запуска процесса на исполнение, связанное с наступлением одного из нескольких возможных видов событий. Например, старт процесса может произойти либо после получения сообщения от клиента по почте, либо путем обращения в офис продаж, либо по таймеру. Этот ЛО не имеет входного потока, только несколько выходных. Число ветвей должно быть известно на этапе моделирования. В каждой выходной ветви размещается одно событие. Если наступает любое из перечисленных событий, то происходит старт процесса, остальные ветви аннулируются (см. п. 1316.7).

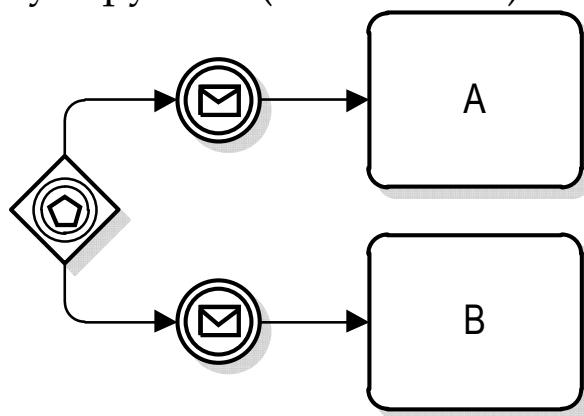


Рисунок 5-13. Событийный оператор «исключающее ИЛИ», (создает новый экземпляр процесса)

Событийный оператор «Искл-ИЛИ-проц», создающий новый экземпляр процесса, не имеет парного событийного ЛО соединяющего потоки, поэтому следует использовать обычный

ЛО слияния «Исключающее ИЛИ».

5.6. СОБЫТИЙНЫЙ ОПЕРАТОР «И» (СОЗДАЕТ НОВЫЙ ЭКЗЕМПЛЯР ПРОЦЕССА)

Событийный оператор «И», создающий новый экземпляр процесса (И-проц), моделирует ситуацию, когда старт процесса определяется наступлением сразу всех событий определенных на схеме в момент моделирования. Одного из событий для старта процесса недостаточно.

Пример (см. Рисунок 5-14) иллюстрирует старт процесса, когда необходимы оба: обращение клиента, а также специальное разрешение от внешней организации.

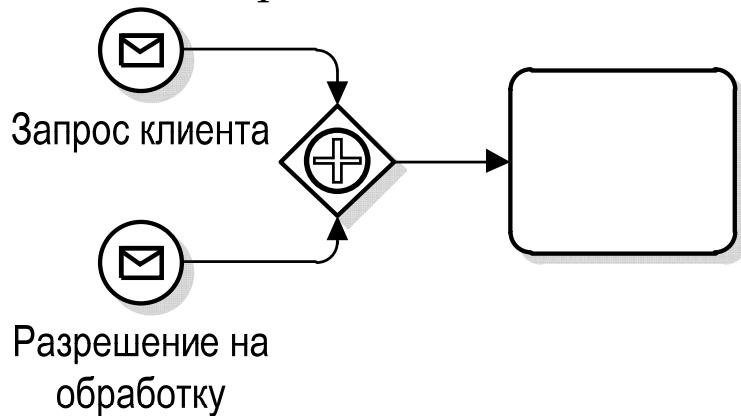


Рисунок 5-14. Событийный оператор «И» (создает экземпляр процесса)

Парный узел слияния спецификацией не предусмотрен.

6.СОБЫТИЯ

Термин «событие» является достаточно общим и используется для обозначения изменения свойств некоторого объекта, произошедшее в определенный момент времени и зафиксированное наблюдателем в оповещении от этого объекта. Обычно событие противопоставляется процессу, который происходит в интервале времени, а событие только в определенные моменты. С каждым событием связана причина его появления и возможное воздействие, которое оно оказывает на дальнейший исполнение процесса. Следует различать понятия события, как явление и как программу обработчик проблемной ситуации.

Событие, в смысле явления, может происходить как в результате выполнения процесса, так и вследствие изменения внешней информационной среды. Будем различать:

- Внешнее событие, инициируемое внешним по отношению к процессу агентом, например, клиент разместил заказ, клиент отозвал свой заказ. Существует входное воздействие в виде оповещения от клиента.
- Временное событие, связанное с отсчетом времени, входной информационный поток отсутствует. Например, событие м.б. связано с окончанием отведенного срока рассмотрения заявки или с наступлением конца календарного периода.
- Событие состояния, связанное с изменением статуса переменной процесса. Например, после обработки заявки ее статус станет Обработана.
- Событие ошибки, связанное с невозможностью выполнить задачу или процесс. Например, операция обращается к внешнему сервису, а последний может быть недоступен. В последнем случае вырабатывается сигнал-ошибка.

События, в смысле обработчик проблемной ситуации, есть некоторая последовательность действий: операция или подпроцесс, выполняемые в случае получения соответствующего оповещения.

6.1. ТИПЫ СОБЫТИЙ

Таблица 6-1. Моделирование событий в нотации BPMN_#

СОБЫТИЯ	Начальные		Промежуточные				Конечное генерирующее	
	Верхнего уровня	Подпроцесс	Поток		Границевые			
		Прерывающее	Не прерывающее	Генерирующее	Обрабатывающее	Генерирующее		
Простое показывает начало процесса или его окончание.	○			○	○	○	○	
Сообщение: принимает и отправляет сообщения.	✉	✉	✉	✉	✉	✉	✉	
Таймер: отмечает моменты времени и временные периоды	⌚	⌚	⌚	⌚	⌚	⌚	⌚	
Эскалация: поднимает уровень решения задания		▲	▲		▲	▲	▲	
Условие: проверяет истинность условия.	☰	☰	☰		☰		☰	
Ссылка: связывает две разные части одного процесса.				➡	➡			
Ошибка: обрабатывает заданный тип ошибок.		⚡			⚡	⚡	⚡	
Отмена: инициирует или обрабатывает отмену транзакции.					✖	✖	✖	
Компенсация: инициирует или обрабатывает откат транзакции		◀◀		◀◀	◀◀	◀◀	◀◀	
Сигнал: связь между процессами один ко многим.	△	△	△	△	△	△	△	
Составное: срабатывает при наступлении нескольких событий.	pentagon	pentagon	pentagon	pentagon	pentagon	pentagon	pentagon	
Параллельное составное: нужны все из перечисленных.	⊕	⊕	⊕		⊕	⊕	⊕	
Прекращение: немедленно завершает все процессы							●	

6.2. КЛАССИФИКАЦИЯ СОБЫТИЙ

События можно классифицировать по разным признакам.

6.2.1. НАЧАЛЬНЫЕ, ПРОМЕЖУТОЧНЫЕ И КОНЕЧНЫЕ

В зависимости от положения в процессе, события классифицируют на *начальные*, *промежуточные* и *конечные*. **Начальное событие** инициирует создание нового экземпляра процесса, например, после получения обращения от клиента с просьбой о выдаче кредита порождается экземпляр процесса, содержащий заявку данного заявителя. **Конечное событие** завершает выполнение всего процесса, либо одного из его потоков. **Промежуточное событие** располагается между начальным и конечным, оно используется для синхронизации ветвей данного процесса или потоков управления разных процессов. Соответственно, у начального события нет входящего потока управления, у завершающего нет исходящего, у промежуточных есть оба потока. Начальное событие имеет контур, выполненный тонкой линией, у промежуточного двойной контур, а у завершающего контур выполнен толстой линией (см.Рисунок 6-1).



Рисунок 6-1. Классификация событий по местоположению в процессе

6.2.2. ГЕНЕРИРУЮЩИЕ И ОБРАБАТЫВАЮЩИЕ

Все множество событий в нотации BPMN можно условно разделить на *генерирующие* и *обрабатывающие*. **Генерирующее событие**, посыпает оповещение в другой процесс или поток управления. **Обрабатывающее событие** принимает это оповещения и инициирует заранее предусмотренную программу обработки. Начальные события могут быть только обрабатывающими. Промежуточные события могут быть и обрабатыва-

вающими и генерирующими. Завершающие - только генерирующими.

На схеме процесса в нотации BPMN генерирующие события имеют закрашенный сплошным цветом маркер внутри окружности, а обрабатывающие события имеют контурный, не закрашенный маркер, расположенный внутри окружности (см. Рисунок 6-2).

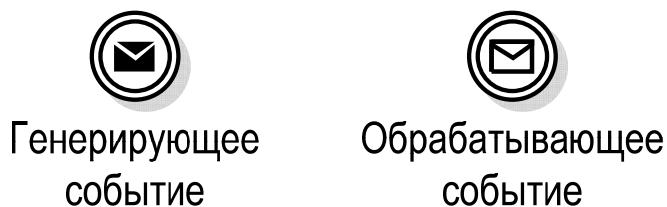


Рисунок 6-2. Классификация событий по типу обработки

6.2.3. НЕЗАВИСИМЫЕ И ПРИКРЕПЛЕННЫЕ

Промежуточные события можно классифицировать на *независимые* и *прикрепленные*. **Независимые события** размещаются на схеме как отдельные элементы, они могут быть генерирующими или обрабатывающими. **Прикрепленные события** привязываются к соответствующим операциям или подпроцессам и могут влиять на их выполнение, они могут быть только обрабатывающими. Пример (см. Рисунок 6-3А) показывает независимые события: отправить сообщение является генерирующим, а ожидать срабатывания таймера является обрабатывающим. Второй пример (см. Рисунок Б) показывает прикрепленное обрабатывающее сообщение.

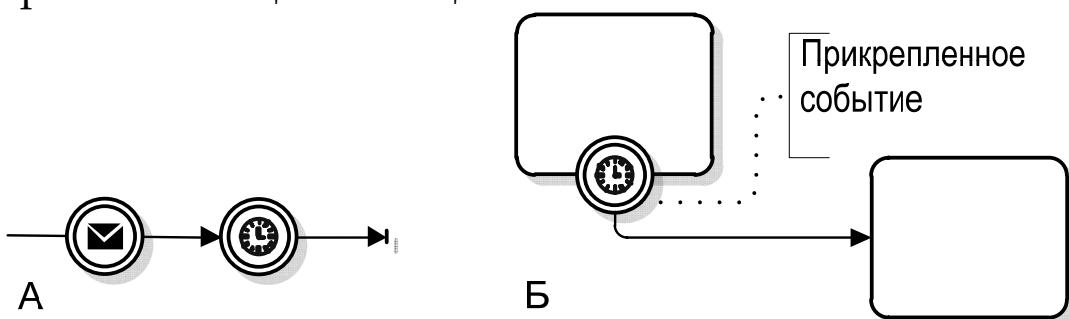


Рисунок 6-3. Независимые и прикрепленные события

6.2.4. ПРЕРЫВАЮЩИЕ И НЕПРЕРЫВАЮЩИЕ

Обрабатывающие прикрепленные события можно классифицировать на прерывающие и непрерывающие. **Прерывающее событие** приостанавливает выполнение операции (в том числе подпроцесса), к которой оно прикреплено. В некоторых случаях, прерывающие события могут завершать выполнение одного или нескольких потоков или даже всего экземпляра процесса в целом. **Непрерывающее событие**, напротив, никак не влияет на выполнение операции или процесса не оказывает, однако, создает дополнительный поток управления внутри процесса. У прерывающих событий контур выполнен сплошной линией, а у непрерывающих – пунктирной.

Пример (см. Рисунок 6-4А) иллюстрирует ситуацию, когда истекло нормативное время, отведенное на исполнение задания, срабатывает таймер, задание отправляется руководителю, который либо выполняет его сам, либо назначает другого исполнителя. При этом операция А прерывается, результаты работы, если они были, будут потеряны. Второй пример (см. Рисунок Б) показывает непрерывающее событие. Исполнение операции А не останавливается, руководителю отправляется оповещение, он может позвонить исполнителю и попросить его ускорить работу.

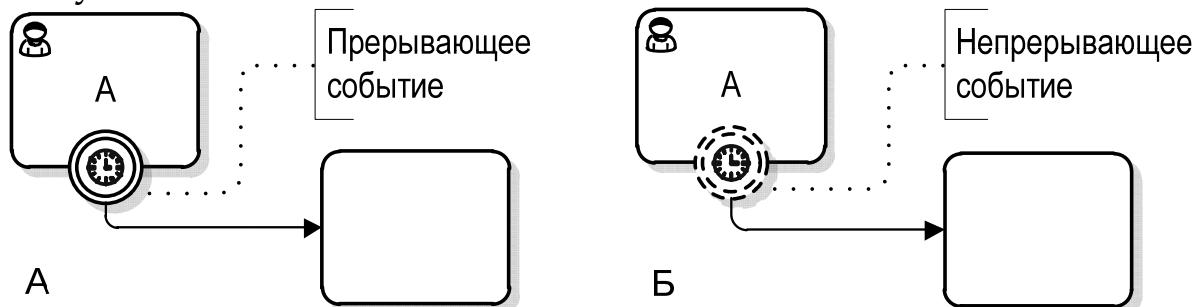


Рисунок 6-4. Классификация событий по влиянию на выполнение

6.3. СОБЫТИЯ И ДАННЫЕ

Следующие типы событий могут нести полезную информационную нагрузку: Сообщение, Эскалация, Сигнал, Ошибка, Составное. Что бы указать необходимую полезную нагрузку, следует использовать механизм ассоциации данных. Когда поток

достигает генерирующего события, происходит отображение входящего информационного потока на внутренние аргументы элемента события. Это отображение можно указать с помощью ассоциации данных. Затем, внутренние аргументы используются, что бы создать полезную информационную нагрузку, генерируемую сообщением. Пример (см. Рисунок 6-5) иллюстрирует ассоциацию данных при отправке сигнала. Если ассоциация отсутствует, то отправляется пустое сообщение, не содержащее полезной нагрузки.



Рисунок 6-5. Ассоциация данных в событиях

Аналогично, но в обратном порядке осуществляется перенос данных из принятого оповещения обратно в процесс.

6.4. СИГНАЛ

Сигнал есть механизм координации и связи между разными процессами или ветвями одного процесса. Они могут передаваться внутри одного процесса, между разными участниками в разных пулах, между разными процессами одного проекта. Сигнал имеет имя, благодаря которому обрабатывающее событие может среагировать только на оповещение определенного типа. Стандарт говорит, что сигнал может иметь «полезную» информационную нагрузку, однако не все реализации BPM поддерживают такую возможность².

Сигнал реализует широковещательную рассылку, при ко-

² Некоторые реализации BPMS допускают передавать внутри сигнала полезный набор данных.

торой оповещение предназначено для приёма всеми процессами получателями. Он имеет источник, но не имеет определенных получателей. Т.о. сигнал получат все слушатели, ожидающие наступления определенного события. Различают графические элементы, генерирующие и принимающие сигнал, первые являются источниками, а вторые получателями оповещения. Таблица 6-2 показывает элементы для работы с сигналами.

Таблица 6-2. Графические элементы для работы с сигналами

Событие, генерирующее сигнал	
Событие, принимающее сигнал	

Событие, генерирующее сигнал, размещается в потоке управления. **Событие, принимающее сигнал**, может размещаться либо в потоке управления, либо прикрепляется к границе некоторой операции.

Механизм сигналов имеет много общего с механизмом обработки ошибок, при этом он оповещает об успешных событиях, тогда как ошибки сообщают о неудачах. Сигналы находят применение в следующих ситуациях:

- Оповещение участников о завершении этапа обработки;
- Синхронизация исполнения нескольких ветвей или разных процессов;
- Координация исполнения цепочки процессов.
- Обработка ошибок;

6.4.1. ОБРАБОТКА СИГНАЛОВ

Рассмотрим особенности обработки сигналов.

Во-первых, сигнал можно посыпать как внутри одного пула, так и между несколькими пулами. Сигналы могут использоваться на всех уровнях подпроцессов.

Во-вторых, сигналы работают по принципу широковещательной связи. Это означает, что при передаче информации используется один источник и неограниченное количество приемников. Связь между одним адресантом и многими адре-

сатами обеспечивается за счет одинакового именования источника (генерирующего события) и приемника сигнала.

В-третьих, при передаче сигнала нельзя идентифицировать конкретный ЭП, в которой должно поступить сообщение. Это означает, что сигнал получат все экземпляры процессов, которые его ожидают и где управление приостановлено до получения соответствующего сигнала³. Процесс источник ничего не знает о процессе получателе и об их количестве, а получатель не знает об источнике или других получателях.

Обработка сигнала осуществляется следующим образом. Когда поток управления доходит до события генерирующего сигнал, последнее осуществляет широковещательную рассылку оповещения, после чего управление немедленно передается на следующую задачу. Ранее того, в другой ветви или в другом процессе точка управления остановилась на событии «принимающем сигнал», где ожидает получения соответствующего оповещения. Сигнал имеет имя, так что получатели могут выделить нужное оповещение из нескольких возможных. Когда нужный сигнал будет получен, управление будет передано на следующую задачу процесса. Если сигнал поступит на «принимающее» событие раньше, чем туда прибудет поток управления, он будет проигнорирован. Если событие «принимающее сигнал» прикреплено к границе некоторой операции, то поступление сигнала инициирует новый поток управления.

В качестве примера рассмотрим процесс пакетной обработки (см. Рисунок 6-6), где изображен один управляющий процесс и заранее неизвестное число подпроцессов, выполняющих полезную работу, например, загрузку данных из подпроцессов во внешнюю учетную систему. Загрузка осуществляется однократно, по сигналу управляющего процесса

³ Те реализации BPMS , которые позволяют передавать внутри сигнала полезный набор данных, допускают, что получатель может среагировать, на получение сигнала определенного содержания.

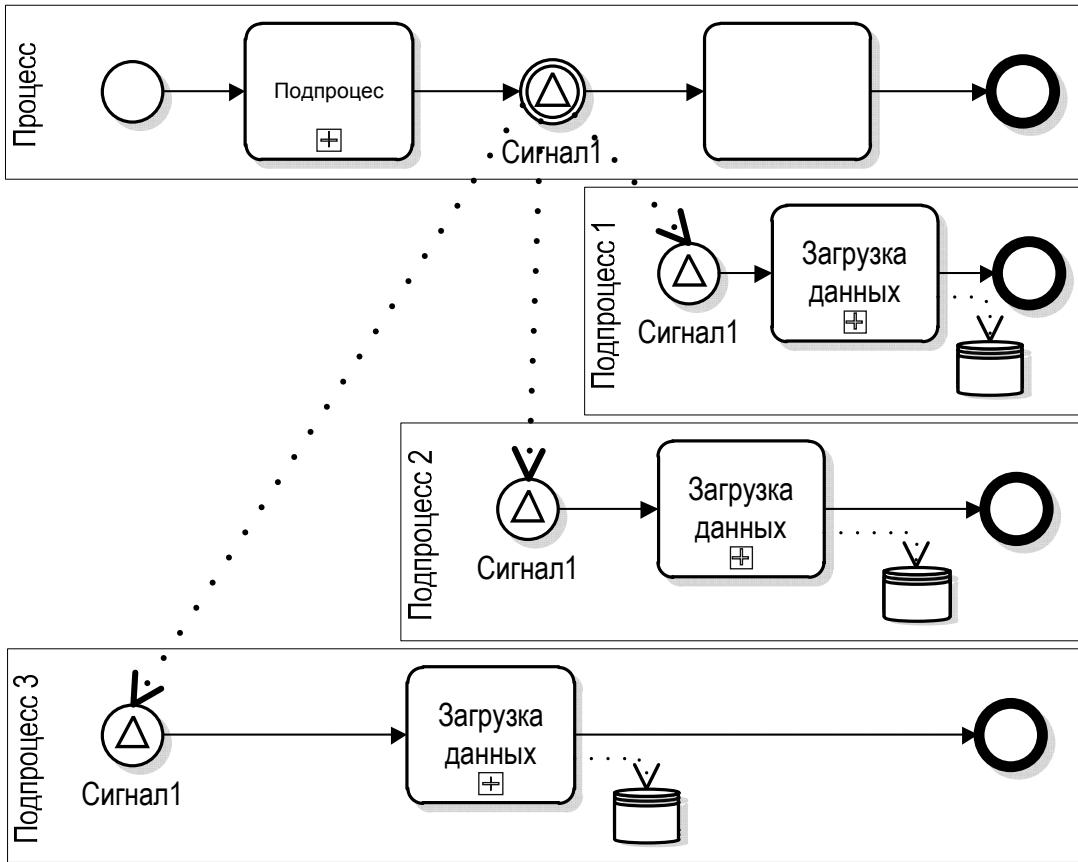


Рисунок 6-6. Пакетная обработка с использованием сигналов

Спецификация BPMN этого явно не определяет, но сигналы могут быть отправлены или получены из внешних ИТ систем, для этого служат специальные адаптеры к промежуточному ПО для рассылки сигналов, позволяющему приложениям, создавать и посыпать оповещения в BPMN, получать и читать обратные послания.

6.5. СООБЩЕНИЯ

Сообщения реализуют адресную передачу оповещения от источника к получателю, они не только извещают о факте наступления события, но так же позволяют передавать некоторый набор данных. Адресация получателя сообщения может осуществляться либо явно, с использованием графического элемента *поток сообщений*, образующих диалоги, либо неявно, с использованием механизма *корреляции* (см. п. 6.5.6).

6.5.1. ПОТОКИ СООБЩЕНИЙ

Процессы могут выполняться длительное время, возможно, параллельно друг другу, так что может потребоваться синхронизация их действий. Для этого можно использовать **потоки сообщений**. Сообщение есть механизм межпроцессной коммуникации, оно несет информационное наполнение и всегда пересылаются адресно между отправителем и получателем. Сообщения образуют информационные потоки между отправителем и получателем, которые изображаются пунктирными стрелками с небольшой полой окружностью у отправителя сообщения и полым треугольным указателем у получателя (см. Рисунок 6-7). Дополнительно на стрелке можно поместить символ конверта с подписью, при этом следует помнить, что мы показываем только заголовок конверта сообщения, а не его содержимое. Для сообщения можно специфицировать его внутреннюю информационную структуру, но она не м.б. визуально отображена на схеме процесса. Спецификация BPMN этого явно не определяет, но сообщения могут быть отправлены или получены из внешних ИТ систем, для этого служат специальные адаптеры к Java Message Service (JMS) – промежуточному ПО для рассылки сообщений, позволяющему приложениям, выполненным на платформе J2EE, создавать, посыпать, получать и читать сообщения.



Рисунок 6-7. Потоки сообщений

Не следует путать потоки сообщений и потоки управления. Первые предназначены, что бы показать порядок исполнения операций процесса, вторые служат, что бы показать об-

мен сообщениями. Первые целиком расположены внутри одного пула, вторые, напротив, показывают обмен только между разными процессами, иными словами, источник и получатель не могут принадлежать одному процессу.

6.5.2. ДИАЛОГИ

Потоки сообщений могут образовывать диалог между участниками, при этом различают сообщения инициировавшее диалог, оно не имеет заполнения (см. Рисунок 6-8А), и ответные сообщения, имеет заполнение серого цвета (см. Рисунок 6-8Б).



Рисунок 6-8. Инициирующее диалог и ответное сообщения

Два участника обмена сообщениями могут общаться по разным поводам, например, обсуждать финансовые условия, сроки и поставку, разные контракты, поэтому возникает желание, сгруппировать сообщения, касающиеся общего предмета переговоров. Диалог есть группа сообщений, объединяемая общей тематикой.

Рассмотрим пример, Поставщик и Покупатель ведут переговоры о нескольких контрактах, все переговоры, касательно одной сделки образуют диалог и можно предположить, что ключом, по которому стороны идентифицируют этот диалог, является номер контракта. Внутри диалога можно выделить поддиалоги, касающиеся разных аспектов условий, у них будет другой ключ (см. Рисунок 6-9). Т.о., диалоги м.б. иерархически вложенными, а ключи составными.

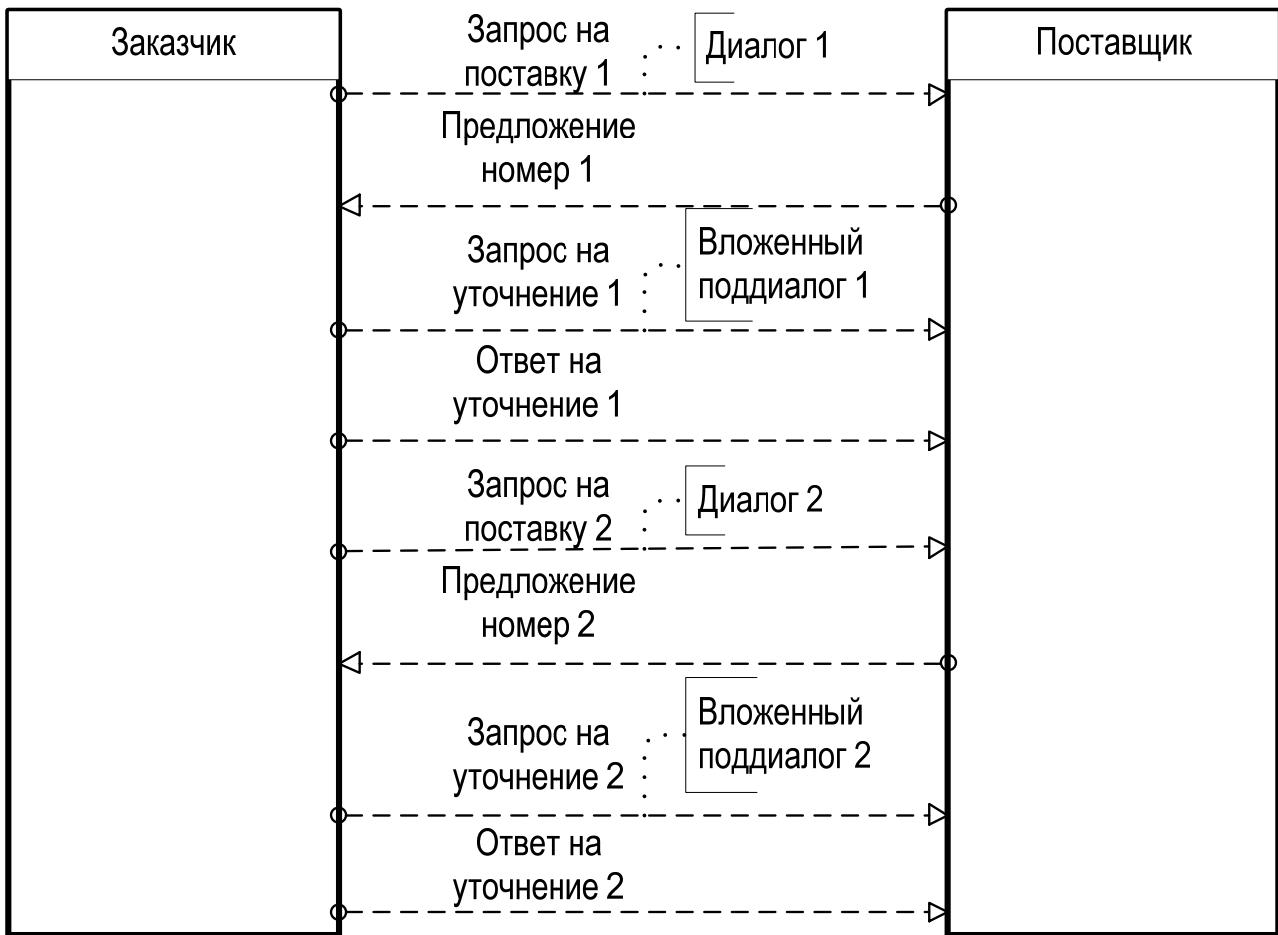


Рисунок 6-9. Диалог участников

Потоки сообщений используются на диаграммах: Оркестровки, Взаимодействия и Хореографии (см. Рисунок 6-10)

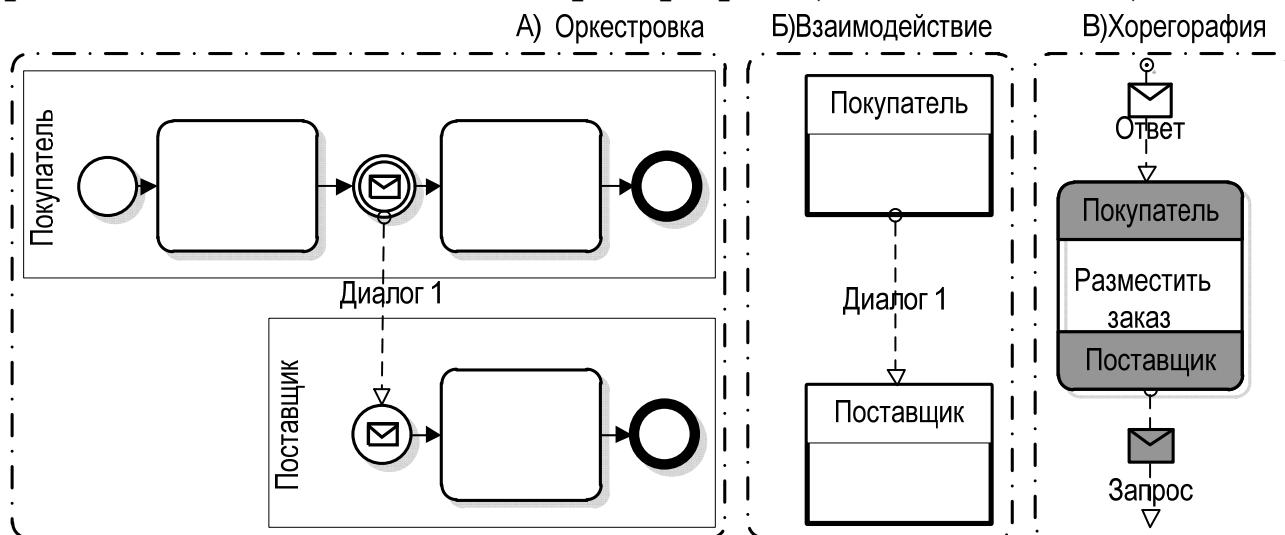


Рисунок 6-10. Потоки сообщений на диаграммах: Оркестровки, Хореографии, Взаимодействия

6.5.3. ОТПРАВКА И ПОЛУЧЕНИЕ СООБЩЕНИЙ

Обмен сообщениями осуществляется с использованием операций: отправить сообщение и принять сообщение или одноименных событий. Таблица 6-3 изображает Операция и событие для отправки и получения сообщения. Одноименные элементы операция и событие полностью взаимозаменяемы. Более того, на разных уровнях детализации процесса в одном случае элемент может изображаться как событие, а в другом, как операция. Например, если диаграмма приватного процесса поддерживает диаграмму публичного процесса, то на разных схемах одно и то же действие может отображаться по-разному.

Получение сообщения может инициировать запуск процесса на исполнение. Для этого стандарт разрешает использование как событие, так и одноименную задачу, принимающую сообщения, последняя имеет соответствующий значок окружности в верхнем левом углу.

Следует учитывать, что согласно спецификации BPMN 2.0 нельзя пересыпать сообщения внутри одного пула. Это накладывает ряд ограничений на моделирование оркестровки. Например, вложенный подпроцесс является частью пула и поэтому не может обмениваться сообщениями с родительским процессом. Одно из возможных решений вопроса заключается в использовании повторно используемых внешних подпроцессов вместо вложенных.

Таблица 6-3. Операция и событие для отправки и получения сообщения

	Отправить	Получить
Операция	 Отправить сообщение	 Принять сообщение
Событие	 Отправить сообщение	 Принять сообщение
Стартовать экземпляр процесса.	 Стартовать процесс	 Стартовать процесс

Пример (см. Рисунок 6-11) иллюстрирует обмен сообщениями между покупателем и поставщиком в ходе приобретения требуемого товара. Процесс начинается с получения сообщения от поставщика. После этого, покупатель начинается диалог по уточнению условий. Для этого используется промежуточное, генерирующее событие отправки сообщения. После того, как поставщик предоставит всю недостающую информацию, покупатель даст ответ на предложение.

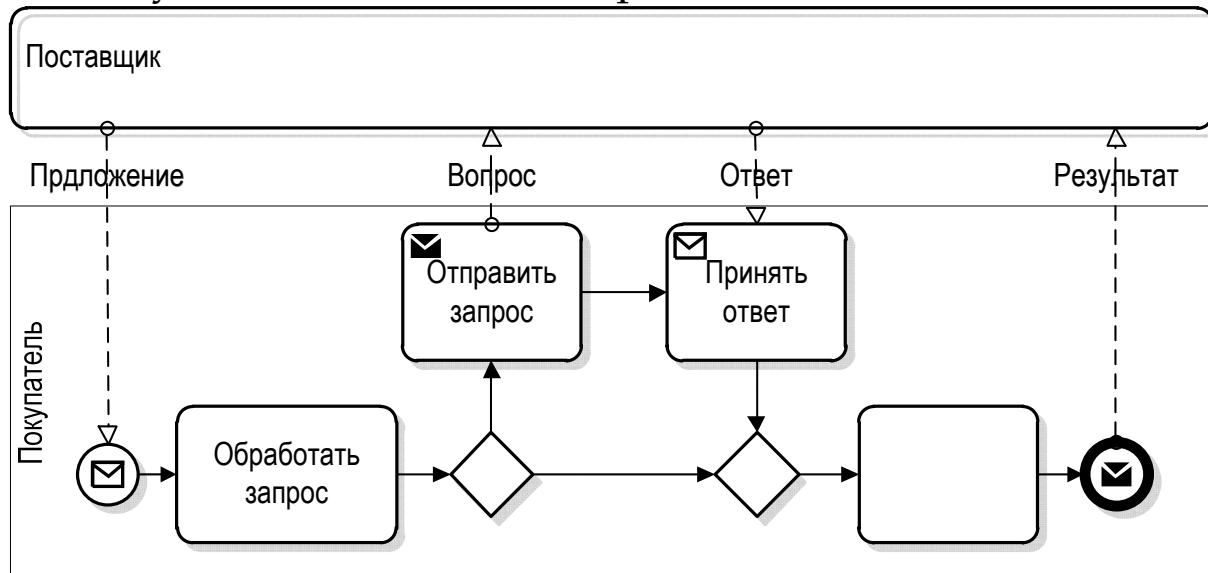


Рисунок 6-11. Обмен сообщениями между двумя участниками

6.5.4. СЕМАНТИКА ОПРАВКИ И ПОЛУЧЕНИЯ СООБЩЕНИЙ

Событие или операция, отправляющее сообщение всегда размещается в потоке управления, так что сразу по прибытии точки управления оно генерирует посылку, после чего управление сразу передается на следующую операцию. Операция, принимающая сообщение, может размещаться только в потоке управления. После прибытия в нее точки управления исполнение приостанавливается до тех пор, пока не будет получено соответствующее сообщение. Событие, принимающее сообщение, может размещаться в потоке управления, при этом оно ведет себя точно так же как и аналогичная операция. Если оно прикреплено к границе операции (подпроцесса), то в случае поступления сигнала создаст новый поток управления. Если обработчик события является прерывающим, то новый поток

управления выполняется вместо основного потока. Если обработчик события является непрерывающим, то новый поток управления выполняется параллельно с основным потоком.

6.5.5. ЯВНАЯ АДРЕСАЦИЯ ПОЛУЧАТЕЛЯ СООБЩЕНИЯ

При пересылке информационных сообщений используется принцип «точка-точка», то есть может быть только один адресант и один адресат. Адресатом является нужный экземпляр процесса. Иногда экземпляр получатель можно указать явно, используя потоки сообщений.

Пример (см. Рисунок 6-12) показывает подпроцесс, исполняемый в цикле, который инициирует требуемое число приглашений, для каждого поставщика в отдельности. Это приглашение инициирует требуемое число экземпляров процессов. Каждый поставщик возвращает подтверждение в процесс получатель, после чего исполнение завершается.

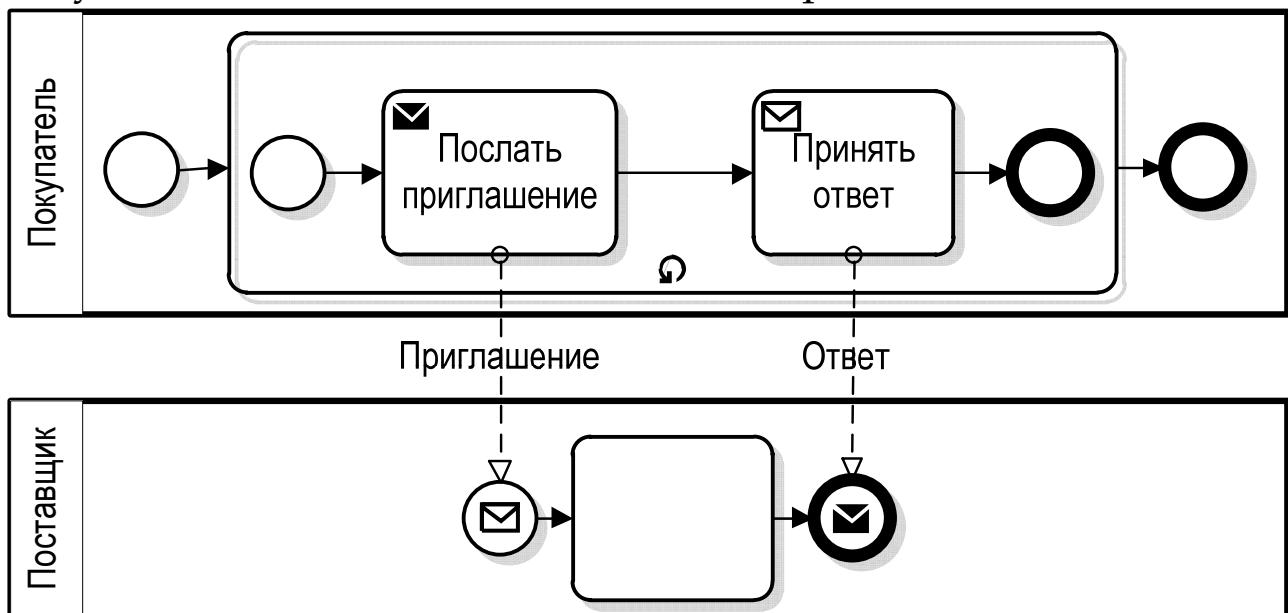


Рисунок 6-12. Явная адресация получателя сообщения

6.5.6. КОРЕЛЯЦИЯ - НЕЯВНАЯ АДРЕСАЦИЯ ПОЛУЧАТЕЛЯ СООБЩЕНИЯ

На схеме процесса не всегда удается явно указать получателя сообщения при помощи знака *поток сообщений*. В этих случаях

для адресации получателя сообщения используется неявная адресация, называемая корреляция.

Корреляция есть механизм, который помогает увязать отдельные события в диалоги, а диалоги привязать к парам процессов. Когда Сообщение отправляется от адресанта (отправителя) к адресату (получателю), необходимо уметь точно указать процесс получатель сообщения. Обычно для этого используется внутренний идентификатор процесса получателя. Однако разработчик процесса на стадии моделирования не знает заранее идентифицирующий номер получателя. В этой ситуации принято использовать механизм *корреляции*, который позволяет подменить индивидуальный идентификатор процесса на контекстную информацию, которая обрабатывается в процессе, например, значение какой-либо переменной процесса. Непременное условие, последняя должна принимать уникальные не повторяющиеся значения, которые позволяют однозначно идентифицировать экземпляр процесса. Например, переменные «номер заказа» или «номер клиента» могут однозначно идентифицировать экземпляр процесса. Напротив, фамилия заказчика не позволяет идентифицировать экземпляр процесса, поскольку она может повторяться.

Для идентификации получателя используется т.н. корреляционный ключ. Он м.б. сложносоставным, так что разные части ключа могут использоваться по-отдельности. Одна часть ключа, как уже указывалось, позволяет указать адресата сообщения, а вторая, позволяет связать сообщение с соответствующим диалогом. Как мы помним, процессы могут обмениваться потоками сообщений по разным темам, желательно разделить эти диалоги. При отправке сообщения системой автоматически генерируется корреляционный ключ, который не имеет определенного значения для бизнес аналитика. Этот ключ автоматически добавляется к сообщению, иницииющему диалог. Когда поступает ответ от получателя, он может содержать этот же ключ. Таким образом, отправитель может сопоставить полученный ответ со своим запросом.

6.6. НАЧАЛЬНЫЕ СОБЫТИЯ

Таблица 6-4 содержит краткий обзор начальных событий.

Таблица 6-4. Обзор начальных событий

СОБЫТИЯ	Верхнего уровня	Подпроцесс	
		Прерывающее	Не прерывающее
Простое	Не типизированное событие, показывает старт процесса верхнего уровня		
Сообщение:	Создает экземпляр процесса после получения сообщения из другого процесса.		
Таймер	Создает экземпляр процесса в назначенное время, либо в цикле, по расписанию.		
Эскалация	Создает процесс, который д.б. исполнен сотрудником с высоким уровнем полномочий		
Условие	Нельзя использовать данные экземпляра процесса, поскольку он еще не создан, доступны глобально известные переменные.		
Ошибка.	Обрабатывает заданный тип ошибок.		
Компенсация	Инициирует или обрабатывает откат транзакции		
Сигнал:	Создает экземпляр процесса после получения сигнала..		
Составное.	Для процесса определено несколько начальных событий, но для создания ЭП необходимо и достаточно совершения только одного		
Параллельное составное	Для создания ЭП должны наступить несколько событий.		

Начальное событие используется для описания процедуры создания экземпляра процесса. Первый столбец показывает события, применяемые для старта процесса верхнего уровня. Второй и третий столбцы показывают события, применяемые для старта событийного подпроцесса. Начальное событие является optionalным и в некоторых ситуациях может опускаться. Пример (см. Рисунок 6-13) показывает подпроцесс без стартового события, обе вложенные операции стартуют одновременно и выполняются параллельно.

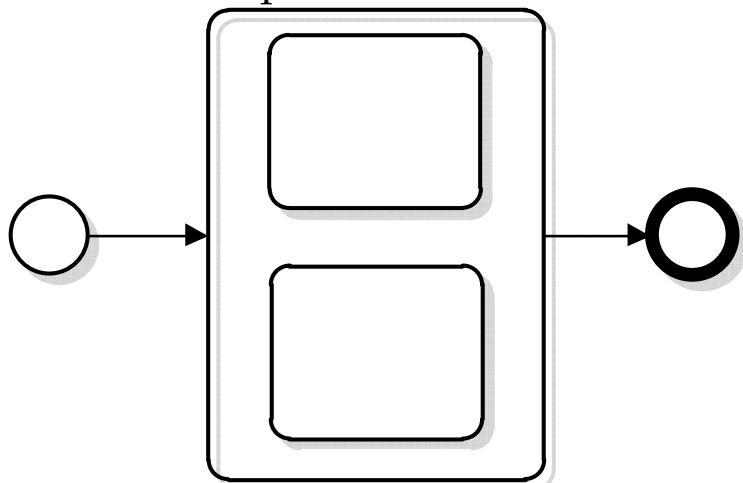


Рисунок 6-13. Подпроцесс без стартового события.

Спецификация допускает изображать на диаграмме несколько точек старта одного процесса, связывая каждую с отдельным событием (см. Рисунок 6-14). Наступление любого из этих событий порождает экземпляр процесса. Однако спецификация не указывает, что случится, если произойдут сразу оба события: неясно, нужно ли игнорировать второе событие, обрабатывать его в рамках ранее запущенного экземпляра или следует породить еще один экземпляр процесса. Поэтому спецификация BPMN не рекомендует использовать много точек старта, а вместо этого применять событийные операторы «ИЛИ» или «И» (см. далее п. 6.7).

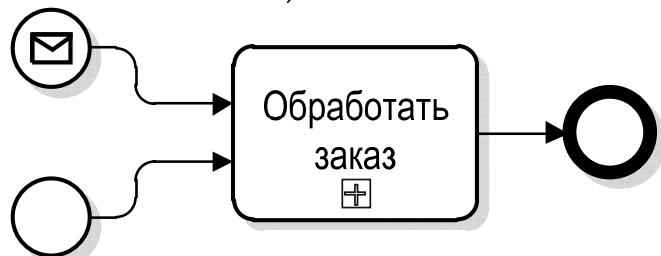


Рисунок 6-14. Процесс с несколькими точками старта

Исключением является подпроцесс, который в зависимости от обстоятельств может выступать либо как процесс верхнего уровня, либо как повторно используемый глобальный. Если хотя бы одно из них есть простое стартовое событие, этот процесс может рассматриваться, как глобальный и может быть вызван извне с помощью вызывающей операции. Рассмотрим типовые сценарии. Пример (см. Рисунок 6-16) показывает такой подпроцесс «двойного назначения».

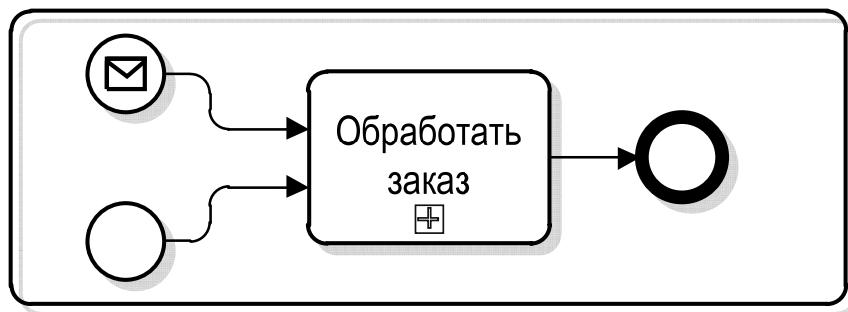


Рисунок 6-15. Процесс «двойного назначения»

6.7. СОСТАВНОЕ ВЗАИМОИСКЛЮЧАЮЩЕЕ СТАРТОВОЕ СОБЫТИЕ

Представим себе, что у процесса есть несколько точек старта, причем все они являются взаимоисключающими. Например, старт процесса продажи м.б. связан с заполнением клиентом формы на веб сайте компании, либо с его звонком в офис, важно, что эти события являются взаимоисключающими. Спецификация предлагает несколько способов решения этой задачи. **Составное взаимоисключающее стартовое событие** используется для запуска процесса от первого из перечисленных на схеме событий (см. Рисунок 6-16А). **Событийный логический оператор «Исключающее ИЛИ»** (см. Рисунок 6-16Б), позволяет добиться того же

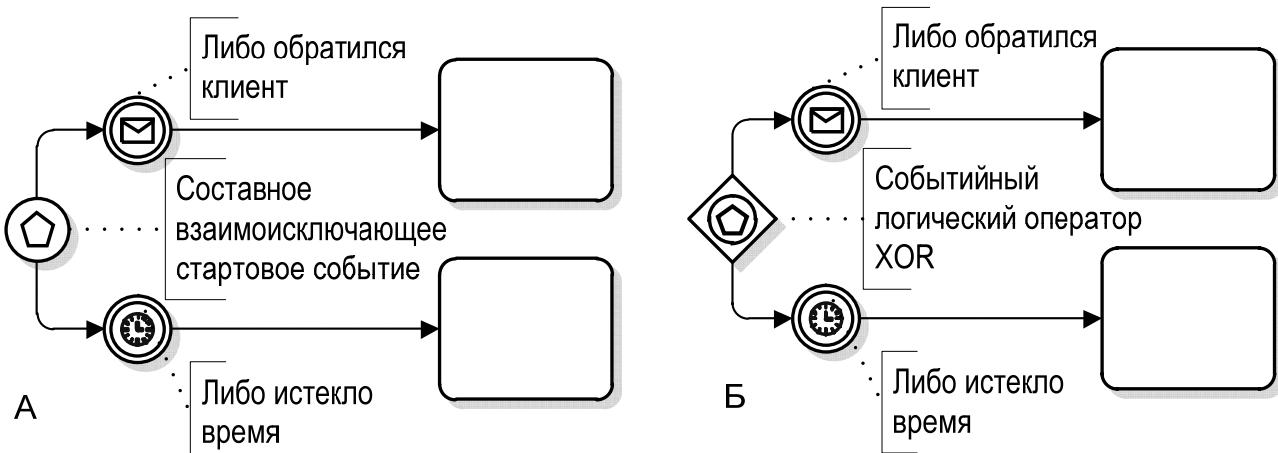


Рисунок 6-16. Пример взаимоисключающего старта

6.8. СОСТАВНОЕ ПАРАЛЛЕЛЬНОЕ СТАРТОВОЕ СОБЫТИЕ

Синхронный старт используется в том случае, если для создания ЭП должно произойти все из перечисленных начальных событий. **Составное параллельное стартовое событие** реализует логику «И» для нескольких событий, позволяя перечислить все нужные для запуска события. Старт произойдет, когда наступит последнее из ожидаемых событий.

Рассмотрим пример (см. Рисунок 6-17), иллюстрирующий создание процесса с использованием составного параллельного стартового события. Сразу же после старта процесс будет ожидать наступления всех из перечисленных промежуточных событий, определенных на схеме.

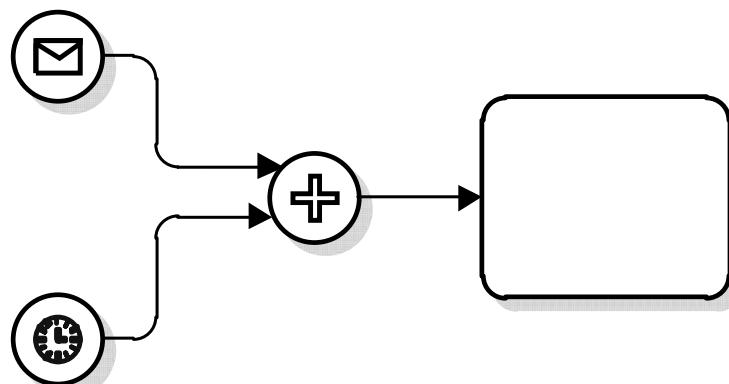


Рисунок 6-17. составное стартовое событие

6.9. СТАРТ СОБЫТИЙНОГО ПОДПРОЦЕССА

Событийный подпроцесс (см. 3.5.6) может иметь только одно стартовое событие следующих типов:

- сообщение,
- таймер,
- эскалация,
- условное,
- ошибка,
- компенсация,
- сигнал,
- составное стартовое событие,
- составное параллельное стартовое событие.

Следует обратить внимание, что стартовое событие м.б. прерывающим и непрерывающим (см. 6.2). Первые останавливают исполнение родительского процесса, а вторые нет.

Пример (см. Рисунок 6-18) иллюстрирует запуск событийного подпроцесса в случае наступления события-эскалация. Обрабатывающий событийный подпроцесс находится на том же уровне иерархии, что и основной процесс. Поскольку инициирующее событие является непрерывающим, подпроцесс будет исполняться параллельно с вызывающим.

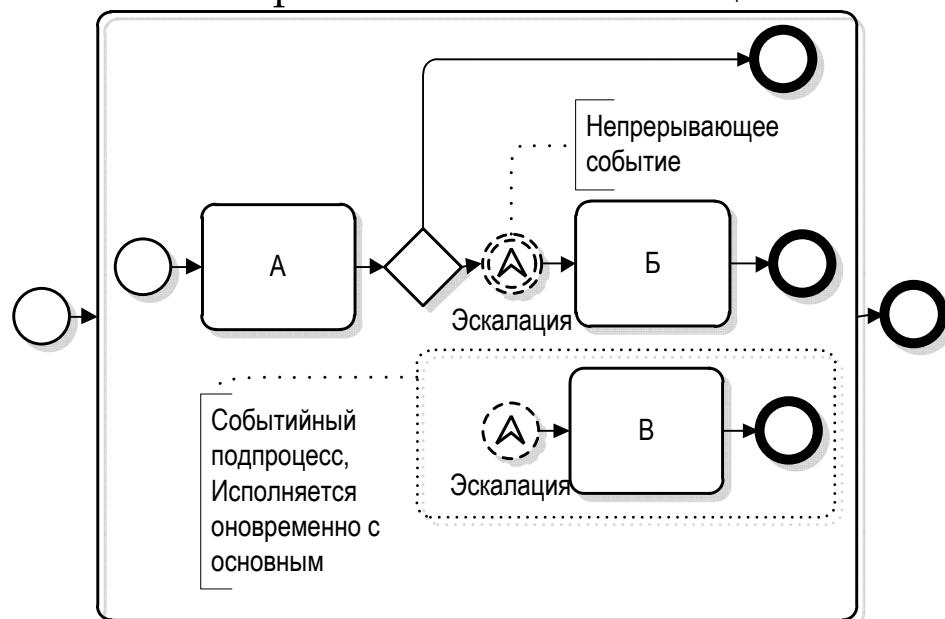


Рисунок 6-18. Событийный подпроцесс для события эскалация

Следует помнить, что *событие-ошибка* м.б. только прерывающим, оно останавливает работу операции, в которой произошел сбой. Напротив, *событие-компенсация* является не прерывающим.

6.10. ЗАВЕРШАЮЩИЕ СОБЫТИЯ

В спецификации BPMN определено 9 разных видов завершающих событий, все они являются генерирующими. Таблица 6-5 показывает свойства завершающих событий.

Таблица 6-5. Обзор завершающих событий

Наименование	Описание	Знак
Простое	Не обладает никакой дополнительной семантикой, кроме как завершение выполнения потока.	
Сообщение	Отправляет сообщение из завершаемого процесса в другой процесс.	
Эскалация	Текущий поток завершается со статусом эскалация, остальные потоки продолжают выполнение.	
Ошибка	Показывает, что процесс завершился с ошибкой. Оповещение имеет уникальное имя т.ч. обработчик реагирует только на определенный тип ошибки.	
Отмена	Используется в транзакционном процессе, обозначает отмену транзакции.	
Компенсация	Показывает, что все завершенные операции на данном уровне подпроцесса должны быть отменены, а данные возвращены в исходное состояние.	
Сигнал	Отправляет сигнал в момент завершения процесса. Сигнал принимают все процессы, в том числе изображенные на разных уровнях и в разных пулах.	
Составное	Завершение процесса связано с генерацией нескольких различных событий, например рассылкой сообщений разным адресатам и т.д.	
Прекращение	Все потоки, включая все дочерние параллельные, должны быть немедленно завершены. Процесс завершается без вызова компенсации или любого другого перехватчика событий.	

6.10.1. ПРОСТОЕ ЗАВЕРШАЮЩЕЕ СОБЫТИЕ

Простое завершающее событие позволяет отобразить на схеме процесса место завершается исполнение одного/нескольких потоков управления. Завершающее событие может иметь несколько входных потоков управления, но не может иметь ни одного выходного.

Экземпляр процесса считается завершенным только тогда, когда будут завершены все возникшие в нем потоки управления. Если модель допускает размножение потоков управления (см. п. 1.13.2), то завершение одного из них не завершает работы остальных. До тех пор, пока хотя бы один поток еще выполняется, процесс считается не завершенным.

Рассмотрим пример (см. Рисунок 6-19), изображающий процесс, который делится на две ветви, у каждой из них свое завершающее событие. Если выполнение верхней ветви завершится первой, то это не остановит исполнение нижней ветви. Процесс закончится только тогда, когда будут окончены обе ветви.

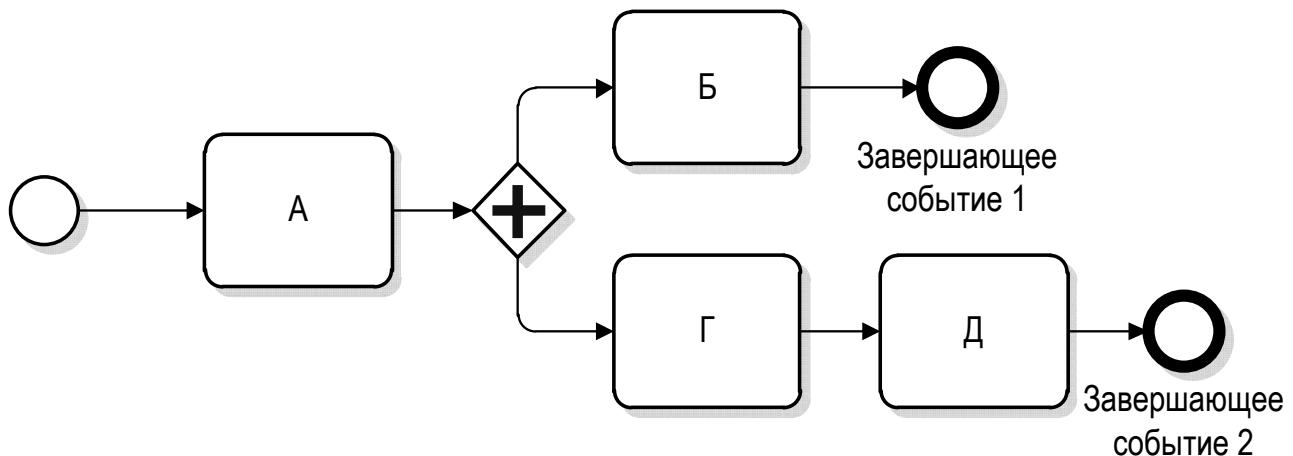


Рисунок 6-19. Простое завершающее событие

6.10.2. СОБЫТИЕ-ПРЕКРАЩЕНИЕ

Возможны ситуации, когда завершение одной ветви должно прекратить исполнение других существующих потоков управления. **Завершающее событие-прекращение** останавливает исполнение всех существующих потоков управления, которые существуют на том же уровне, включая всех вложенные подпроцессы, циклические и многопоточные операции.

Рассмотрим пример (см. Рисунок 6-20), иллюстрирующий завершение процесса с помощью *простого события* и *событие-прекращение*. Главный процесс инициирует событийный подпроцесс, а затем в цикле исполняет второй подпроцесс «Обработать заказ». Если операция «Проверить оплату» будет нормально завершена, то главный процесс завершится после того, как «Обработка заказа» будет выполнена. Но если в ходе проверки оплаты обнаружится, что средства не поступили, выполнение всего процесса будет полностью остановлено. При этом, будут завершены: параллельные ветви, вложенные подпроцессы, циклические и многопоточные операции. Таким образом, *событие-прекращение* остановит работу запускаемого в цикле вложенного подпроцесса обработать заказ.

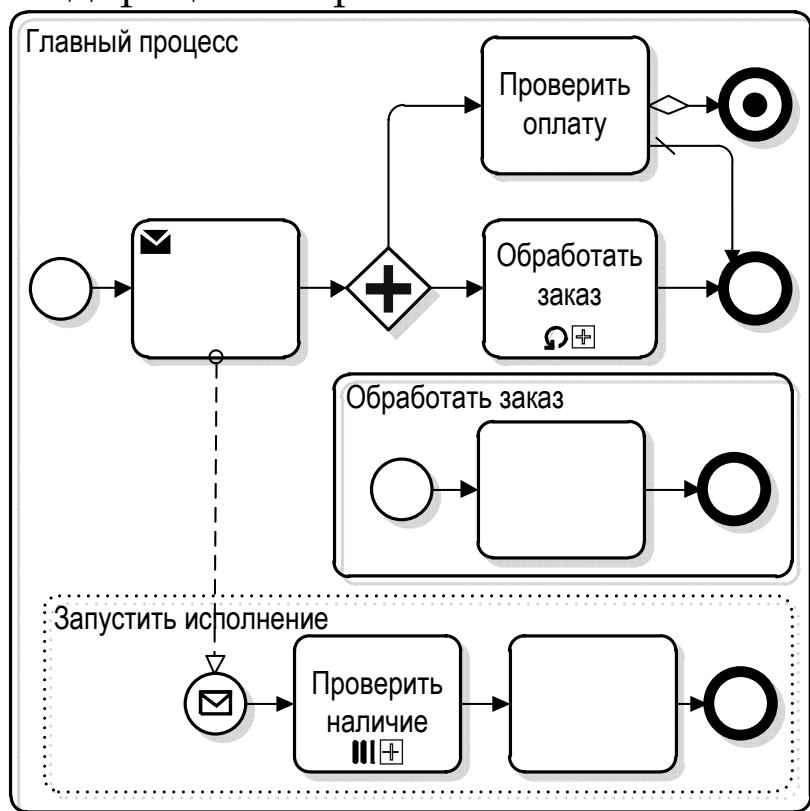


Рисунок 6-20. Событие-прекращение

6.10.3. ЗАВЕРШАЮЩЕЕ СОБЫТИЕ-СООБЩЕНИЕ

Завершающее событие-сообщение используется для запуска следующего процесса в цепочке. Представим себе, что сквозной процесс разделен на цепочку взаимодействующих подпроцессов. Один из эффективных способов связывания отдельных

подпроцессов заключается в отправке сообщения, инициирующего следующий процесс. Т.о. завершающее генерирующее событие-сообщение связывается диалогом с обрабатывающим стартовым событием-сообщением. Учитывая, что сообщения несут полезную информационную нагрузку, удается передать на вход следующего всю нужную информацию.

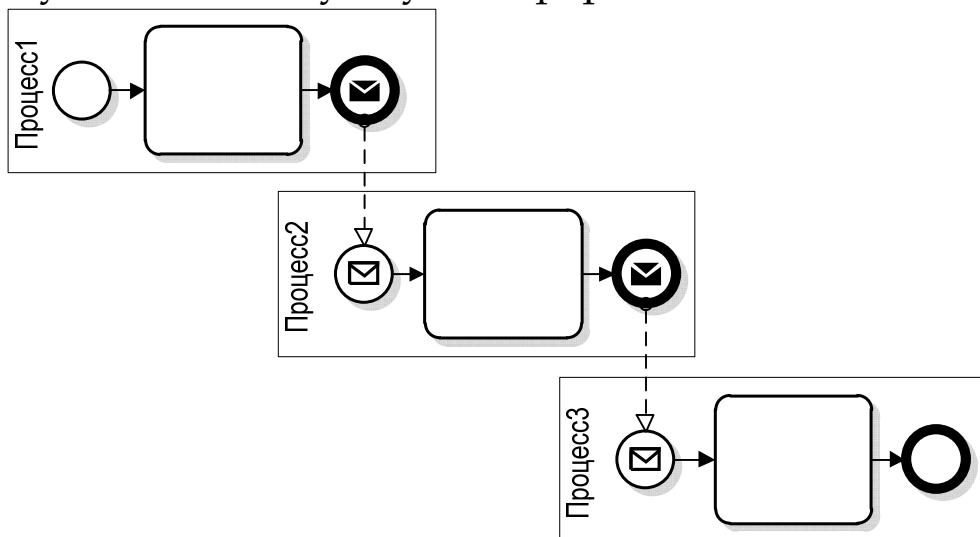


Рисунок 6-21. Завершающее событие-сообщение

6.11. СОБЫТИЯ, УСТАНАВЛИВАЮЩИЕ СТАТУС ЗАВЕРШЕНИЯ ПОДПРОЦЕССА

В программировании, когда одна программа вызывает подпрограмму, она хочет знать, чем завершается вызываемая процедура. Для этого последняя возвращает в вызывающую статус своего завершения. При моделировании бизнес-процессов есть два способа передать статус завершения. Во-первых, можно установить пользовательский статус завершения явно, например, присвоив какой либо пользовательской переменной заранее оговоренное значение. Во-вторых, можно возвратить статус, воспользовавшись одним из нижеперечисленных завершающих событий:

1. Ошибка.
2. Эскалация.
3. Отмена
4. Компенсация .

События, устанавливающие статус завершения используются только совместно с соответствующими обрабатывающими граничными событиями (см. 6.14). Каждое событие имеет уникальное имя, таким образом, в процессе м.б. несколько окончательных событий-ошибки, их всегда можно различить, по имени.

6.11.1. СОБЫТИЕ-ОШИБКА

Событие-ошибка используется, что бы передать в вызывающий процесс информацию о произошедшей ошибке. Оповещение имеет уникальное имя т.ч. обработчик среагирует только на ошибки определенного типа.

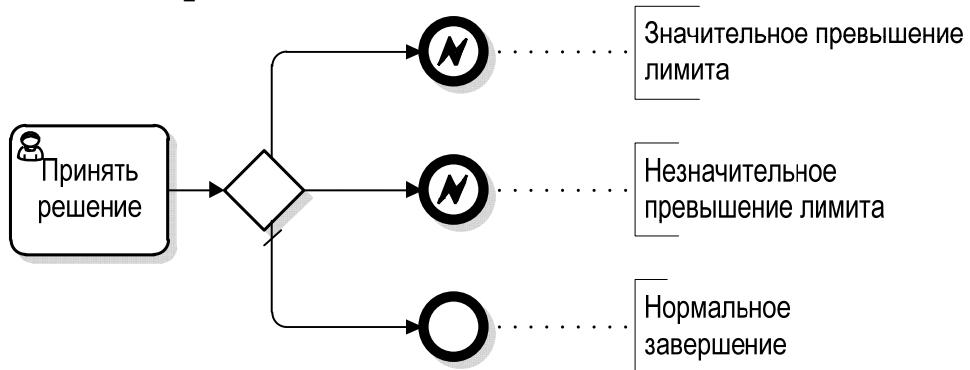


Рисунок 6-22. Завершающее событие-ошибка

6.11.2. СОБЫТИЕ-ЭСКАЛАЦИЯ

Событие-эскалация отмечает ситуацию, когда работа не м.б. выполнена, поскольку у сотрудника отсутствуют полномочия, необходимые для ее выполнения. Работа д.б. поручена сотруднику с большим лимитом ответственности. Пример (см. Рисунок 6-23) иллюстрирует операцию, связанную с принятием решения. Если сотрудник в состоянии принять решение, то происходит нормальное завершение. Если же не хватает полномочий, он выбирает продолжение, связанное с эскалацией.

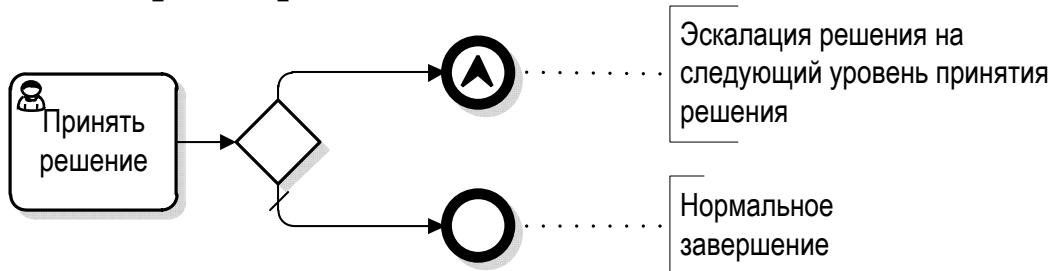


Рисунок 6-23. Завершающее событие-эскалация

6.11.3. СОБЫТИЕ-ОТМЕНА

Событие-отмена используется только внутри транзакционного процесса, что бы показать, что транзакция должна была отменена. Посыпает оповещение прикрепленному промежуточному *событию-отмене*.

Событие-отмена завешает все потоки внутри исполняющейся транзакции. Оно всегда связано с некоторой подпрограммой отката транзакций, которая либо определена пользователем явно, либо выполняет набор действий по умолчанию. В последнем случае будут последовательно, но в обратном порядке вызываться программы отката данных для каждой из операций, входящих в состав процесса.

Пример (см. Рисунок 6-24) иллюстрирует завершение процесса с помощью *события-отмена*. Главный процесс является транзакционным, об этом свидетельствует двойная линия границы. Процесс имеет две точки завершения. Первая – простое завершение, соответствует нормальному сценарию, после этого управление возвращается в вызывающий процесс. Вторая точка завершения это событие-отмена, исполнение транзакционного процесса завершается, управление передается на операцию «Оповестить клиента». В этом примере все завершающие события являются взаимоисключающими.

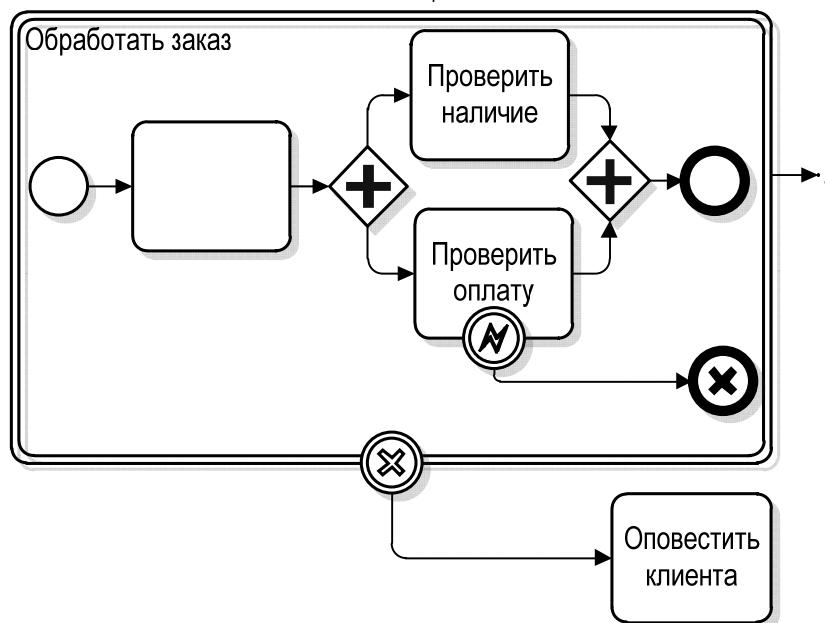


Рисунок 6-24. Пример завершение процесса

6.11.4. СОБЫТИЕ-КОМПЕНСАЦИЯ

Событие-компенсация показывает, что все завершенные операции на данном уровне подпроцесса должны быть отменены, а данные возвращены в исходное состояние.

Пример (см.) изображает автоматическую операцию. Если операция завершится ошибкой, то управление будет передано на завершающее событие-компенсация. Оно выработает оповещение, которое вызовет откат данных в других операциях, помеченных знаком компенсации.

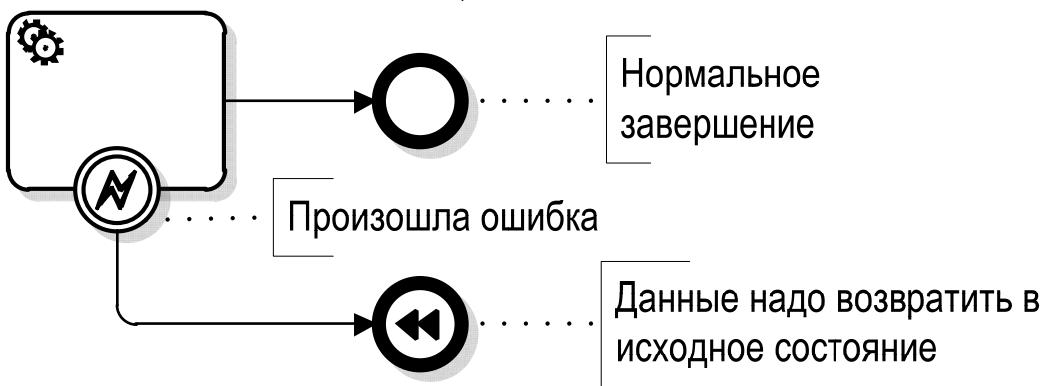


Рисунок 6-25. Завершающее событие-компенсация

6.12. ПРОМЕЖУТОЧНЫЕ СОБЫТИЯ

Промежуточное событие располагается на диаграмме процесса между начальным и конечным событиями, оно используются для обработки исключительных ситуаций и для синхронизации отдельных ветвей данного процесса или разных процессов. Промежуточное событие может использоваться для моделирования следующих ситуаций:

- Получения входящих или отправки исходящих сообщений внутри процесса;
- Отображения временных задержек при выполнении процесса;
- Отображение дополнительных операций, откатывающих данные для незавершенных транзакций;

Существует два способа использования промежуточных событий:

- 1 Обработчик события размещается в потоке управления процесса и может использоваться для целей генерации

- или обработки события.
- 2 Обработчик события прикрепляется к границе операции и может использоваться только для обработки события. Поведение генерирующих и обрабатывающих промежуточных событий, размещаемых в потоке, различается.

6.12.1. ПРОМЕЖУТОЧНЫЕ СОБЫТИЯ, РАЗМЕЩАЕМЫЕ В ПОТОКЕ УПРАВЛЕНИЯ ПРОЦЕССА

Когда поток управления достигает *генерирующего события* размещенного в потоке, то немедленно происходит обработка этого события (отправляется сообщение, сигнал и т.д.) после чего поток управления немедленно покидает текущий элемент и продолжает движение далее по процессу.

Когда поток управления достигает знака *обрабатывающего события*, исполнение останавливается до тех пор, пока не произойдет соответствующее событие (например, получено сообщение, сигнал и т.д.). После этого поток управления покидает элемент обработки события. Спецификация не указывает явно, что произойдет, если событие произойдет до того, как поток управления прибудет в обрабатывающее событие. В большинстве случаев такое событие не будет обработано и будет потеряно.

Промежуточное событие влияет на выполнение только текущего потока процесса, оно не может создать новый экземпляр процесса или прекратить выполнение текущего.

Таблица 6-6 дает краткую характеристику различным обработчикам событий, которые могут размещаться непосредственно в потоке управления процесса.

Таблица 6-6. Обработчики событий, размещаемые в потоке управления

СОБЫТИЯ	ОПИСАНИЕ	Поток	
		Генерирующее	Обрабатывающее
Простое	Обработчик отсутствует. Используется для отображения статуса исполнения.		
Сообщение.	Используется для обмена сообщениями между процессами, работает по принципу «один адресат - один адресант».		
Таймер	Приостанавливает выполнение экземпляра процесса. Таймер может только обрабатывать событие, но не инициировать его.		
Эскалация	Изменяет уровень принятия решения, может быть только генерирующими.		
Условное	Условие может быть любым, в том числе составным и содержать в себе другие обработчики событий, которые не могут быть визуализированы на схеме процесса.		
Ссылка	Связывает две точки процесса и подразумевает неявную передачу управления. Улучшает читаемость схемы процесса.		
Ошибка	Обрабатывает заданный тип ошибок.		
Отмена.	Инициирует отмену транзакции		
Компенсация	Используется для отката данных. Может быть только генерирующими		
Сигнал	Широковещательная связь между процессами и подпроцессами на разных уровнях.		
Составное	Обработка одного события из множества или генерация всех определенных событий.		
Параллельное составное	Обработка всего множества параллельных событий		

6.12.2. ПРОСТОЕ ПРОМЕЖУТОЧНОЕ СОБЫТИЕ

Простое промежуточное событие используется, чтобы отобразить на схеме процесса этап обработки некоторого объекта. Этап связан с получением объектом определенного статуса.

Пример (см. Рисунок 6-26) показывает простое промежуточное событие, размещаемое в потоке. Оно отображает статус исполнения процесса, но не позволяет изобразить состояние объекта процесса, они отображаются подписями к переходам управления.

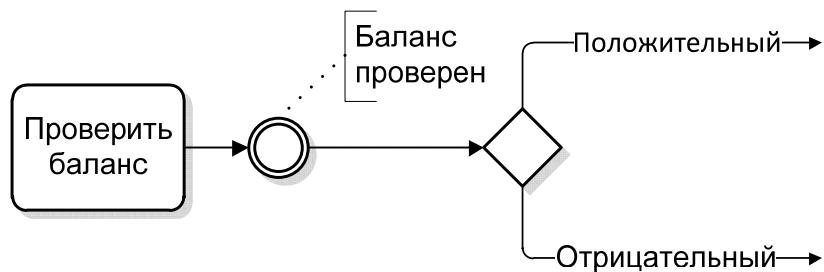


Рисунок 6-26. Простое промежуточное событие, размещаемое в потоке

6.12.3. ПРОМЕЖУТОЧНОЕ СОБЫТИЕ ДЛЯ РАБОТЫ С СООБЩЕНИЯМИ

Промежуточные события отправить и принять сообщения используются в основном для синхронизации двух процессов. Пример (см. Рисунок 6-27) показывает два промежуточных события для работы с сообщениями, размещенные в потоке. Первое отправляет сообщение, а второе ждет получения ответа. Таким образом, два процесса синхронизируют свою работу.

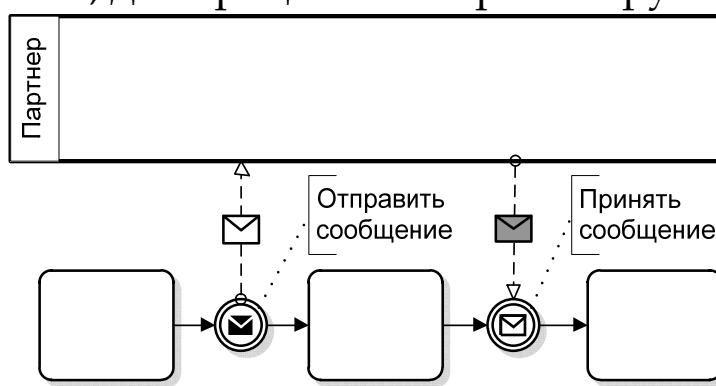


Рисунок 6-27. События-сообщения, размещенные в потоке

6.12.4. СОБЫТИЕ-ТАЙМЕР

Промежуточное событие-таймер позволяет определить задержку исполнения. Пример (см. Рисунок 6-26) иллюстрирует использование промежуточного *события-таймер*, размещенного в потоке, который выполняет роль элемента задержки.

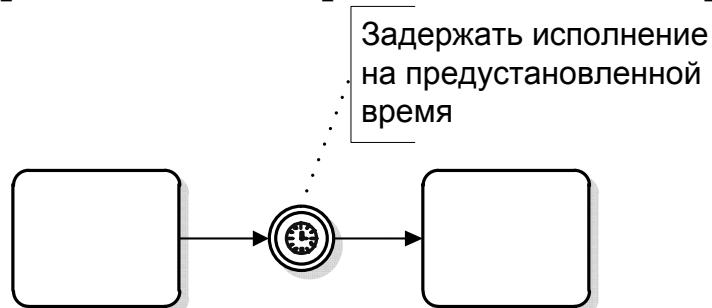


Рисунок 6-28. Промежуточное событие-таймер

6.12.5. ПРОМЕЖУТОЧНОЕ СОБЫТИЕ-УСЛОВИЕ

Промежуточное событие-условие используется для маршрутизации потока управления. Пример (см. Рисунок 6-29А) иллюстрирует использование промежуточного *события-условие*, размещенного в потоке. Пример (см. Рисунок Б) показывает логический оператор «исключающее ИЛИ». Различие между двумя вариантами в том, что логический оператор проверяется один раз, когда тока управления достигнет узла. *Событие-условие* задержит исполнение до тех пор, пока заданное условие не будет выполнено.

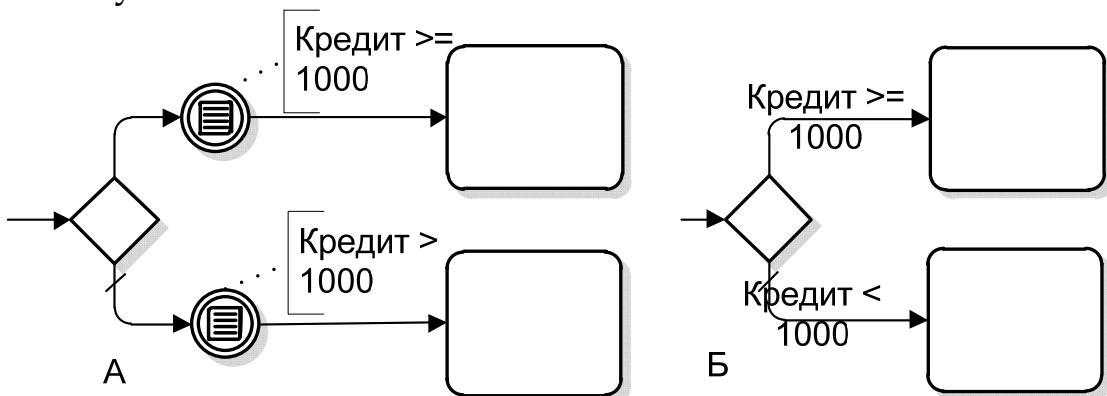


Рисунок 6-29. Промежуточное событие-условие

6.12.6. ПРОМЕЖУТОЧНОЕ СОСТАВНОЕ СОБЫТИЕ

Промежуточное составное событие обрабатывает первое из списка ожидаемых событий, остальные игнорируются. Пример (см. Рисунок 6-30) изображает процесс, который приостановлен до наступления любого из двух ожидаемых событий. Исполнение будет продолжено, если обратится клиент, либо закончится срок ожидания.

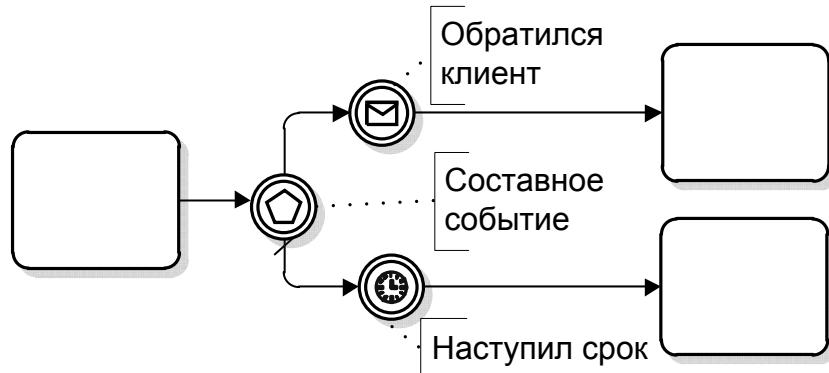


Рисунок 6-30. Промежуточное событие-условие

6.12.7. ПРОМЕЖУТОЧНОЕ СОСТАВНОЕ ПАРАЛЛЕЛЬНОЕ СОБЫТИЕ

Промежуточное составное параллельное событие срабатывает при наступлении всех событий из списка ожидаемых. Пример (см. Рисунок 6-30) изображает процесс, который приостановлен до срабатывания обоих ожидаемых событий. Исполнение будет продолжено, если обратится клиент, либо наступит крайний срок.

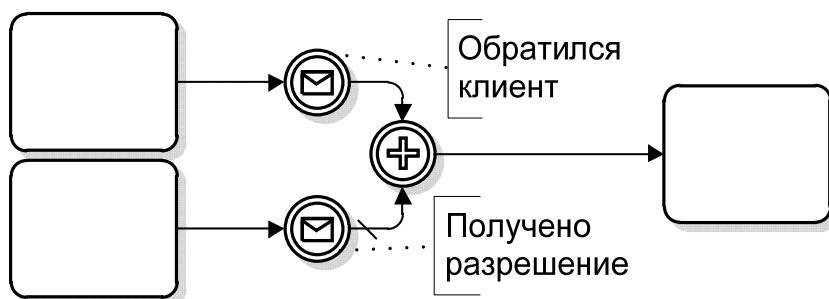


Рисунок 6-31. Промежуточное составное параллельное событие

6.12.8. СОБЫТИЕ-ССЫЛКА

Событие-ссылка является единственным графическим элементом в данной категории, которое не отображает какое-либо явление или объ-

екты внешнего мира. Оно используется для передачи управления между двумя операциями на диаграмме процесса, расположеными далеко друг от друга на диаграмме процесса. По сути, оно является аналогом стрелки потока управления, но используется в случае, когда изобразить переход, по каким-то причинам, затруднительно. События-ссылки являются промежуточными и используются только для соединения частей одного процесса, расположенных внутри одного пула. Их нельзя применять для отображения межпроцессного взаимодействия (см. Рисунок 6-32). Событие ссылка имеет имя, т.ч. допускается несколько разных ссылок на одной схеме.

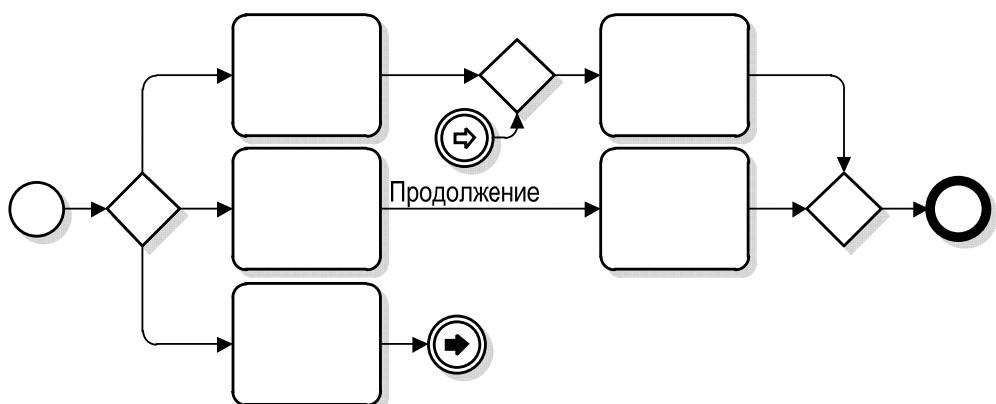


Рисунок 6-32. Событие-ссылка.

6.13. СОБЫТИЯ, ПРИКРЕПЛЯЕМЫЕ К ГРАНИЦАМ ОПЕРАЦИЙ

События, прикрепляемые к границам операций, бывают только обрабатывающими. Наступление события изменяет нормальный сценарий исполнения, возникает дополнительный поток управления, на который указывает обработчик события. Дальнейшее выполнение операции, к которой прикреплено событие, зависит от типа прикрепленного события: прерывающее - приостанавливает ее исполнение, а непрерывающее продолжает её работу. После окончания дополнительного потока управление м.б. возвращено в прерванную операцию. Рассмотрим более подробно прикрепленные события (см. Таблица 6-7).

Таблица 6-7. События, прикрепляемые к операции

СОБЫТИЯ	Описание	Обраба- тывающ-	
		Прерывающее	Непрерывающее
Сообщение	Срабатывает при получении сообщения.		
Таймер	Срабатывает один раз в определенный момент времени либо несколько раз по расписанию.		
Эскалация	Изменяет уровень принятия решения, может быть только генерирующими.		
Условие	Условие может быть составным и содержать в себе обработчики событий, которые не могут быть визуализированы на схеме процесса.		
Ошибка	Перехватывает именованные и безымянные ошибки. Прерывает выполнение операции, к которой прикреплен обработчик.		
Отмена	Используется в транзакционных процессах, всегда прерывает выполнение всех операций.		
Сигнал	Принимает широковещательное оповещение. Отличается от Ошибки более широким диапазоном использования.		
Составное	Перехватывает сложные составные события, при этом достаточно, чтобы произошло хотя бы одно событие из списка.		
Параллельное составное	Перехватывает сложные составные события, необходимо, чтобы произошли все события из списка.		

6.14. РЕТРАНСЛЯЦИЯ СОБЫТИЯ

Рассмотрим более подробно семантику исполнения события, прикрепленного к границе операции, оно выполняет задачу ретрансляции факта инцидента, произошедшего внутри операции наружу в процесс.

Пример (см. Рисунок 6-33.) показывает операцию с прикрепленным к границе событием-ошибкой. Это событие «слушает», что происходит внутри операции. Допустим операция автоматическая и вызывает веб-сервис. Если последний завершится ошибкой, прикрепленное событие сможет ретранслировать эту ошибку в процесс, что бы обработать ее средствами BPMS.

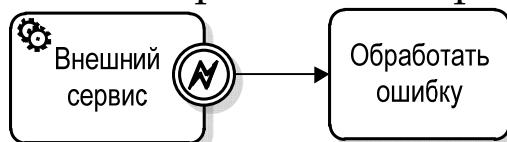


Рисунок 6-33. Ретрансляция ошибки из операции

Пример (см. Рисунок 6-34)

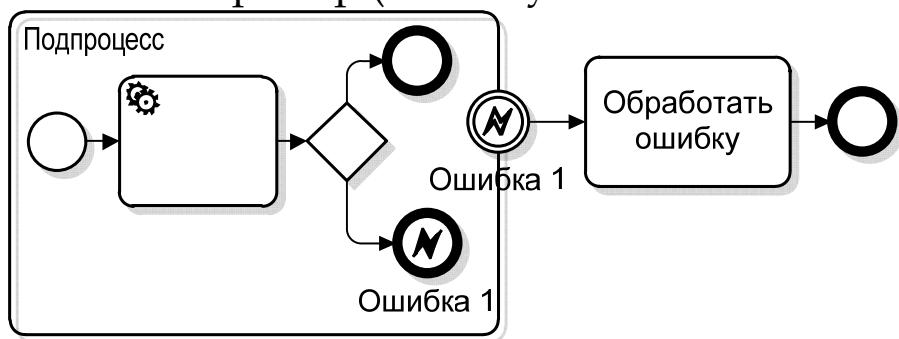


Рисунок 6-34) показывает процесс с двумя вариантами завершения, одна ветка заканчивается *событием-завершение*, а вторая – генерирующим *событием-ошибки*. Прикрепленное к границе подпроцесса обрабатывающее событие-ошибки перехватывает внутреннюю ошибку и передает ее на обработку вне подпроцесса.

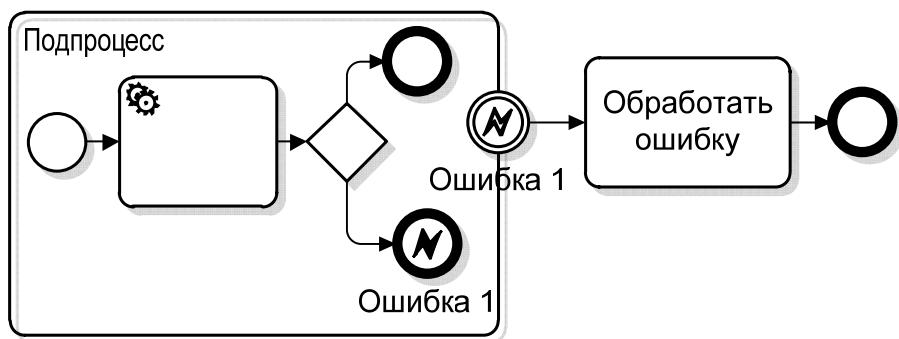


Рисунок 6-34. Ретрансляция события

6.14.1. ОБРАБОТКА ПРЕРЫВАЮЩИХ И НЕПРЕРЫВАЮЩИХ СОБЫТИЙ

Семантика выполнения прерывающих и не прерывающих обрабатывающих граничных событий сильно отличается между собой. Рассмотрим данный вопрос более подробно.

Прикрепленное прерывающее обрабатывающее граничное событие, вызывает аварийное прекращение соответствующей операции. В этот момент управление передается другой операции, которая связана с обработчиком безусловным переходом, происходит передача управления на **поток обработки исключительной ситуации**. После выполнения всех действий по ликвидации последствий аварии, новый поток можно либо завершить и тем самым остановить процесс. Либо можно вернуть управление в прерванную операцию.

Прикрепленное непрерывающее обрабатывающее граничное событие не останавливает выполнение соответствующей операции. В этом случае генерируется *дополнительный* поток управления, который выполняется параллельно с основным маршрутом процесса. Необходимо следить за логикой выполнения потоков, которые были созданы обработчиком не прерывающего события. Такие потоки могут либо сливаться с основным маршрутом, либо завершаться отдельно.

6.14.2. ПРЕРЫВАЮЩЕЕ СОБЫТИЕ-ТАЙМЕР

Прикрепленное прерывающее граничное событие-таймер используется, что бы задать нормативную длительность выполнения операции.

Рассмотрим пример использования прерывающего *события-таймер* (см. Рисунок 6-35). Истек отведенный срок исполнения задания. Таймер срабатывает и создает новый поток управления, направленный операцию обработать таймаут. При этом основной поток прерывается, управление в него больше не возвращается. Т.о., прерывающее событие ведет себя как ЛО «исключающее ИЛИ». Затем оба потока объединяются на логическом операторе («исключающее ИЛИ»).

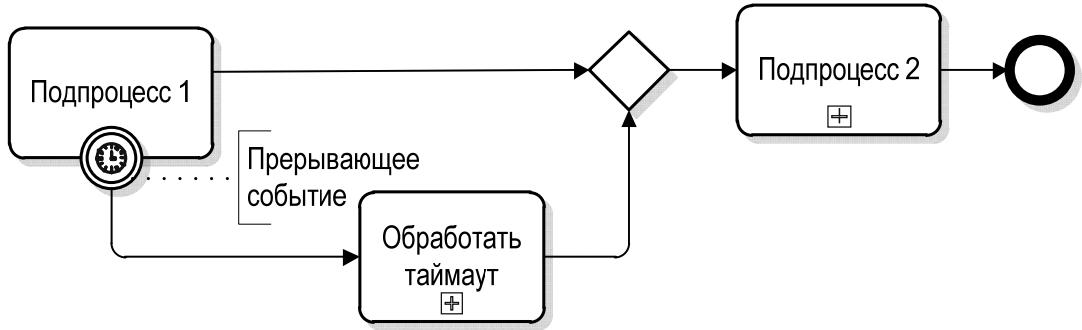


Рисунок 6-35. Обработка прерывающего прикрепленного события

6.14.3. НЕПРЕРЫВАЮЩЕЕ СОБЫТИЕ-ТАЙМЕР

Прикрепленное непрерывающее граничное событие-таймер используется, что бы оповестить других участников об истечении нормативной длительности выполнения операции.

Непрерывающее событие-таймер (см. Рисунок 6-36) инициирует альтернативный поток, но не прекращает основной. Например, необходимо информировать руководителя о задержке исполнения, при этом, работа исполнителя не останавливается. В случае непрерывающего события следует следить, что бы дополнительный альтернативный поток был явно завершен в нужное время.

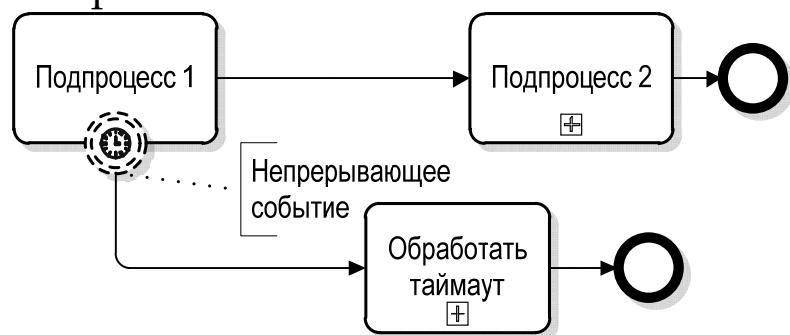


Рисунок 6-36. Обработка непрерывающего прикрепленного события-таймер

Третий пример (см. Рисунок 6-37) показывает ситуацию, когда обработка таймаута осуществляется в событийном вложенном подпроцессе. Значок таймера в левом верхнем углу последнего указывает, что стартовое событие обрабатывает оповещение от таймера.

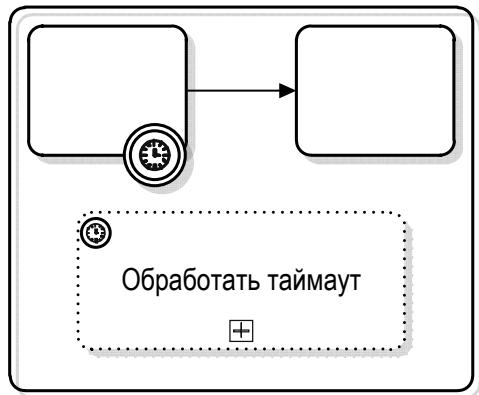


Рисунок 6-37. Непрерывающее прикрепленное событие-таймер

Следует учитывать, что непрерывающие прикрепленные обработчики событий в момент своего выполнения порождают новый поток обработки. После завершения обработки данный поток необходимо завершить. Как правило, это происходит за счет отдельного завершающего события.

7. ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ

Исключительные ситуации возникают всякий раз, когда нормальный ход выполнения процесса необходимо временно приостановить. Так же как и в программировании, под исключительной ситуацией в бизнес-процессе понимают возникшую ошибку или бизнес-исключение, которые делают дальнейшее исполнение невозможным или бессмысленным. Обработка исключительной ситуации - это механизм, предназначенный для описания реакции системы на исключения и ошибки выполнения. При этом нормальный ход выполнения процесса прерывается, управление передается специальному подпроцессу, который выполняет обработку возникшей исключительной ситуации. После завершения обработки, управление может быть возвращено в прерванный процесс.

7.1. КЛАССИФИКАЦИЯ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

Исключительные ситуации можно классифицировать по разным принципам, рассмотрим примеры:

1. По месту возникновения:

- В операции:
 - a. Автоматическая операция обращается к внешнему сервису, а он оказался недоступен.
 - b. При выполнении интерактивной операции не удается подобрать исполнителя, соответствующего заданным критериям отбора.
- В подпроцессе или процессе:
 - c. Завершение процесса с определенным статусом, требующим специальных мер обработки.
 - d. Истечание времени, отведенного на исполнение отдельной операции, этапа или всего процесса целиком.
- Во внешней среде:

Заказчик отменил, ранее сделанный заказ. Стоит ли продолжать исполнение процесса, если его результат уже никому не нужен.

- В системе:

Ошибка в движке ВРМ или в одной из библиотек программ.

2. По источнику возникновения:

- Системные, как видно из названия, генерируются системой в результате ошибок и отказов;
- Бизнес исключения м.б. заложены в бизнес-логику или являются результатом действий оператора.

3. По времени возникновения:

- Синхронные - происходят, когда процесс находится в определенном состоянии. Например, таймаут сервиса возможен только пока выполняется операция;
- Асинхронные – происходят на любой стадии исполнения. Например, клиент может аннулировать заказ в любое время;

4. По влиянию на текущий процесс:

- Прерывающие, исполнение процесса приостанавливается до окончания процедуры обработки;
- Не прерывающие, исполнение процесса не приостанавливается, возникает дополнительный поток управления;

5. По возврату в операцию, инициировавшую исключительную ситуацию:

- С возвратом в точку останова;
- Без возврата.

6. По необходимости отката данных:

- С компенсацией, данные возвращаются в состояние, предшествующее началу исполнения операции;
- Без компенсации, данные не откатываются.

7. По уровню обработки особой ситуации:

- На уровне операции, в которой произошел отказ;
- На уровне подпроцесса, в котором находится операция, в которой произошел отказ;
- На уровне родительского процесса.

7.2. СОБЫТИЯ ДЛЯ ОБРАБОТКИ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

Для обработки особых ситуаций применяются следующие виды событий:

- Исключение;
- Ошибка;
- Эскалация;
- Окончание;
- Компенсация;
- Отмена.

Первые два имеют много общего, работают одинаково, по сути они отличаются лишь названием. Если возникло *исключение*, соответствующее действие может быть завершено, но с не-приемлемым для бизнеса результатом. Если произошла *ошибка*, действие не завершается. Например, недоступность внешнего источника информации в одной ситуации может являться ошибкой, а в другой – исключением, означая, что действие надо выполнить вручную. Для обработки ситуации, когда для решения операции следует привлекать руководителя, используется другой обработчик, который называется *эскалация*. Предполагается, что исполнителю для завершения выполнения операции не хватает полномочий или ресурсов и он вынужден обратиться за помощью к руководителю.

Окончание предназначено для обработки таких отказов, возникающих на этапе исполнения, которые не могут быть исправлены, т.ч. дальнейшая обработка не имеет смысла;

Компенсация и *Отмена* используются в транзакционных подпроцессах, где может возникнуть необходимость в откате данных в состояние, предшествующее началу выполнения транзакции.

Обратим внимание, что особые ситуации могут быть безымянными и именованными, последние имеют уникальное идентифицирующее имя. Если произошло именованное событие и сгенерировано оповещение, то принять это оповещение сможет только обрабатывающее событие, имеющее такое же

имя. Т.о. в процессе м.б. одновременно несколько однотипных принимающих событий, причем каждое реагирует на определенное оповещение. При этом если имя принимающего обрабатывающего события не задано явно, то оно будет принимать все оповещения данного вида.

У событий для обработки исключительной ситуации есть область действий, которая ограничена границами процесса. Если возникает необходимость работать с оповещением вне границ процесса, то используют ретрансляцию оповещения вовне процесса.

7.3. МЕСТО ВОЗНИКНОВЕНИЯ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

Рассмотрим, как обрабатываются особые ситуации, возникающие на разных уровнях процесса:

7.3.1. ИСКЛЮЧИТЕЛЬНАЯ СИТУАЦИЯ В ОПЕРАЦИИ

В ходе исполнения операции могут возникнуть системные ошибки. Например, при исполнении автоматической операции оказался не доступен внешний сервис, к которому она обращается, или произошла ошибка авторизации, а при выполнении интерактивной операции, оказалось невозможно подобрать исполнителя, который бы удовлетворял всем выдвигаемым критериям. На возникшую ситуацию следует отреагировать, а для этого необходимо сделать ошибку в операции известной на уровне процесса и принять соответствующие меры. Прикрепленное к границе операции событие ретранслирует внутреннюю ошибку вовне операции или подпроцесса.

Пример (см. Рисунок 7-1А) иллюстрирует операцию с прикрепленным генерирующим событием. Если в ходе исполнения этой операции возникнет внутренняя ошибка, то прикрепленное событие должно узнать о ней и послать оповещение другим операциям процесса. Хотя спецификация BPMN об этом явно не указывает, но в большинстве реализаций BPM в

свойствах прикрепленного события можно указать тип конкретной системной ошибки, по которому оно срабатывает. Генерирующее событие посыпает оповещение, которое м.б. обработано на уровне процесса.

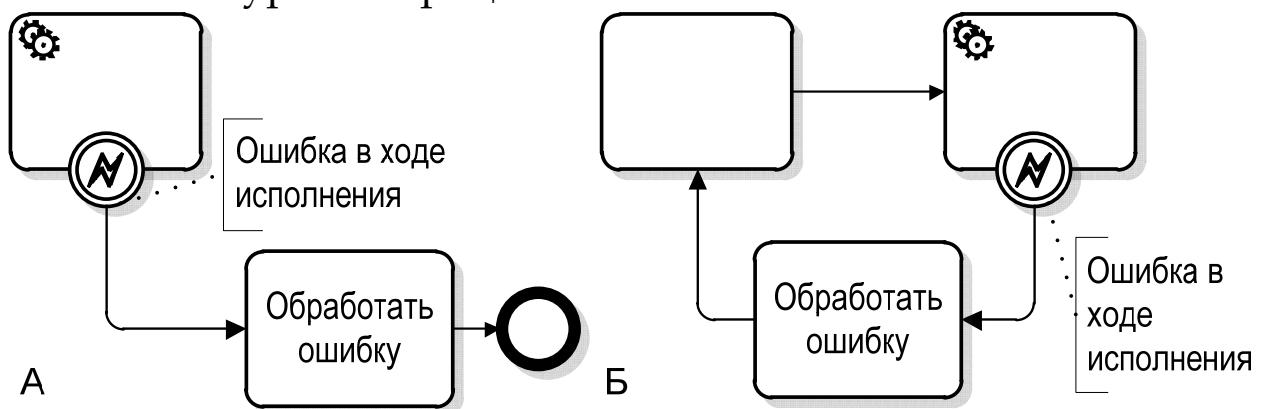


Рисунок 7-1. Обработка события прикрепленного к границе операции

После обработки ошибки можно вернуть управление в нужную точку процесса. Пример (см. Рисунок 7-1Б) показывает, что после обработки управление можно вернуть на предыдущую операцию для повторной обработки.

7.3.2. ИСКЛЮЧИТЕЛЬНАЯ СИТУАЦИЯ В ПРОЦЕССЕ

Бизнес исключение м.б. заложено в логику процесса на стадии проектирования. Предположим, процесс (см. Рисунок 7-2) может завершиться одним из трех возможных результатов: одобрением заказа, отказом, а возможно, что решение не принято, например, по причине отсутствия всей необходимой информации. Первые два варианта мы рассматриваем как нормальное завершение, а третье как бизнес исключение, дальнейшее рассмотрение не целесообразно без дополнительной обработки. Оканчивая одну из ветвей генерирующими событием-ошибкой, мы явно установили статус завершения подпроцесса. Т.о. вызывающий процесс сможет понять, что в подпроцессе произошло бизнес исключение и обработать его соответствующим образом.

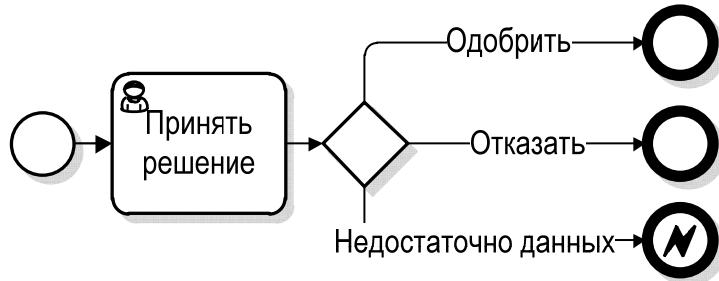


Рисунок 7-2. Простое бизнес исключение

7.3.3. ИСКЛЮЧИТЕЛЬНАЯ СИТУАЦИЯ ВО ВНЕШНЕЙ СРЕДЕ

Рассмотрим, что произойдет, если источник особой ситуации лежит вне текущего процесса, в другом процессе. Мы будем помнить, что все рассмотренные нами до сих пор генерирующие события, рассыпали оповещения, которые видны только в пределах данного пула. Вне этого пула оповещение, сгенерированное событием, не известно. Единственным каналом обмена между процессами является поток сообщений, так что сгенерированное оповещение можно пересыпать между процессами только через поток сообщений.

Рассмотрим пример (см. Рисунок 7-3), Процесс1 порождает внешний Процесс2. Позднее в Процессе1 происходит внутренняя ошибка, которую фиксирует прикрепленное к соответствующей операции обрабатывающее событие, оно генерирует оповещение, но последнее известно только в рамках Процесса1. Что бы сделать это оповещение известным вне данного процесса, обработчик особой ситуации включает операцию отправки сообщения. Поток сообщений достигает Процесса2 и обрабатывается прикрепленным к Процессу2 граничным событием. Последнее создает специальный поток обработки. В зависимости от ситуации, прикрепленное граничное обрабатывающее событие м.б. прерывающим и непрерывающим.

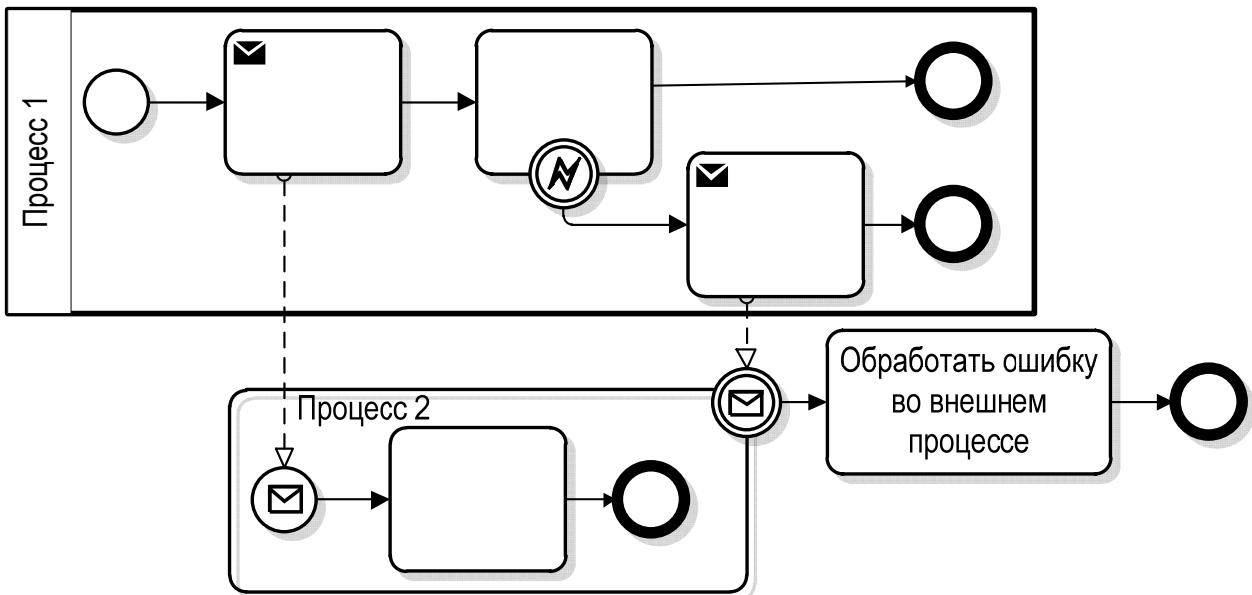


Рисунок 7-3. Исключительная ситуация во внешней среде

7.4. УРОВНИ ОБРАБОТКИ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

Исключительная ситуация м.б. обработана на уровне операции, в которой она возникла, на уровне подпроцесса, в котором находится дефектная операция, на уровне всей системы. Т.о. возникает иерархия, определяющая порядок обработки особых ситуаций. Представим себе, что в некоторой операции зафиксирован отказ. Рассмотрим порядок обработки особых ситуаций.

- В первую очередь будет выполнено обрабатывающее событие, прикрепленное к границе отказавшей операции.
- Если такой обрабатывающей операции нет, то будет выполнен событийный вложенный подпроцесс.
- Если событийного, вложенного подпроцесса нет, будет использовано обрабатывающее событие, прикрепленное к границе процесса, окружающего отказавшую операцию.
- Если таковой нет, то ошибка будет задокументирована в файле журнала ошибок BPMS.

7.4.1. ОБРАБОТКА НА УРОВНЕ ОПЕРАЦИИ

Пример (см. Рисунок 7-1) иллюстрирует обработку ошибки на уровне отказавшей операции.

7.4.2. ОБРАБОТКА ВО ВЛОЖЕННОМ ПОДПРОЦЕССЕ

Если необходимо отреагировать на событие прямо в подпроцессе, то можно воспользоваться т.н. событийным вложенным подпроцессом, показываемом внутри пула, изображаемого пунктирной линией. Этот подпроцесс начинается с одноименного события, так что причинно-следственная связь ясна из контекста (см Рисунок 7-4).

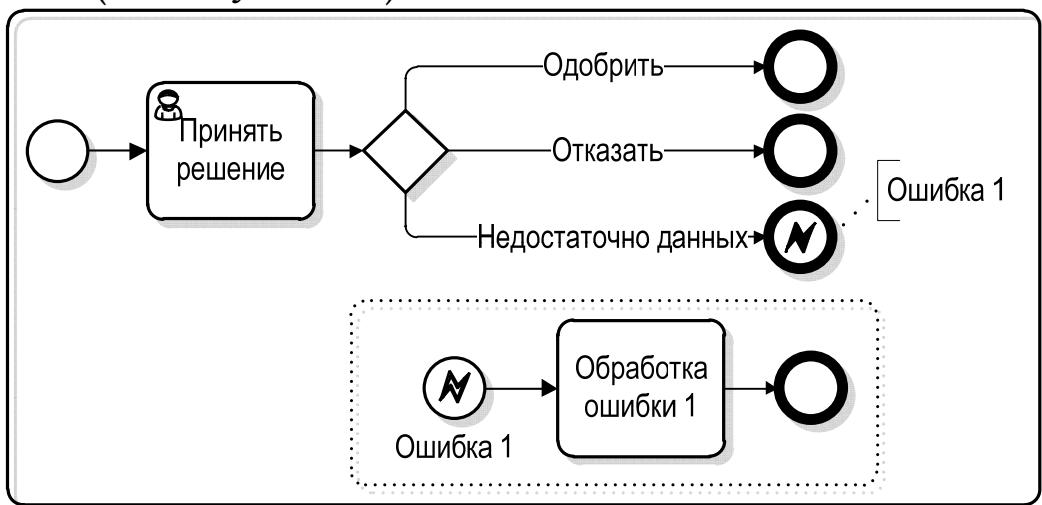


Рисунок 7-4. Обработка бизнес исключения во вложенном подпроцессе

7.4.3. ОБРАБОТКА В ВЫЗЫВАЮЩЕМ ПРОЦЕССЕ

Повторно рассмотрим предыдущий пример, на этот раз предположим, что число вариантов завершения увеличилось, возможна ситуация, когда нет кворума для принятия решения, мы объявим эту ситуацию, как Ошибку 2. Будем обрабатывать эту ошибку в вызывающем процессе. Для этого он должен иметь

прикрепленное обрабатывающее событие (см. Рисунок 7-5)

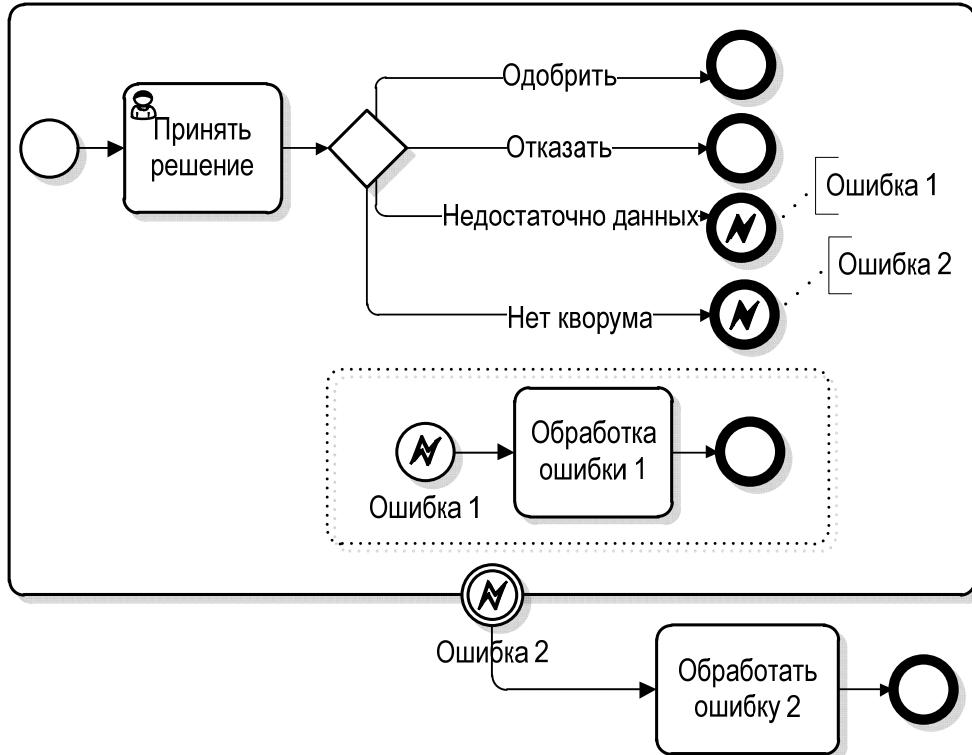


Рисунок 7-5) одноименное тому, что вызвало ошибку. Теперь, в случае нестандартного завершения подпроцесса со статусом Ошибка 2, управление будет передано одноименному прикрепленному событию, последнее создает новый поток управления.

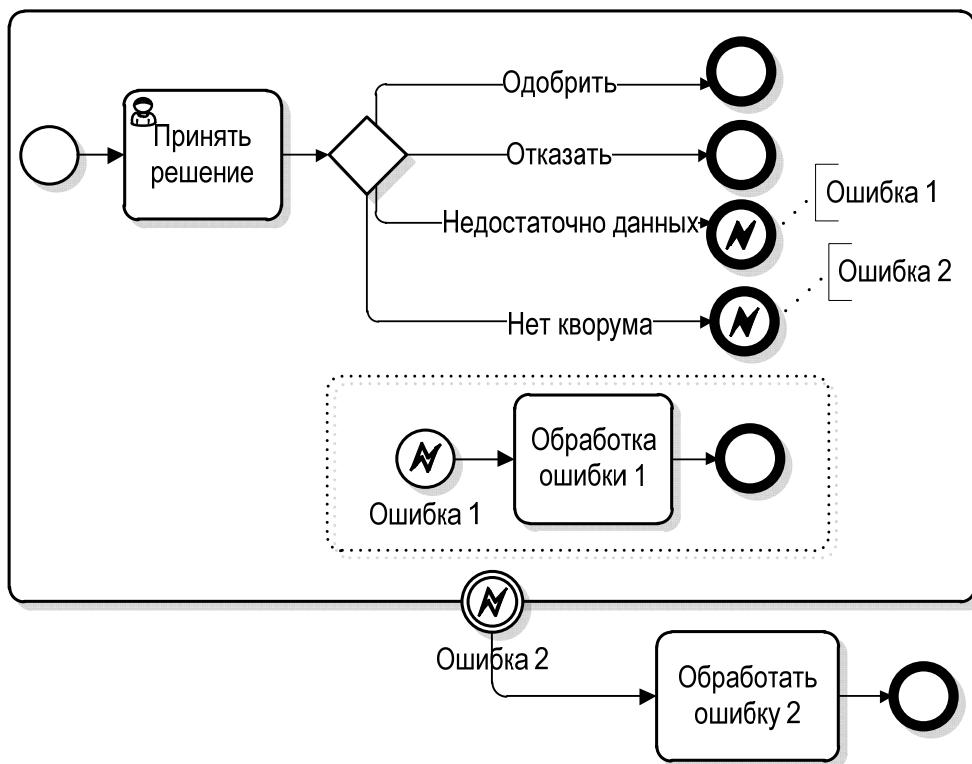


Рисунок 7-5. Обработка события прикрепленного к границе процесса.

7.5. ДЕЙСТВИЯ ПОСЛЕ ОБРАБОТКИ ИСКЛЮЧИТЕЛЬНОЙ СИТУАЦИИ

После завершения обработки исключительной ситуации управление может быть возвращено в точку прерывания или м.б. продолжено без возврата. Рассмотрим примеры.

7.5.1. БЕЗ ВОЗВРАТА В ТОЧКУ ПРЕРЫВАНИЯ;

Пример (см. Рисунок 7-6) показывает обработку ошибки, которая прерывает основной процесс, т.ч. операция «Б» в случае ошибки не исполняется.

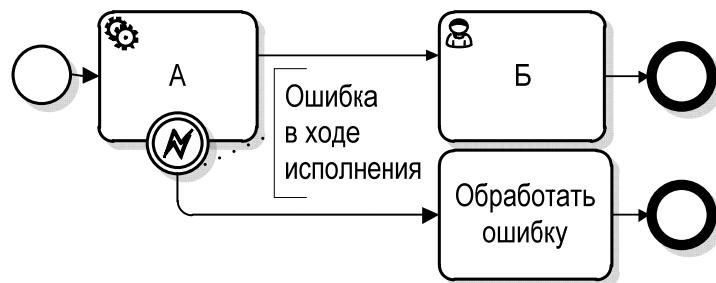


Рисунок 7-6. Обработка ошибки без возврата в основной процесс

7.5.2. С ВОЗВРАТОМ В ТОЧКУ ПРЕРЫВАНИЯ

Пример (см. Рисунок 7-7) иллюстрирует обработку ошибки, с возвратом в основной процесс.

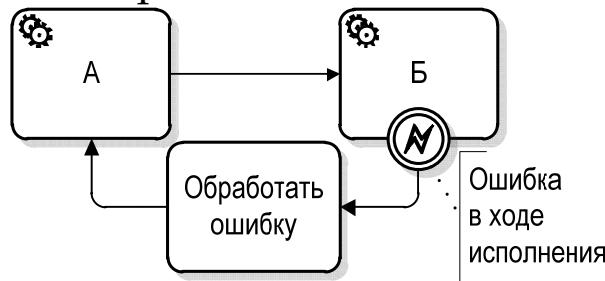


Рисунок 7-7. Обработка ошибки с возвратом в основной процесс

7.6. ВЛИЯНИЕ ИСКЛЮЧЕНИЯ НА ТЕКУЩИЙ ПРОЦЕСС

По влиянию исключения на текущий процесс различают события:

- Прерывающие исполнение процесса, он приостанавливается до окончания процедуры обработки;

- Не прерывающие, исполнение процесса не приостанавливается, возникает дополнительный поток управления;

7.6.1. ИСКЛЮЧЕНИЯ, ПРЕРЫВАЮЩИЕ ИСПОЛНЕНИЕ ПРОЦЕССА

Если событие прерывает обработку процесса, то при этом создается новый поток управления. Завершение нового потока управления означает окончание всего процесса. При этом не всегда легко понять, куда вернется точка управления. Рассмотрим пример (см. Рисунок 7-8). Во время выполнения операции, которая вызывает внешний сервис, произошла исключительная ситуация, например, сервис оказался недоступен. Хотя на схеме это явно не указано, внутренняя ошибка сервисной операции известна внутри подпроцесса. Оповещение об ошибке будет перехвачено обрабатывающим событием «Обработка ошибки внешнего сервиса», которое инициирует запуск событийного подпроцесса. Поскольку событие является прерывающим, то основной поток управления будет приостановлен. После того, как событийный подпроцесс будет завершен, начнет выполняться операция «B». Т.о., в случае возникновения ошибки во внешнем сервисе, операция «A» не будет выполнена.

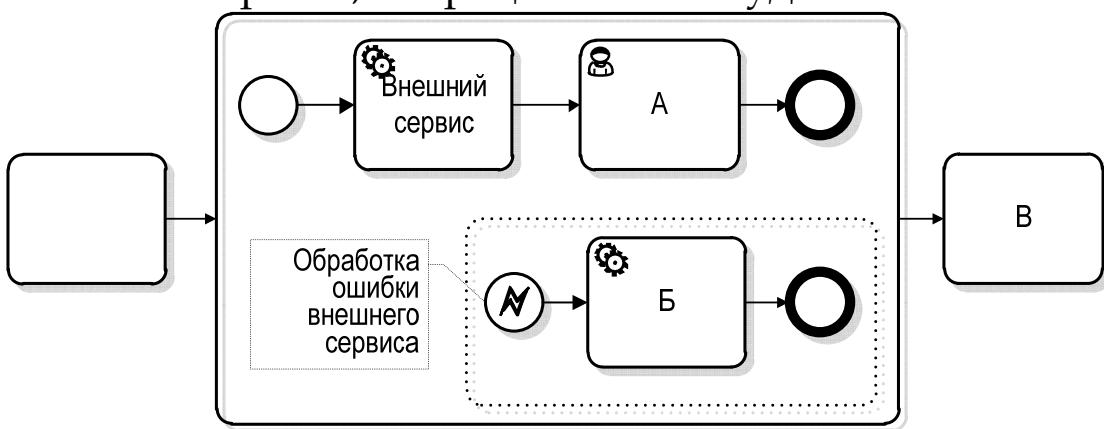


Рисунок 7-8. Исключение, прерывающее исполнение процесса

7.6.2. ИСКЛЮЧЕНИЯ, НЕ ПРЕРЫВАЮЩИЕ ИСПОЛНЕНИЕ ПРОЦЕССА

Если исключение не прерывает исполнение операции, то проблемы с возвратом управления не возникает, однако по-

является необходимость явного завершения дополнительного потока. Рассмотрим пример (см. Рисунок 7-9), на выполнение операции «Выполнить работу» отведено нормативное время. Если лимит исчерпан, то прикрепленное граничное событие-таймер инициирует новый поток управления. Поскольку событие является непрерывающим, основной поток продолжает исполнение. После окончания обработки исключительной ситуации, дополнительный поток следует явно завершить. Следует быть внимательным и стараться исключить ситуации, когда основной и вспомогательный потоки объединяются, это может породить нежелательные последствия.

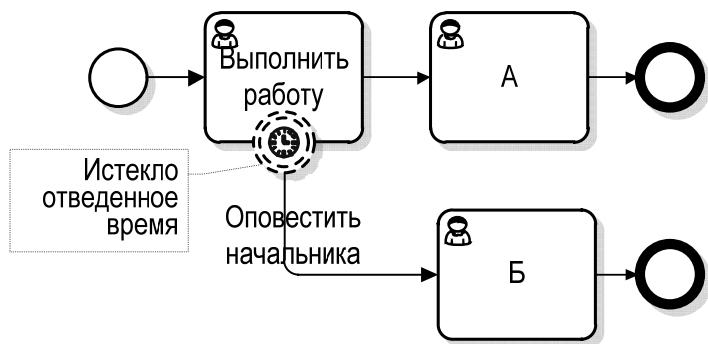


Рисунок 7-9. Исключение, непрерывающее исполнение процесса

7.7. СОБЫТИЕ-ЭСКАЛАЦИЯ

Эскалация – означает вынос проблемы для обсуждения на более высоком уровне принятия решения, при невозможности ее разрешения на текущем. Эскалация применяется в ситуации, когда у исполнителя не хватает собственных полномочий, знаний или ресурсов. Возникшая проблема не связана со временем или отставанием от норматива, для этой цели используется событие-таймер. Решение об эскалации, либо принимает сам исполнитель, либо оно основывается на некотором правиле. Обычно, при эскалации исполнитель прерывает работу и передает ее руководителю, но в отдельных ситуациях он может продолжить исполнение, тогда говорят о совместной работе над задачей.

Событие-эскалация во многом похоже на событие-ошибка, однако имеет ряд серьезных отличий:

- 1) Эскалация, в отличие от ошибки, может не прерывать выполнение операции, к которой она была прикреплена.
- 2) Эскалация действует на том же уровне подпроцесса, на котором была сгенерирована;
- 3) Эскалация, в случае прерывания выполнения операции, прекращает работу только того потока, в котором она была сгенерирована. Ошибка, как известно, прекращает выполнение всех потоков на том же уровне подпроцесса;
- 4) После обработки эскалации, выполнение потока, в котором она произошла, может быть продолжено;

7.7.1. ОБРАБОТКА ЭСКАЛАЦИИ

Для интерактивных операций «характерны» исключительные ситуации, генерируемые оператором. Например, если в ходе исполнения сотрудник поймет, что у него нет достаточных полномочий для принятия решения, он должен поднять уровень принятия решения, для этого предусмотрено событие эскалация. Можно выделить два способа эскалации: во-первых, сотрудник отдает задание руководителю и теряет возможность продолжить исполнения, а во-вторых, он делится заданием, у него остается возможность продолжить работу над заданием. В первом случае применяется прерывающее событие, а во втором непрерывающее.

Рассмотрим пример прерывающей эскалации (см. Рисунок 7-10), показывающий, что исполнитель, выполняющий операцию, может выбрать одну из опций: «принять решение» или «отказать», однако может случиться, что у него недостаточно полномочий для принятия решения. Если он выбирает вариант эскалации, то точка управления приходит в генерирующее прерывающее событие, которое создает оповещение, дальнейшая работа подпроцесса прерывается. Дальнейшая обработка эскалации осуществляется в соответствии с последовательностью, описанной в предыдущем параграфе: сперва ищется вложенный событийный подпроцесс, затем прикрепленное граничное событие, если их нет, то протоколируется ошибка.

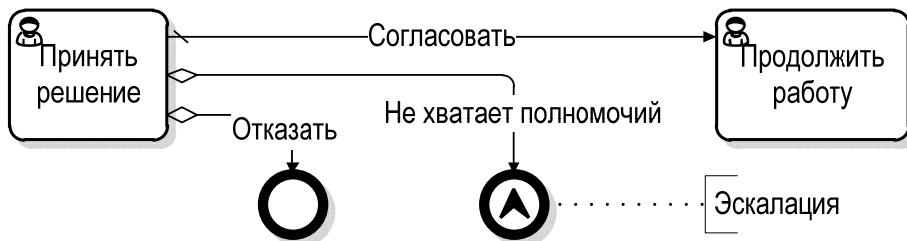


Рисунок 7-10. Обработка прерывающего события-эскалация

7.7.2. ОБРАБОТКА НЕПРЕРЫВАЮЩЕГО СОБЫТИЯ-ЭСКАЛАЦИЯ

Пример (см. Рисунок 7-11) показывает обработку эскалации с использованием непрерывающего события. Если у исполнителя не хватает полномочий, то он выбирает вариант эскалации, но поток управления не прерывается, т.ч. управление передается следующей операции. Параллельно стартует событийный подпроцесс, руководитель должен консультировать исполнителя.

За кажущейся простотой этого сценария скрывается определенная сложность, связанная с организацией правильной работы с данными. Поскольку обработка эскалации осуществляется во вложенном подпроцессе, который имеет доступ к контексту экземпляра процесса, оба, исполнитель и руководитель, имеют доступ к одним данным.

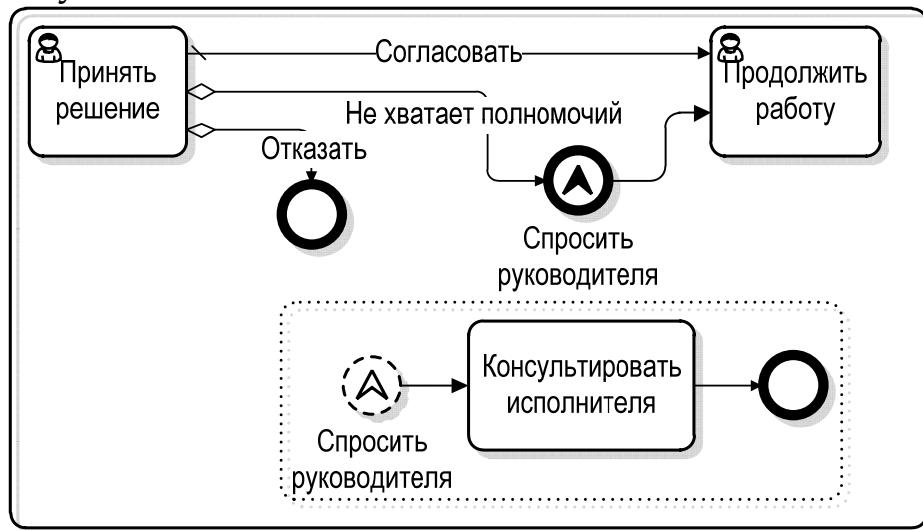


Рисунок 7-11. Обработки непрерывающего события-эскалация

Рассмотрим пример (см. Рисунок 7-12), изображающий вложенный подпроцесс, в котором осуществляется обработка

заявки на предоставление кредита. Поскольку сумма заявки превышает установленный для операциониста лимит ответственности, поток направляется на генерирующее событие эскалация, которое вырабатывает оповещение. Обрабатывающее событие эскалации, которое прикреплено к границе подпроцесса, перехватывает оповещение и ретранслирует его, направляет задание руководителю с необходимым уровнем полномочий. В этом примере использовалась прерывающая эскалация, т.ч. исполнитель прекратил работу над заданием.

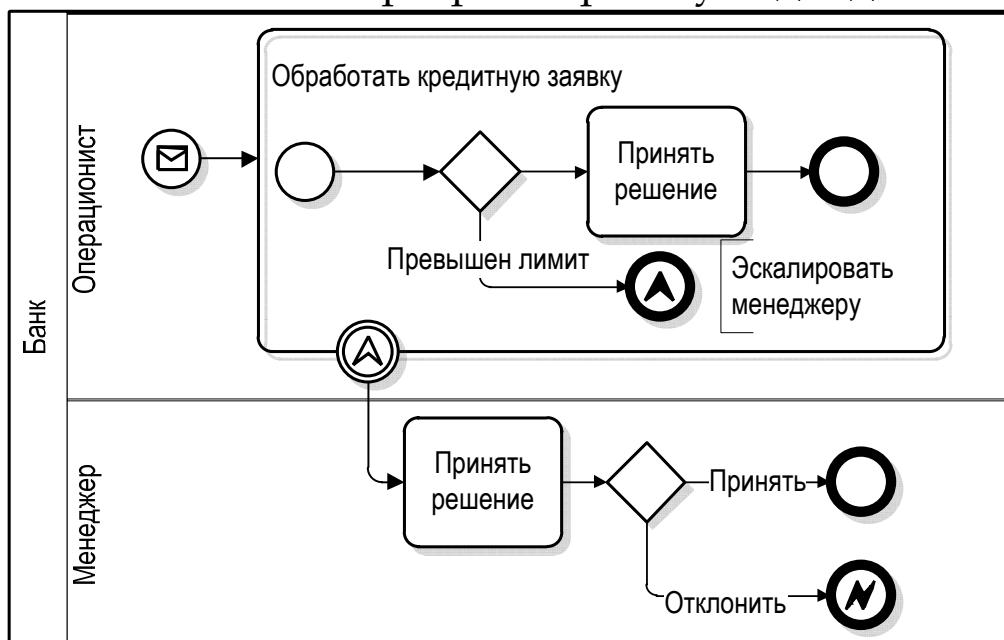


Рисунок 7-12. Обработки непрерывающего события-эскалация

7.8. ОБРАБОТКА КОМПЕНСАЦИИ

Событие компенсация тесно связано с понятием транзакция. Обычно транзакции связаны с выполнением комплексных изменений, производимых во многих информационных системах, для транзакционного взаимодействия которых разработан протокол WS Transaction. Он определяет протоколы:

- Завершения транзакции;
- Ненадежного и надежного двухфазного подтверждения.

Большинство реализаций BPM имеют специальный адаптер, который может анализировать оповещения, пересылаемые по протоколу WS Transaction, т.ч. если последняя не будет завершена, то обрабатывающее событие сможет «услышать» это

оповещение и сгенерировать оповещение внутри процесса.

Рассмотрим первый пример (см. Рисунок 7-13), после регистрации заявки происходит резервирование товара и только потом проверка, на основании чего принимается решение. Если заявка принята, то процесс успешно завершается, но если произошел отказ, то регистрацию следует отменить. Ветка, связанная с отказом заканчивается генерирующим *событием-компенсация*. Это оповещение инициирует откат данных. Все операции, действия которых необходимо отменить и произвести откат данных, помечены обрабатывающим граничным событием компенсацией. Каждая отменяемая операция имеет ассоциативную связь (показывается пунктирной линией) с дополнительной операцией, которая возвращает данным первоначальное значение. После завершения исполнения компенсирующего воздействия возврат управления в компенсируемую операцию не предусмотрен. Операции, требующие компенсации, обрабатываются в обратном порядке.

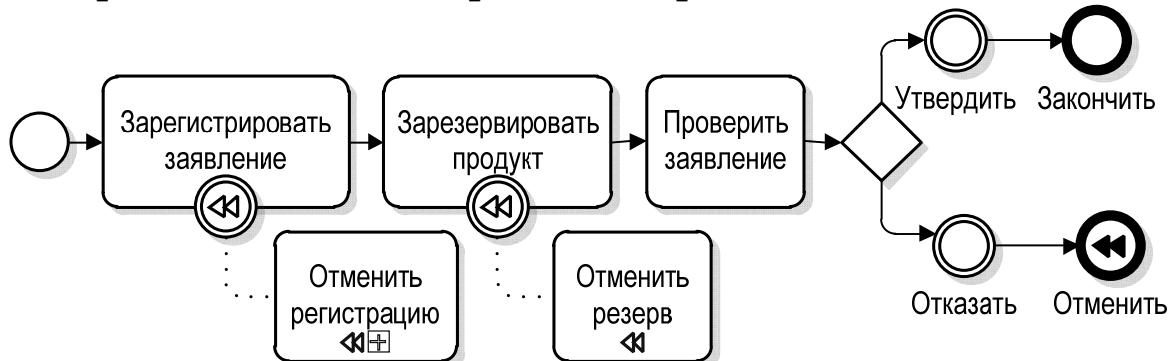


Рисунок 7-13. Обработка события-компенсация

Откат выполняется только для успешно завершенных транзакций процесса. Если какая-то операция не была завершена вовсе, то откат не производится, поскольку неизвестно, были ли реально изменены данные. Например, если операция прервана по причине возникновения исключительной ситуации, то откат невозможен до завершения обработки исключения.

Компенсируемая операция д.б. «видима» для генерирующего события, поэтому последнее должно располагаться:

- В потоке текущего процесса;

- Внутри вложенного событийного подпроцесса, который «принадлежит» текущему процессу;

Рассмотрим пример, иллюстрирующий последний случай (см. Рисунок 7-14), он изображает событийный подпроцесс, запускаемый обрабатывающим *событием-компенсация*. Последнее срабатывает, получив оповещение от генерирующего *события-компенсация*.

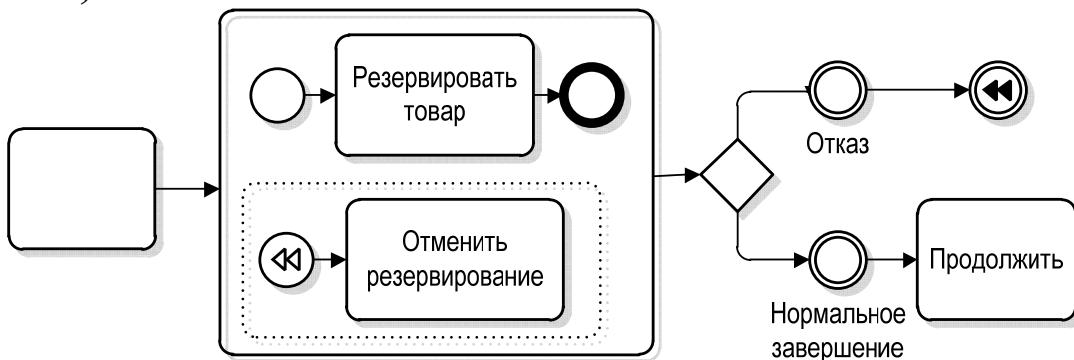


Рисунок 7-14. Откат с использованием событийного подпроцесса

Если текущий подпроцесс содержит операции, помеченные граничным обрабатывающим *событием-компенсация*, то происходит следующее: все успешно завершенные операции последовательно откатываются в состояние предшествующее их началу.

Рассмотрим следующий пример (см. Рисунок 7-15), многопоточный подпроцесс может завершиться со статусом нормальное завершение или отказ. В последнем случае возникает необходимость произвести откат данных, измененных в подпроцессе. Поскольку последний является многопоточным, откат будет выполнен столько раз, сколько запускался подпроцесс.

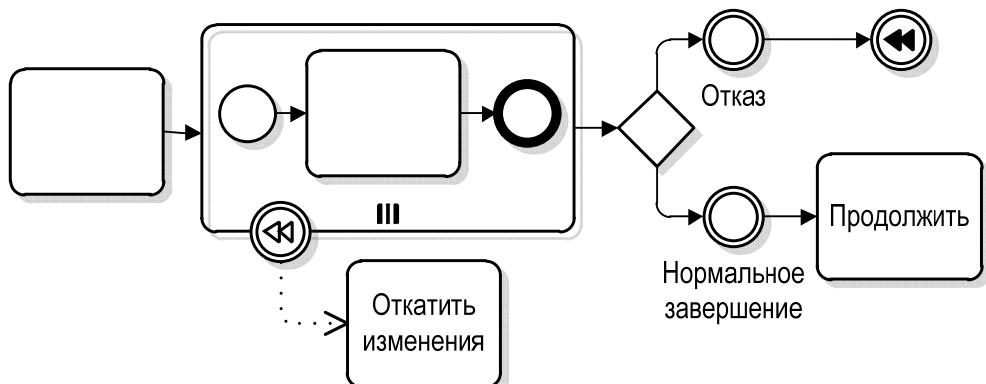


Рисунок 7-15. Откат, инициируемый событием-компенсация

Следующий пример связан с использованием транзакционного подпроцесса (см. п. 3.5.7). Этот подпроцесс имеет два варианта завершения. При нормальном завершении транзакционного процесса, выполнение продолжится в родительском процессе. В случае завершения подпроцесса генерирующим событием-отмена (см. п. 6.11.3) происходит следующее: сперва производится откат всех операций, помеченных как компенсируемые; после этого подпроцесс считается завершенным со статусом отмена. Обрабатывающее событие-отмена, прикрепленное к границе транзакционного подпроцесса, передает управление на операцию, обрабатывающую отмену, затем потоки управления сливаются.

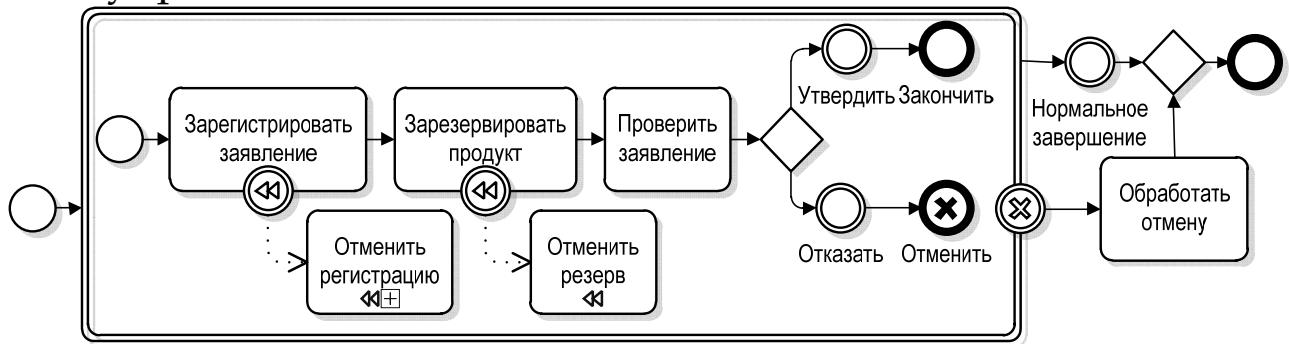


Рисунок 7-16. Пример использования отмены и компенсации

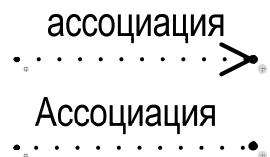
8.ОБЪЕКТЫ ДАННЫХ

Выполнение бизнес-процесса всегда связано с одним или несколькими информационными потоками. Несмотря на то, что нотация BPMN не предусматривает возможность моделирования структур данных, она позволяет связать абстрактные информационные объекты с операциями, размещенными на схеме процесса. Т.о. на схеме процесса можно изображать объекты данных: физических, абстрактных, информационных и т.д. Предполагается, что информационный объект может иметь внутреннюю структуру. По умолчанию она определяется при помощи XML схемы документа. Элементы нотации не хранят внутри себя описание структуры объекта, но содержат ссылку, указывающую на такое описание. На схеме процесса ссылка визуализируется при помощи элемента *ассоциация данных*. В качестве языка запросов к элементам XML-документа используется XPath.

Всю информацию, которая обрабатывается в процессе можно разделить на временную и постоянную. К первому типу относятся те данные, которые существуют только пока выполняется данный процесс. После его завершения эта информация становится не нужной и ликвидируется. Для обозначения временных объектов на схеме процесса используется элемент «Объект данных», «Входные и Выходные данные», «Коллекции объектов данных». Ко второму типу относится информация, которая существует вне рамок процесса, например запись о клиенте в БД CRM. Соответственно для обозначения места хранения таких данных используется элемент «Хранилище данных». Наконец, сообщения так же позволяют отобразить на схеме обмен информационными потоками.

Таблица 8-1 показывает графические элементы, используемые для отображения на схеме процесса потоков данных.

Таблица 8-1 Графические элементы для изображения данных

Вид	Семантика исполнения
 Объект данных	Изображает информационный объект, обрабатываемый в ходе исполнения процесса. Внутренняя структура данных объекта на схеме не отображается.
 Объект данных [Статус обработки]	Изображает статус информационного объекта по ходу обработки
 Коллекция объектов	Изображает информационный объект сложной структуры типа массив. Операции процесса могут обрабатывать элементы массива по отдельности.
 Объект входящий	Изображает информационный объект, необходимый для начала обработки всего процесса или отдельной операции процесса.
 Объект выходящий	Изображает информационный объект на выходе процесса или отдельной операции процесса.
 Хранилище данных	Предметно-ориентированная информационная база данных, предназначенная для сохранения результата выполнения операции процесса.
 Направленная ассоциация  Ассоциация	Показывает направление передачи объекта данных от источника к получателю Связывает объект данных с потоком управления.
 Инициирующее сообщение	Отображает конверт информационной посылки, инициирующей диалог.
 Ответное сообщение	Ответное сообщение, полученное инициатором переписки.

8.1.1. ЖИЗНЕННЫЙ ЦИКЛ И ДОСТУПНОСТЬ ОБЪЕКТОВ ДАННЫХ

Жизненный цикл объекта данных связан с жизненным циклом того процесса, где он используется, он доступен внутри процесса, в котором он используется, включая все вложенные подпроцессы нижнего уровня, но он не доступен вне этого процесса. Рассмотрим пример (см. Рисунок 8-1), он изображает семейство вложенных подпроцессов (см. п. 3.5.1), в каждом определен свой объект данных.

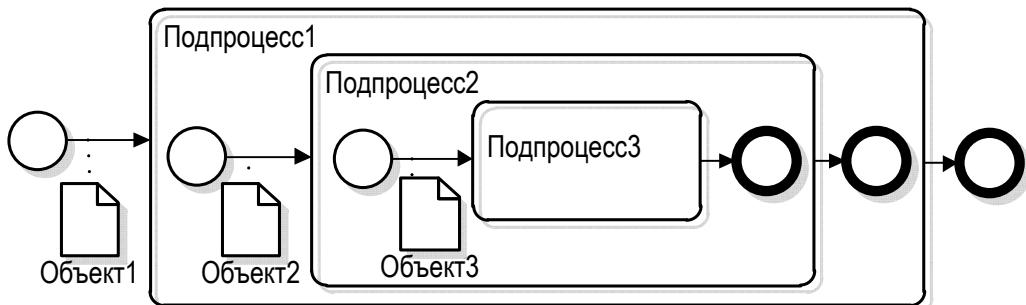


Рисунок 8-1. Вложенные подпроцессы

Следующая схема (см. Рисунок 8-2) иллюстрирует зоны видимости объектов данных, изображенных на предыдущем рисунке. Объект данных 1 будет доступен для всего «процесса 1», включая все вложенные подпроцессы. Объект данных 2 будет доступен для «Процесса 2», и «Процесса3», но он недоступен в «Процессе1» и т.д.

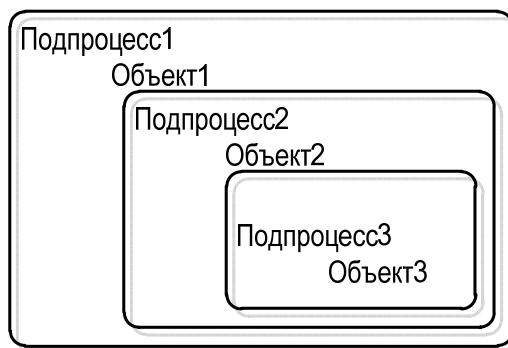


Рисунок 8-2. Зоны видимости объекта данных

Рассмотрим следующий пример, (см. Рисунок 8-3), теперь процесс является повторно используемым (см. п. 3.5.2). Объект данных, определенный в родительском процессе, не известен в вызываемом. При запуске экземпляра подпроцесса будет создана копия соответствующего объекта данных, так что в ходе

исполнения подпроцесса изменения не будут видны во внешнем процессе. После окончания выполнения подпроцесса происходит перенос информации из локальной копии объекта данных подпроцесса в объект данных внешнего процесса. При отмене выполнения данного экземпляра подпроцесса все находящиеся в нем копии *объектов данных* уничтожаются, копирования их во внешний процесс не происходит.

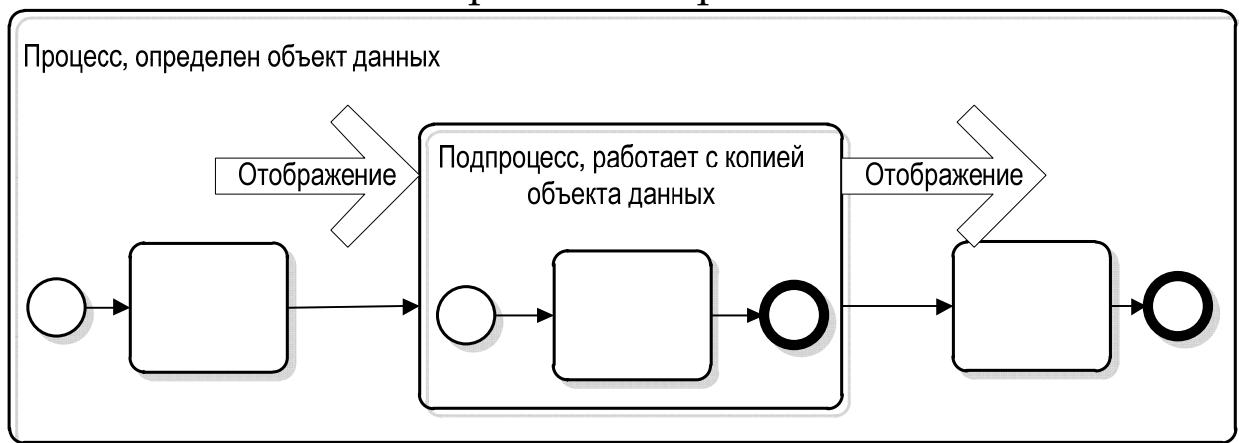


Рисунок 8-3. Передача данных в повторно используемый подпроцесс

8.1.2. СПОСОБЫ ОТОБРАЖЕНИЯ АССОЦИАЦИИ НА СХЕМЕ ПРОЦЕССА

Существует два способа отображения факта передачи объекта данных между операциями процесса. Во-первых, направленная ассоциация данных явно указывает движение информационного объекта от источника к получателю. При этом возникает ощущение, что потоки управления и данных существуют независимо друг от друга (это не вполне верно – если поток управления направлен на логический оператор «ИЛИ», направление маршрутизации определяется значением информационного объекта).

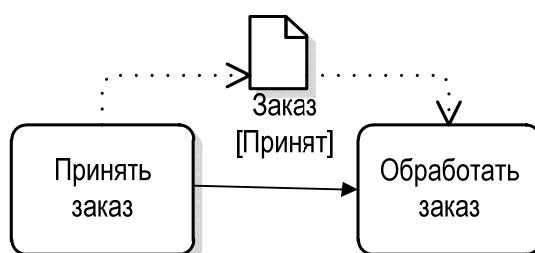


Рисунок 8-4. Направленная ассоциация данных

Во-вторых, стандарт допускает альтернативное обозначение передачи объекта данных, когда ненаправленная ассоциация связывает объект данных с потоком управления. В этом случае, источник и получатель, направление передачи объекта данных определяются стрелкой потока управления. Такая форма подчеркивает, что потоки управления и данных суть одно и то же.

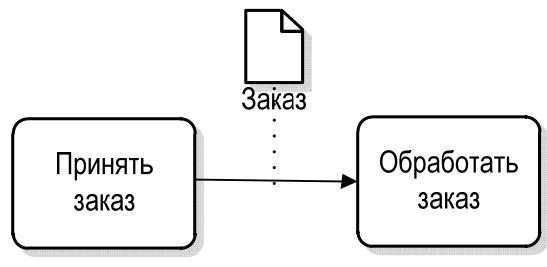


Рисунок 8-5. Ненаправленная ассоциация данных

8.1.3. СТАТУС ОБРАБОТКИ ДОКУМЕНТА

В ходе моделирования процесса может возникнуть желание отобразить **статус обработки объекта данных**, например заказа клиента. После выполнения операции «Принять заказ» статус заявки имеет значение «Принята», после выполнения операции «Обработать заказ», ее статус изменится на «Обработана», а операция «Выдать заказ» устанавливает статус «Выдан».

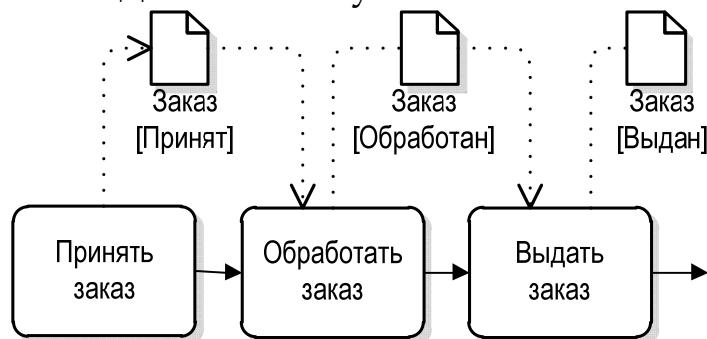


Рисунок 8-6. Статус обработки информационного объекта

8.1.4. КОЛЛЕКЦИЯ ОБЪЕКТОВ ДАННЫХ

Коллекция объектов данных рассматривается как набор однотипных элементов, образующих массив. Часто возникает задача обработать по отдельности все элементы массива. Задачу можно решить, если воспользоваться операцией или подпроцессом параллельного (циклического) исполнения. Если по-

дать коллекцию объектов данных на вход операции параллельного исполнения, то можно так настроить отображение массива на входные аргументы операции, что последняя будет выполнена для каждого из элементов этого массива. Используя коллекция объектов данных можно автоматически породить столько экземпляров процесса, сколько объектов данных хранится в массиве.

Рассмотрим пример, компания подготовила приглашение на конференцию, используя данные из CRM сформировала список, затем рассыпает приглашение по почте. Подпроцесс «отправить приглашение» должен быть выполнен столько раз, сколько приглашенных указано в списке. Список приглашенных оформлен как коллекция документов. Необходимо дополнительно настроить отображение входных данных операции.

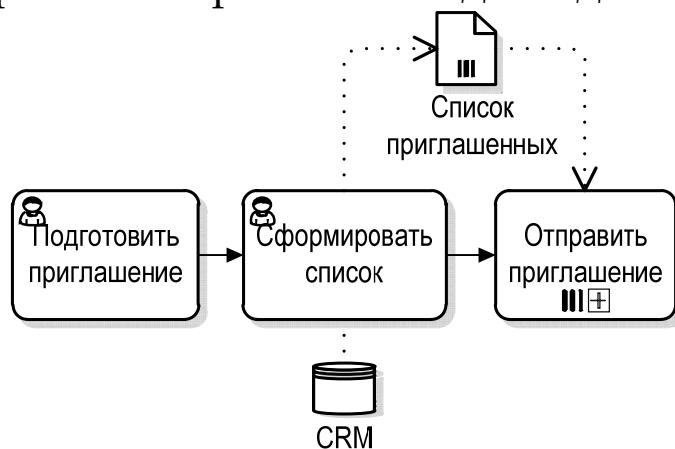


Рисунок 8-7. Коллекция объектов данных

8.1.5. ХРАНИЛИЩЕ ДАННЫХ

В ходе моделирования процесса возникает необходимость отобразить сохранение информации во внешнем хранилище данных или чтение информации из некоторого внешнего источника, например справочника. Для изображения места хранения контекстных данных процесса используется графический элемент **хранилище данных**. Учитывая, что под данными в нотации BPMN понимается не только информация в цифровом виде, то под хранилищем может подразумеваться склады, помещения, СУБД и т.д. Стрелка ассоциации указывает направление передачи информации: во внешний источник – запись, из внешнего источника – чтение. Пример (см. Рисунок 8-8)

илюстрирует прием заказа, при этом, надо записать данные о текущем заказе в CRM, затем извлечь из CRM информацию о предыдущих заказах, что бы рассчитать скидку клиента.

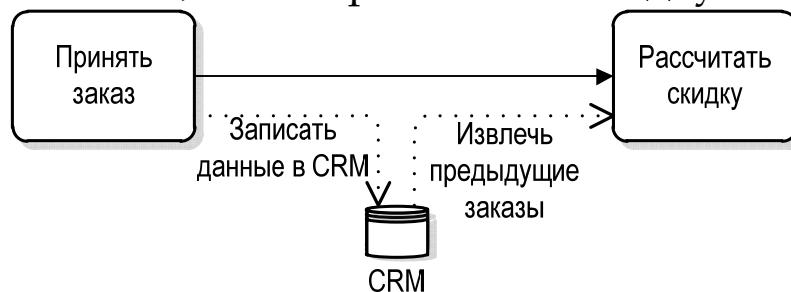


Рисунок 8-8. Хранилище данных

8.1.6. ВНЕШНИЙ ВХОД И ВЫХОД ПРОЦЕССА

В ходе моделирования часто возникает необходимость специфицировать **входные и выходные данные процесса**, для этого можно использовать объекты Внешний вход и Внешний выход процесса. Пример, (см. Рисунок 8-9А) изображает входной документ. Следует быть внимательными, входной документ должен поступить на вход процесса еще до начала его исполнения. Второй пример (Рисунок Б) изображает выходной документ.

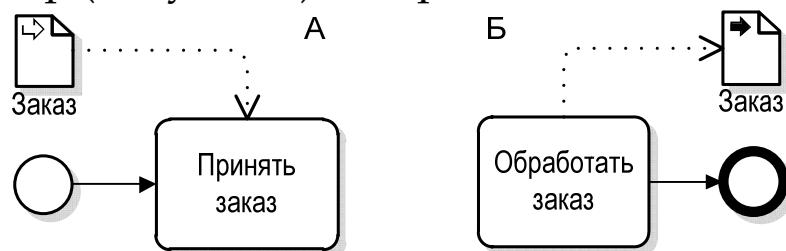


Рисунок 8-9. Входные и выходные данные процесса

Если аналитик не уверен, что документ окажется в наличии до начала работы процесса, надежнее использовать вместо Внешнего входа чтение информации из *внешнего хранилища*, (см. Рисунок 8-10).

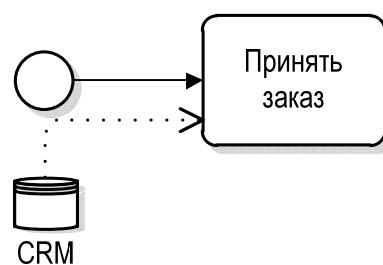


Рисунок 8-10. Чтение информации из внешнего хранилища

8.1.7. СЕМАНТИКА ИСПОЛНЕНИЯ АССОЦИАЦИИ ДАННЫХ

Для того, что бы ассоциация могла быть исполнена, все источники данных, определенные в ассоциации, должны быть в наличие и находиться в активном состоянии. Ассоциация данных не может быть «выполнена», если хотя бы один из источников не доступен. Это означает, что источник должен получить управление либо в результате прибытия маркера с потоком управления, либо в результате получения сигнала, который активирует соответствующую операцию.

Ассоциация может опционально содержать правило трансформации объекта данных. Трансформация не имеет визуального представления на схеме процесса и может отображаться только во внутренней мета модели процесса. Если трансформация не была задана или отсутствуют ссылки на нее, то может быть указан один единственный источник, а содержимое этого источника будет просто копироваться в операцию получатель. Если трансформация определена, то будет выполнено описываемое преобразование, а результат будет передан в операцию получатель. В данном случае может быть указано любое количество источников (от нуля и более), однако, эти источники не обязательно используются в выражении.

Ассоциация данных может использоваться на разных участках диаграммы процесса, причем объекты, источники и получатели данных могут изменяться.

8.1.8. ПОЛЕЗНАЯ ИНФОРМАЦИОННАЯ НАГРУЗКА СИГНАЛОВ И ОПОВЕЩЕНИЙ

Сигналы и оповещения (эскалация, ошибки, сигналы) способны нести полезную информационную нагрузку. Для связывания объектов данных процесса и полезной информационной нагрузки используется ассоциация данных. Для генерирующих событий входной набор данных отображается на аргументы оповещения. Для обрабатывающего события все наоборот, аргументы оповещения отображаются на выходной набор данных. Если полезная нагрузка не предусмотрена, то генери-

рующее событие отправляет пустое сообщение, а обрабатывающее не возвращает полезных данных.

8.1.9. ПОТОКИ ДАННЫХ И УПРАВЛЕНИЯ

У многих аналитиков возникает ощущение, что потоки данных и потоки управления не связаны между собой, что не вполне верно.

Процесс описывает последовательность изменения состояния системы во времени. В производственном процессе фиксируются изменения состояния материального продукта. Но в бизнес-процессе изменение состояния системы фиксируется в информационных объектах. Несмотря на невидимость битов и байтов они реально существуют в окружающем материальном мире. При их обработке изменяются не физические свойства носителя, а информация, содержащаяся на нем. Принято считать, что бизнес-процессы имеют дело с обработкой именно информации. К этому классу можно условно отнести процессы документооборота. Хотя документ (лист бумаги) есть вполне материальный объект, решения в ходе работы принимаются на основании информации, а не физических свойств документа, например цвета или плотности бумаги

Спецификация BPMN 2.0. использует термин поток операций, но не определяет его явно. Для облегчения восприятия вводится понятие токен, который определяется как «теоретическое понятие», используемое для определения поведения исполняемого процесса. Токен движется вдоль процесса и, тем самым определяет статус обработки. Попытаемся понять, что есть токен.

Когда диаграмма BPMN подготавливается для исполнения в BPMS, необходимо определить т.н. переменную процесса. Под переменной процесса понимается информационный объект, который используется для записи результата выполнения очередного действия, что бы передать его на вход следующего. Переменная процесса движется вдоль диаграммы, ее прибытие в узел активирует выполнение очередной работы. Переменная

процесса есть суть объект управления. Т.о. токен можно ассоциировать с переменной процесса.

Когда аналитик строит аналитическую модель процесса, то в качестве объекта управления он, обычно, выбирает какой либо документ, важный с точки зрения выполняемого задания. Например, в процессе «согласование кредитной заявки» объектом управления являются документы «Заявка». Именно в ней участники ставят свои отметки о ходе рассмотрения дела.

Если строится модель процесса в BPMN, в качестве переменной процесса выбирается объект данных. По мере продвижения объекта управления, в нем происходят качественные изменения, отражающие ход выполнения операций процесса. Например, выполнение операций «Рассмотреть заявку» приводит к качественному изменению состояния «Заявки». Т.о. можно говорить, что движение объекта данных «Заявка» образует поток управления, который можно ассоциировать с токеном процесса.

8.2. ЗАМЕЧАНИЕ О РАБОТЕ С ДАННЫМИ:

Нотация BPMN определяет семантику исполнения ЛО «И» только в части потоков управления, при этом она умалчивает о работе с данными. При создании исполняемых процессов работу с данными необходимо учесть. Давайте рассмотрим, что происходит с данными.

Логический оператор «И» разветвляющий входной ПУ на несколько параллельных потоков управления не создает для каждого из них отдельной копии данных. Т.о. каждая ветвь работает с общим для всех объектом данных. Возможны два сценария:

- Каждая ветвь работает со своим элементом общего объекта данных
- Все ветви работают с одним и тем же элементом общего объекта данных.

Рассмотрим первый пример, экономист и юрист производят согласование договора. Согласование экономиста и юриста есть две разные функции, оба работают со своей частью договора, зоны их ответственности не пересекаются. Поэтому каждый из них помещает свои комментарии в отдельное, специально предназначеннное для этого поле. Во втором примере функции, выполняемые в параллельных ветвях, совпадают. Если два работника выполняют одну и ту же функцию, то ни имеют доступ к одному и тому же объекту данных. Возникает коллизия, изменения, выполненные первым, могут быть отменены вторым, причем первый об этом не узнает.

9. ЗОНЫ ОТВЕТСТВЕННОСТИ (ДОРОЖКИ И ПУЛЫ)

В практике реинжиниринга бизнес-процессов принято создавать процессно-ролевую модель, которая закрепляет за каждой операцией соответствующих исполнителей. Концепцию группировки сотрудников по ролям выдвинули Г.Рэмлер и А.Брейч. Они предложили изображать на диаграмме процесса дорожки, группирующие пользователей по функциям или что то же самое по работам, которые они должны выполнить. Дорожкам дается имя, которое позволяет однозначно идентифицировать соответствующую категорию пользователей.

9.1. РОЛЕВАЯ МОДЕЛЬ

Роль – есть группа сотрудников, которые имеют право выполнить данную операцию. При аналитическом моделировании в качестве объекта рассматривается операция процесса, т.о. роль объединяет тех исполнителей, которые имеют обязательство выполнить данную операцию. Поскольку права доступа не назначаются пользователю индивидуально, а приобретаются им через определенную ему *роль*, управление доступом сводится к назначению пользователю необходимых *ролей*. Это упрощает администрирование системы в случае добавления нового пользователя или смены им подразделения. Чтобы множества операций, связанных с различными ролями, не пересекались, вводится иерархическая связь между ролями. К примеру, роль «секретарь» может включать в себя роль «регистратор» и, плюс к тому, еще несколько дополнительных операций. Каждый пользователь системы может участвовать в нескольких ролях, а роль может объединять многих участников. Выполнение пользователем соответствующей операции процесса разрешено, если ему присвоена соответствующая роль. Она обладает следующими свойствами:

1. В одну роль могут входить несколько сотрудников;
2. Отдельный сотрудник может иметь много ролей;

3. Роль не связана с должностью, например, руководитель и его подчиненный могут иметь одинаковое право выполнить операцию;

Трактовка понятия *роль* в программировании и бизнес-информатике в немного отличаются. Программирование определяет роль как категорию сотрудников, которые обладают сходными правами доступа к объектам системы. В бизнес моделировании в качестве объектов системы рассматривают только операции процесса. В результате, роль определяет права доступа к операциям, но не определяет права доступа к данным процесса. Т.о. у объектов данных на схеме процесса нет определенных владельцев. Как следствие, пользователю невозможно делегировать права на какой-то определенный информационный объект. Например, пользователь имеет право выполнить какую-то операцию, при этом нельзя определить раздельные права доступа к отдельным элементам данных или документам. Если пользователь имеет право выполнить несколько операций процесса, нельзя указать раздельные права для разных операций. Либо у него есть доступ ко всем подобным объектам системы, либо нет вообще.

Роль не учитывает права доступа к экземплярам процесса, например, два сотрудника в одной роли, но работающие в разных территориальных подразделениях не должны видеть процессы друг друга. Ролевая модель не в состоянии различить этих участников. Так же роль не учитывает полномочия сотрудника, связанные с должностями, например, сотрудник и его руководитель.

Ролевая модель не указывает реального исполнителя, а только некоторое множество кандидатов, поэтому распределение работ между сотрудниками осуществляется в два этапа. Сначала, происходит отбор потенциальных исполнителей, позволяющий среди всех сотрудников организации найти тех, кто удовлетворяют установленным критериям: опыта, навыков, знаний, квалификации должности, и др. Затем из числа кандидатов назначается реальный исполнитель, кому будет пору-

чена работа. В описанной процедуре роль участвует в отборе, но не участвует в назначении.

Первоначально предполагалось использовать ролевые модели в проектах по реорганизации организационной структуры предприятия и перераспределения обязанностей сотрудников. В этом случае вопрос назначения реального исполнителя для операции процесса не являлся предметом рассмотрения. Кроме того, в проектах по реорганизации организационной структуры роль часто подменяется должностью. Но в системах управления бизнес-процессами необходимо указать критерии отбора и назначения реального исполнителя, поэтому с применением ролевой модели возникают трудности.

9.1.1. КАК ПРОБЛЕМА РЕШАЕТСЯ В BPM

В первой версии нотации BPMN термины *роль* и *дорожка* были, по сути, синонимами, так что *дорожка* определяла исполнителей, которым поручено выполнение задания. Во второй версии стандарта ситуация изменилась. Понятие *роль* и *дорожка* по-прежнему используются, однако они не являются однозначными синонимами. В некоторых ситуациях дорожка определяет исполнителя, а в других случаях – нет.

Будем различать две категории исполнителей: физические и юридические лица. В случаях, когда моделируется *оркестровка приватных (внутренних)* бизнес-процессов, под исполнителем мы будем иметь в виду физическое лицо, участника бизнес-процесса. Если моделируются схемы *взаимодействия*, *диалоги*, или *хореография*, участниками считаются юридические лица. Наконец, на схемах *публичных (внешних)* процессов участниками являются как компании, так и физические лица – сотрудники этих компаний.

9.2. ПУЛ И ДОРОЖКА

Для отбора исполнителей используются графические элементы нотации Пул (Pool), Дорожка (Lane).

9.2.1. ПУЛ

Пул это графический элемент, объединяющий одну или несколько дорожек, в первом случае *пул* и *дорожка*, фактически, совпадают. *Пул* ограничивает процесс и, т.о. выступает в роли контейнера для потока операций процесса, который может пересекать границы дорожек внутри пула, но не может выходить за границы пула. Нотация допускает изображать пул как «черный ящик», процесс внутри подразумевается, но не изображается.

Пулы часто применяются при межорганизационном взаимодействии, позволяя графически отобразить организации участников взаимодействия, они ассоциируются с внутренней сущностью метамодели Участник (Participants), который, как правило, отвечает за выполнение процесса, заключенного в соответствующем Пуле. Внутренняя переменная Участник (Participants) может быть определена двумя способами: путем указания имени КомпанииУчастника (PartnerEntity) или РолиУчастника (PartnerRole). Первая есть обозначение конкретной организации или ее организационной единицы, а вторая – есть обобщенное название роли действующего лица. Например, представим, что модель описывает межорганизационное взаимодействие, в котором участвуют две компании «Рога и копыта» и «Одесская бубличная артель «Московские баранки»». Иначе их можно было бы назвать «Заказчик» и «Поставщик». РольУчастника позволяет создать модель процесса, которая на привязана к конкретных участникам. У одного заказчика м.б. несколько поставщиков, причем взаимоотношения с каждым из них описывается одной моделью процесса. В последнем случае РольУчастника может иметь дополнительный признак мультипликативности, указывающий, что в процессе участвуют несколько Поставщиков. Связь между Пулами отображается посредством потоков сообщений.

9.2.2. ДОРОЖКА

Дорожка есть графический элемент, используемый для группировки и разделения операций процесса. Группировка

означает размещение нескольких операций процесса в пределах одной дорожки. Критерий группировки в спецификации BPMN не определен, так что аналитик свободен группировать операции процесса по своему усмотрению. Например, дорожка может объединять операции, выполняемые одним пользователем или сотрудниками одного подразделения. Аналитики, часто ассоциируют дорожки с названием структурного подразделения, должностью или конкретным исполнителем. В первом случае дорожка может именоваться «отдел продаж», «бухгалтерия», во втором - «старший экономист» или «главный бухгалтер», в третьем «Иванов Иван Иванович». Во всех примерах дорожка оказывается жестко привязана к организационно-штатной структуре конкретного предприятия или к его сотрудникам, так что смена структуры приведет к изменению модели процесса.

Дорожки могут быть вложенными, например, может существовать множество дорожек отделов предприятия, а также внутреннее множество дорожек для должностей, существующих внутри каждого отдела. Пример (см. Рисунок 10-1) показывает процесс со вложенными, иерархически организованными дорожками. Принципы группировки на разных дорожках могут различаться.

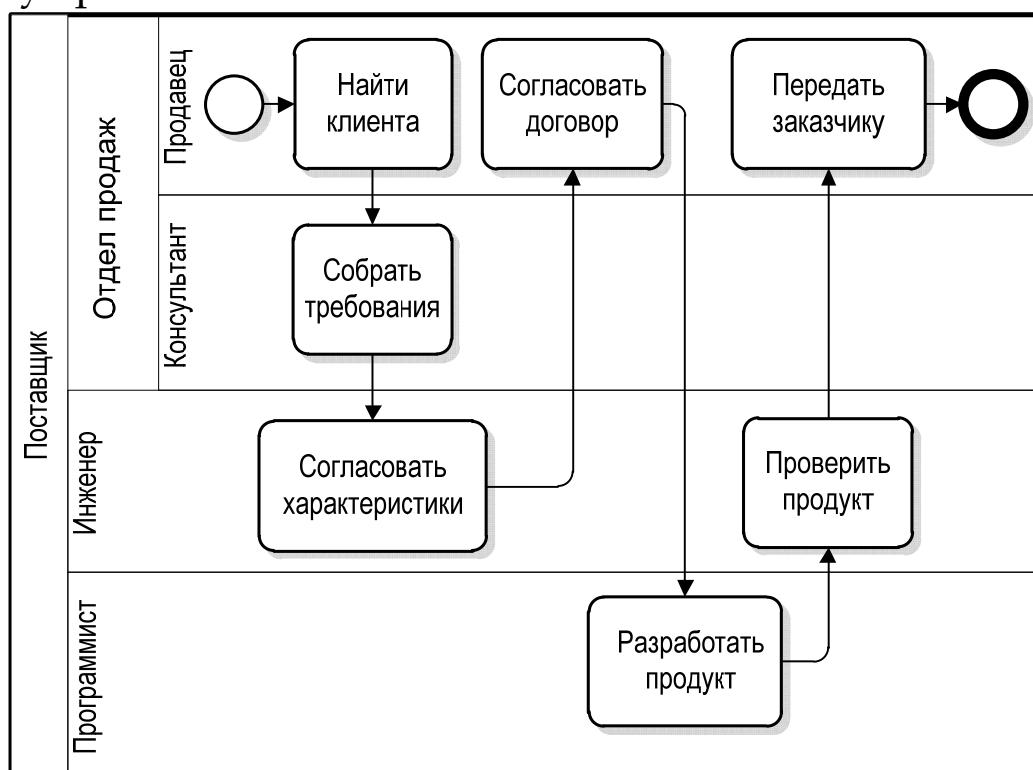


Рисунок 9-1, Иерархия дорожек процесса

С каждой Дорожка связан атрибут РазделительныйЭлемент (partitionElement), посредством которого указываются элементы процесса, которые необходимо разместить внутри данной дорожки. Все дорожки внутреннего множества указывают на разделительный элемент одного типа, например, ссылаются на атрибут ресурс, однако, каждая дорожка ссылается на отдельный экземпляр этого атрибута.

Графический элемент операция изображает работу, выполняемую отдельным пользователем. Нотация BPMN различает семь типов операций (см. п. 4.1), причем, только два типа используются для описания взаимодействия с человеком участником: пользовательская операция описывает интерактивное взаимодействие пользователя с системой, а ручная - работу, выполняемую вне системы, так что участник только подтверждает факт ее исполнения. Рассмотрим отбор и назначения сотрудников для операций, исполняемых человеком.

9.3. ОТБОР ИСПОЛНИТЕЛЕЙ

Спецификация BPMN 2.0 определяет два подхода к отбору лица, выполняющего действие или являющегося ответственным за его выполнение: либо явно указав его имя, либо используя параметрический запрос к внешнему каталогу пользователей. Графическому элементу операция соответствует два внутренних атрибута Исполнитель (Performer) и Ресурс (Resource). Первый позволяет явно указать на физическое лицо, группу лиц, роль или должность в организации, например, Иван Иванов или «Операционист», «сотрудник отдела продаж». Второй указывает на исполнителей косвенно, помогая определить некоторую совокупность лиц, которые могут стать исполнителем соответствующего действия, например «Продавцами» могут выступать сотрудники в разных должностях, возможно работающие в разных подразделениях, для этого используется параметрический запрос к внешнему каталогу пользователей. Нотация явно не определяет тип и формат запроса, поэтому он может включать перечисление всех необхо-

димых критерииев отбора, например: сотрудник клиентского отдела, работающий с физическими лицами по направлению потребительское кредитование в определенном территориальном отделении, имеющий лимит ответственности более 1000 рублей и знающий английский язык. Результатом запроса является список потенциальных исполнителей, из которых в дальнейшем одного надо будет назначить исполнителем. Спецификация BPMN явно не определяет правила и критерии отбора исполнителя, оставляя это на решение аналитика.

9.4. СПЕЦИФИКАЦИЯ WS-HUMAN TASK

Что бы воспользоваться отбором через атрибут ресурс придется использовать внешние, по отношению к спецификации BPMN 2.0 средства, например спецификацию, WS-Human Task. В этом случае, атрибут Исполнитель (Performer) получит соответствующее значение из внешней, подключаемой сущности ИндивидуальныйИсполнитель (HumanPerformer). В этом месте происходит стыковка с внешней моделью WS-Human Task. Атрибут РольРесурса (ResourceRole) сохранен для обеспечения совместимости с предыдущей версией спецификации BPMN 1.0, его использование в текущей версии не специфицируется.

Пример (см. Рисунок 10 1) показывает пул, объединяющий две дорожки, каждая из которых содержит по одной операции. Рассмотрим влияние графических элементов пул, дорожка, операция на выбор исполнителя. Из анализа спецификации BPMN 2.0 становится ясно, что графические элементы пул и дорожка не определяют реального исполнителя операции. Связанные с ними атрибуты КомпанияУчастник (PartnerEntity) и Роль-Участника (PartnerRole) не оказывают влияния на отбор и назначение исполнителя задания, размещенного внутри дорожки. Напротив, атрибуты операции: Исполнитель (Performer) и Ресурс (Resource) непосредственно влияют на отбор участников. Спецификация BPMN не разделяет отбор и назначение исполнителей. Если в результате запроса будут

отобрана группа исполнителей, все они получат предложение выполнить данное задание, так что каждый из них может добровольно назначить себя самого исполнителем.

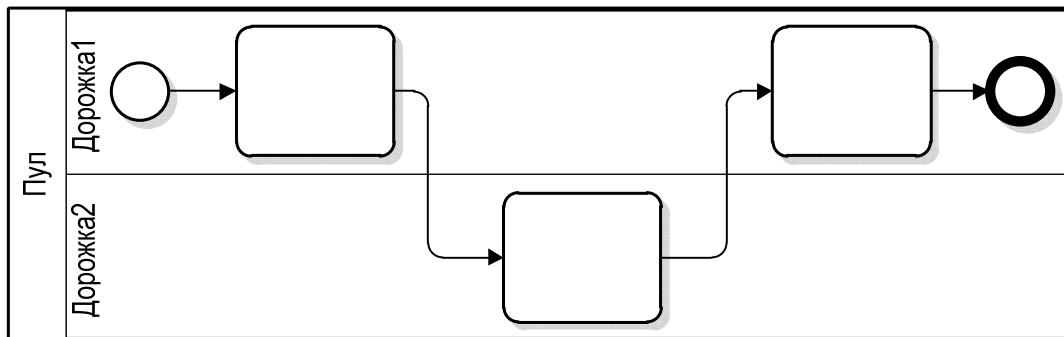


Рисунок 10 1. Пул и Дорожка

Итак, можно сделать вывод, что нотация BPMN включает графические элементы пул и дорожка, однако, по сути, не использует их для выбора исполнителя интерактивной операции. Из-за расплывчатости формулировок и вследствие отсутствия точного разъяснения что обозначают сущности Участник, Роль, Исполнитель и Ресурс, отсутствия четкого описания их взаимосвязи, оказывается трудно различить, как происходит отбор и назначение, сложно понять их реальное назначение.

Попытаемся объяснить их предназначение. Первые относятся к межпроцессному взаимодействию, вторые к внутренним процессам организации. Разница в том, что модели межпроцессного взаимодействия не являются исполнимыми, поэтому в них сохранена ролевая модель, что действительно удобно. Внутренние процессы являются исполнимыми, ролевая модель в них отсутствует, поэтому спецификация BPMN предлагает использовать либо прямой выбор конкретного исполнителя, либо параметрический запрос, возвращающий список Потенциальных исполнителей.

Рассмотрим, как происходит назначение исполнителей с использованием спецификации WS-Human Task.

9.5. НАЗНАЧЕНИЕ ИСПОЛНИТЕЛЕЙ В WS-HUMAN TASK

Спецификация WS-Human Task оперирует концепцией *универсальная пользовательская роль* (generic human role), которая включает следующие понятия:

- *Логическая группа* участников, под которой понимается совокупность пользователей, объединенных некоторым общим критерием отбора. Логическая группа формируется динамически путем запроса к внешнему каталогу пользователей. Формат запроса и его параметры не специфицированы.
- *Инициатор*, лицо породившее Задание (экземпляр процесса). В спецификации не указан состав действий, которые инициаторы могут для этого совершить.
- *Ответственный* (stakeholders), несущий полную ответственность за конечный результат выполнения, он может контролировать исполнение операции, выполнять административные функции, в т.ч. назначать и переназначать исполнителей;
- *Потенциальные исполнители* (potential owners) это группа участников, которым м.б. предложено исполнение задания. Она формируется путем отбора из числа всех исполнителей только тех, кто удовлетворяет соответствующим критериям выборки.
- *Исключенный из числа исполнителей* (excluded owners), пользователь или группа пользователей, которые не соответствуют критериям отбора, используется, что бы определить потенциальных исполнителей методом исключения.
- *Актуальный исполнитель* (actual owner), это пользователь, который назначен на исполнение данного задания, выбирается из числа Потенциальных исполнителей. Он вправе выполнять все действия, связанные с операцией, приостанавливать ее, менять приоритет, отказаться от назначения. Исполнитель может быть выбран только из числа кандидатов.

- *Бизнес администратор* (business administrator), выполняет функции, аналогичные *ответственному*, но не несет ответственности за результат.
- *Информируемый* (notification recipient), это участник, который получает оповещения о заслуживающих его внимания бизнес событиях, например принятии важного решения, отгрузке товара и т.д.

Универсальные пользовательские роли *Потенциальный исполнитель*, *Исключеный из числа исполнителей* и *Актуальный исполнитель* связаны с этапами выполнения операции. Для этого, в спецификации WS-Human Task представлена соответствующая модель жизненного цикла (см. Рисунок 9-2). Опишем основные стадии выполнения операции.

Выполнение операции начинается с состояния «Создано», все данные, необходимые для выполнения подготовлены, происходит внутренняя инициализация задания, начинается отбор потенциальных исполнителей. Для этого осуществляется запрос к внешнему каталогу сотрудников. Запрос возвращает перечень потенциальных исполнителей, соответствующих установленным критериям отбора. По результатам инициализируются универсальные пользовательские роли: «Потенциальные исполнители», «Ответственный» и «Администратор».

Этот список м.б. пустой, содержать имя единственного исполнителя или включать группу лиц, которые могут выполнить задание. В первом случае операция становится доступной сотрудникам с ролями «Ответственный» и «Администратор», они в ручном режиме выбирают пользователей из корпоративного справочника и устанавливают им роль кандидата. Во втором случае, операция м.б. предложена единственному исполнителю или сразу отправлена на «Исполнение». В третьем случае, задание предлагается каждому из кандидатов. Операция в состоянии «Предложена», видна в рабочем портале каждого из кандидатов в виде нового поручения. Теперь любой из кандидатов сможет самостоятельно назначить себя исполнителем выбранного задания. Если назначение состоится, операция изменяет свое состояние на «Зарезервирована», она продолжает

отображаться в рабочем списке только непосредственного исполнителя и становится недоступной остальным кандидатам.

Итак, операция обрела одного актуального исполнителя, присутствует у него в очереди входящих заданий, не видна остальным исполнителям, находится в состоянии «Зарезервировано». Теперь назначенный пользователь может выполнить три действия: делегировать операцию другому исполнителю, отменить свое назначение или приступить к исполнению. При делегировании операция передается другому исполнителю, причем остается в состоянии «Зарезервировано». При отмене назначения задание возвращается в состояние «Готово», при этом снова становится видно всем потенциальным исполнителям. Наконец, если сотрудник решил приступить к исполнению, в его экранном интерфейсе открывается окно, содержащее реальное поручение, а состояние задания меняется на «В работе».

Представим себе, что исполнитель открыл экранное окно, посмотрел поручение, но не захотел выполнять работу и решил завершить сеанс, закрыв ненужное ему задание. При этом операция возвращается в состояние «Зарезервирована», откуда она м.б. повторно вызвана на исполнение, делегирована другому пользователю или возвращена в состояние «Готова», что бы произвести новое назначение исполнителя. Исполнитель, который начал выполнять задание, может работу приостановить, что бы заняться чем-то другим, задание переходит в состояние «Приостановлено», а когда исполнитель освободится, он сможет возобновить работу. Предположим, пользователь не прерывает работу над заданием, выполнил все предусмотренные действия, нажал кнопку закрытия задания «OK», при этом задание переходит в состояние «Завершена». В случае же неудачного завершения задачи (завершение с сообщением об ошибке), задача переходит в состояние Отказано. Если в ходе исполнения задачи возникла неустранимые ошибка, состояние изменится на «Ошибка».

Задача м.б. не обязательной для исполнения и пропущена при выполнении, будет установлено состояние Пропущена.

Состояние «Удалена», соответствует ситуации, когда показатель исполнения процесса (например время) вышел за пределы допустимого диапазона, т.ч задача снята с исполнения.

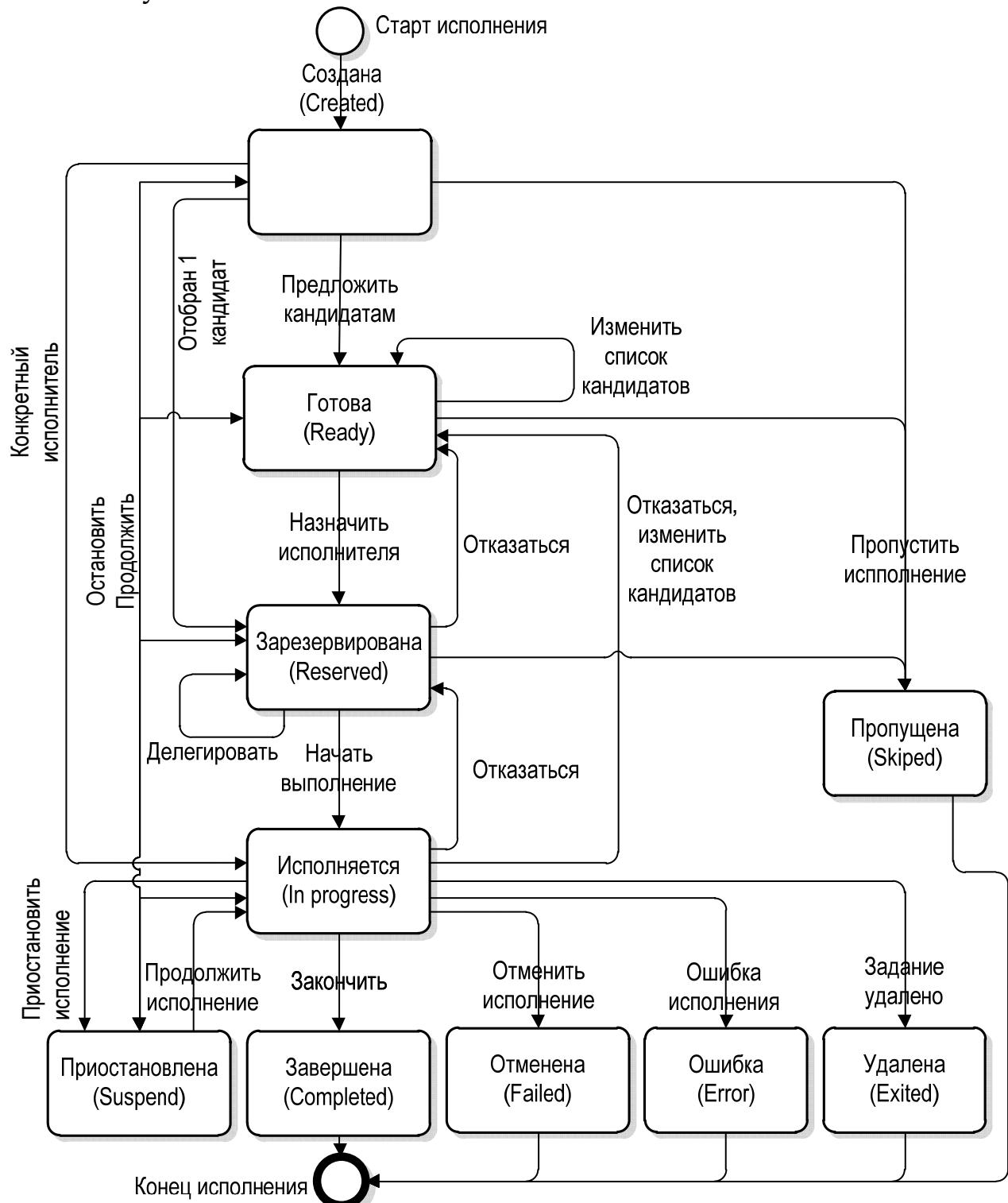


Рисунок 9-2 Основные стадии выполнения операции

10. ПРОЦЕССНЫЕ ПАТТЕРНЫ

В предыдущих разделах мы обсудили по отдельности все элементы нотации BPMN, используемые для оркестровки, описали семантику их исполнения. Однако если связать в определенную последовательность несколько элементов нотации BPMN, то их поведение окажется более сложным, чем когда мы рассматривали их по отдельности.

Термин *паттерн* в информатике обозначает способ решения характерной задачи проектирования, он определяет типовую последовательность действий для определенных стандартных ситуаций. Набор паттернов, возникающих при моделировании оркестровки бизнес-процессов, был описан и размещен на сайте www.workflowpatterns.com. В первоначальном виде этот материал касался в первую очередь систем workflow, затем был адаптирован для BPMN 1.0. Авторы проанализировали работу большинства доступных workflow систем, выделили типовые ситуации и обобщили их в виде типовых последовательностей. В данном разделе мы рассмотрим только типовые модели управления потоками операций, другие типы паттернов: управления данными, обработки исключительных ситуаций и распределения ресурсов остаются вне нашего внимания. Мы не будем рассматривать паттерны в порядке их нумерации, но сгруппируем их так же, как это было предложено авторами классификации.

10.1. БАЗОВЫЕ ПРОЦЕССНЫЕ ПАТТЕРНЫ

В этом разделе мы рассмотрим простейшие типовые последовательности операций. При этом мы обсудим различные варианты реализации типовых ситуаций, допустимые с точки зрения стандарта BPMN 2.0

10.1.1. ПОСЛЕДОВАТЕЛЬНОЕ ИСПОЛНЕНИЕ (СР1)

Последовательное исполнение является наиболее простым способом реализации процесса, когда каждая последующая операция начинается только после завершения предыдущей. При этом выход завершенной операции является входом для следующей, операции соединены только безусловными переходами, ветвления отсутствуют (см. Рисунок 10-1).

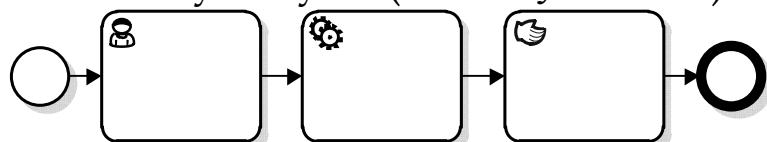


Рисунок 10-1, Последовательное исполнение

Недостатком подобного способа исполнения является высокая длительность исполнения. Можно предположить, что некоторые операции могут исполняться независимо друг от друга, при этом их можно выполнять параллельно.

10.1.2. ПАРАЛЛЕЛЬНОЕ ИСПОЛНЕНИЕ (СР2)

Параллельное исполнение применяется в случае, когда выполняемые операции не зависят одна от другой. Пример (см. Рисунок 10-2) иллюстрирует способы реализации параллельного исполнения. Во-первых, можно разделить входной поток при помощи ЛО «И» (рис А), при этом активируются сразу все выходные ветви, так что операции могут исполняться одновременно. Этот способ является наиболее очевидным с точки зрения нотации. Во-вторых, можно изобразить одну операцию и две стрелки безусловных переходов, исходящих из нее (рис. Б). С точки зрения стандарта BPMN 2.0 конструкция является допустимой, но некоторые средства моделирования не разрешают изображать два безусловных перехода, исходящие из одной операции. В-третьих, можно изобразить вложенный подпроцесс, объединяющий две суб-задачи, которые будут запущены одновременно (рис В).

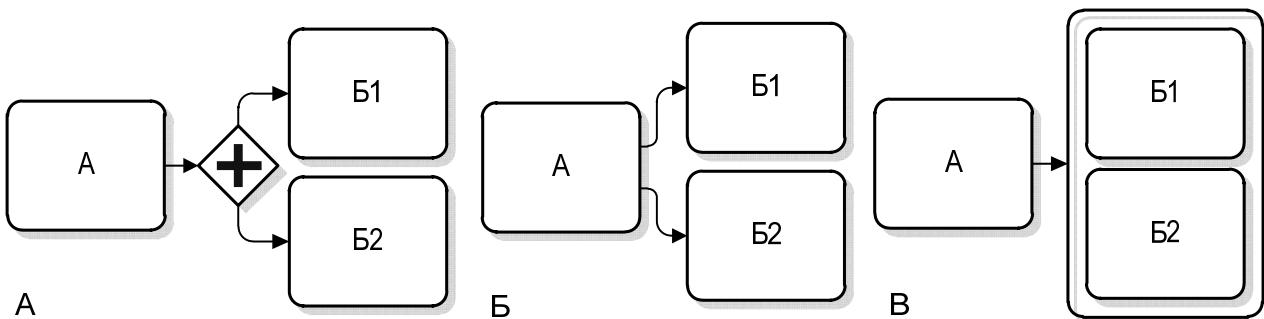


Рисунок 10-2. Параллельное выполнение

10.1.3. СИНХРОНИЗАЦИЯ ПОТОКОВ (СР3)

Синхронизация означает, что несколько параллельных потоков соединяются в один таким образом, что процесс будет продолжен только после прибытия последнего из входящих. Пример (см. Рисунок 10-3) иллюстрирует варианты реализации паттерна синхронизации потоков. Во-первых, можно использовать ЛО «И» (рис. А), который сформирует исходящий поток только после того, как на вход поступят все ожидаемые входящие потоки. Во вторых, можно использовать подпроцесс, содержащий две вложенные задачи (рис. Б).

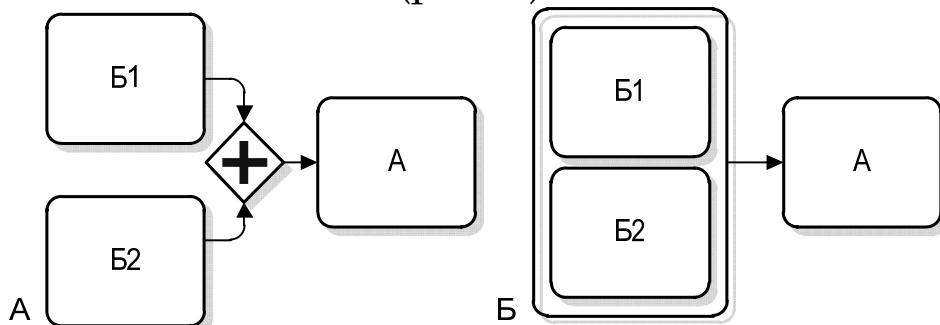


Рисунок 10-3. Синхронизация потоков

10.1.4. АЛЬТЕРНАТИВА (СР4)

Альтернатива предполагает выбор одной из нескольких исключающих друг друга возможностей продолжения. Разветвление потока на несколько альтернативных ветвей может быть реализовано несколькими способами. Во-первых, можно использовать разветвляющий ЛО «Исключающее ИЛИ» (см. Рисунок 10-4А). Если операция A, с которой начинается ветвление - автоматическая, ее следует трактовать и изображать как бизнес правило определения (см. п. 3.1.5), которое устанавливает

критерий применимости какого-либо бизнес понятия, называемого фактом. В зависимости от того, истинен факт или ложен, происходит ветвление на один из альтернативных маршрутов.

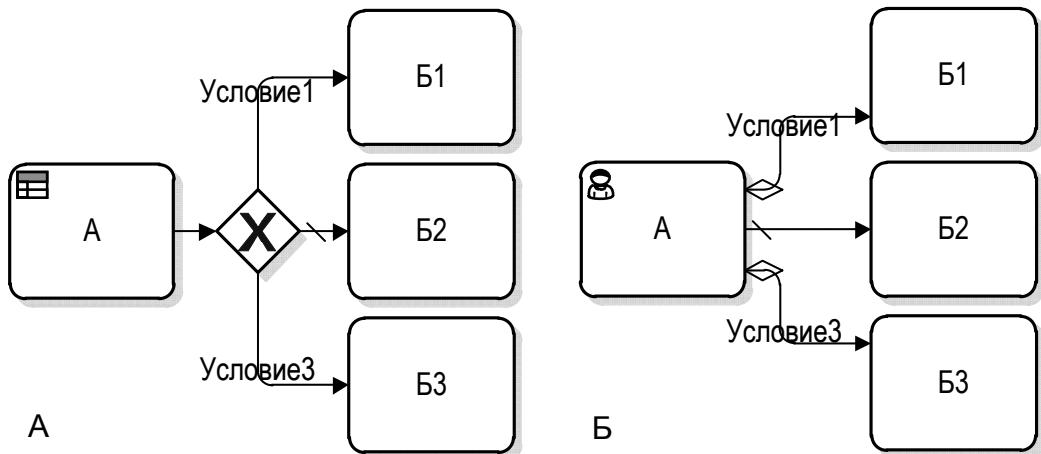


Рисунок 10-4, Выбор взаимоисключающих возможностей

Во-вторых, существует альтернативный способ моделирования такого ветвления, изображаемый несколькими стрелками управления, исходящими из одной задачи (Рис. Б). Переходы, выполняемые по условию, помечены мини ромбами. Один из переходов, выполняемый по умолчанию, если ни одно из условий не верно, помечен косой чертой. Если операция А является интерактивной, тогда паттерн изображают ветвление, выполняемое по явному указанию пользователя. В этом случае, в интерфейсе конечного пользователя д.б. предусмотрено соответствующее число кнопок управления, нажатие на любую приводит к выбору одного из альтернативных вариантов продолжения.

10.1.5. ПРОСТОЕ СЛИЯНИЕ (CP5)

Простое слияние параллельных ветвей описывает такое объединение потоков, когда каждый из входящих потоков порождает соответствующий экземпляр потока на выходе. Образуется несколько маркеров потока управления, которые последовательно движутся друг за другом по одному маршруту. Например, мы разослали запросы в несколько подразделений, каждое подготовило свой ответ, эти ответы рассматриваются по отдельности, по мере их поступления. Следует быть вниматель-

ными, поскольку операция, следующая за простым слиянием, будет выполнена столько раз, сколько маркеров потока управления придет из параллельных ветвей.

Простое слияние альтернативных параллельных потоков может быть реализовано несколькими способами. Во-первых, можно воспользоваться объединяющим ЛО «Исключающее ИЛИ» (см. Рисунок 10-5А). Такое поведение может оказаться неожиданным, поскольку нам кажется, что узел «исключающее ИЛИ» пропускает на выход только один из входящих потоков, а если на вход поступят сразу оба потока, то узел не сработает. Дело в том, что потоки приходят на узел не одновременно, а последовательно, при этом первый поток не дожидается следующего и сразу проходит на выход. Т.о. в текущий момент времени только одна из параллельных ветвей м.б. активной. В большинстве случаев, такое поведение не является ожидаемым, следует позаботиться, что бы входящие ПУ были действительно альтернативными. Для этого ЛО ветвления и слияния «Исключающее ИЛИ» должны использоваться в паре.

Второй вариант предполагает, что на вход некоторой операции соединен с выходами нескольких предшествующих операций. Поток управления по любой из входящих ветвей создает поток управления на выходе, так что каждый из них инициирует выполнение операции В (см. Рисунок 10-5Б). Для этой ситуации справедливы сделанные выше замечания об альтернативности входных потоков. Но в этом случае, найти и отследить поведение парного оператора ветвления будет сложнее.

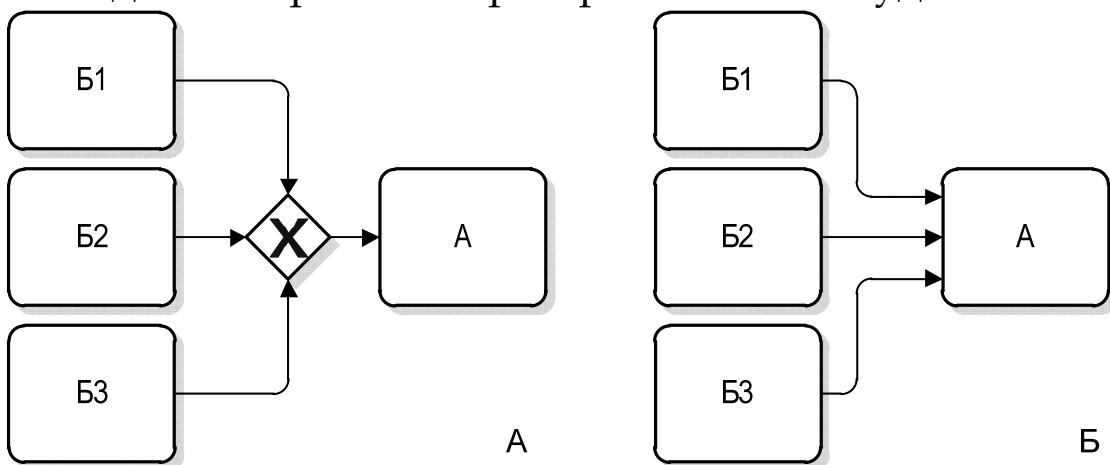


Рисунок 10-5. Простое слияние

10.1.6. МНОЖЕСТВЕННЫЙ ВЫБОР (СР6)

Множественный выбор используется, когда необходимо активировать часть из параллельных ветвей, для которых выполняется некоторое условие. В отличие от *Альтернативы*, может быть выбрано сразу несколько или ни одного варианта продолжения. Например, клиент позвонил в call-центр и сообщил об аварии, оператор должен оповестить оперативные службы. Какие службы будут вызваны, зависит от обстоятельств аварии, может случиться, что будут вызваны все службы или ни одной.

Для реализации паттерна можно использовать три формы записи. Во-первых, можно использовать оператор «ИЛИ» (см. Рисунок 10-6А). Если несколько условий выполняются одновременно, то будут активированы сразу несколько выходных потоков. Во-вторых, можно использовать оператор ЛО «Комплексное условие» (Рис. Б). В-третьих, может быть использована форма записи, когда несколько переходов исходят из одной операции (см. Рис В).

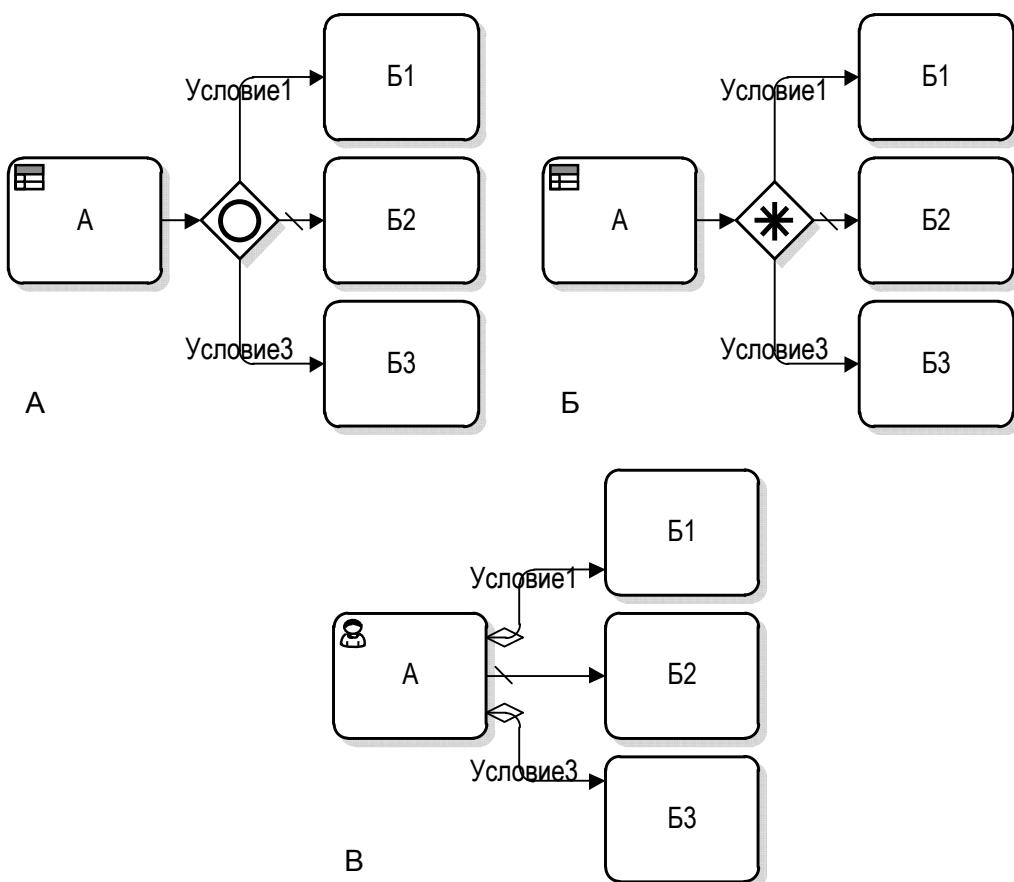


Рисунок 10-6, Множественный выбор

Следует иметь в виду, что для формы записи, изображенной на рисунке В справедливы ранее сделанные замечания о сложности в отслеживания непарных ЛО ветвления и слияния.

10.2. ПАТТЕРНЫ СЛИЯНИЯ И СИНХРОНИЗАЦИИ

Эта группа паттернов описывает более сложные сценарии объединения потоков.

10.2.1. СТРУКТУРИРОВАННЫЕ ПАТТЕРНЫ

Термин структурированный означает, что парные логические операторы должны быть явно видны на схеме процесса. Хочется провести аналогию с открывающей и закрывающей скобками в обычном программировании, они должны быть явно выделены в тексте программы.

10.2.2. СТРУКТУРИРОВАННОЕ СЛИЯНИЕ С СИНХРОНИЗАЦИЕЙ (CR7)

Структурированное слияние с синхронизацией показывает деление одного потока на несколько параллельных ветвей, а затем их слияние в единый поток, который возникнет, только после того, как все ранее активированные параллельные потоки будут завершены. Например, в случае наступления аварии мы оповестили две из трех служб, теперь мы ожидаем, прихода подтверждений оповещения из обеих ветвей. Исполнение будет продолжено, когда поступит последнее оповещение.

Проблема при синхронизации нескольких параллельных ветвей возникает, когда нам заранее не известно число ожидаемых входящих потоков. Одно из возможных решений заключается в том, что бы использовать операторы разветвления и слияния в паре. Этот паттерн называется структурным, поскольку узлы ветвления и слияния используются парно.

Пример (см. Рисунок 10-7) иллюстрирует использование парных операторов ветвления и слияния «Исключающее

ИЛИ». Параллельные маршруты являются альтернативными, поэтому на выходе формируется только один ПУ.

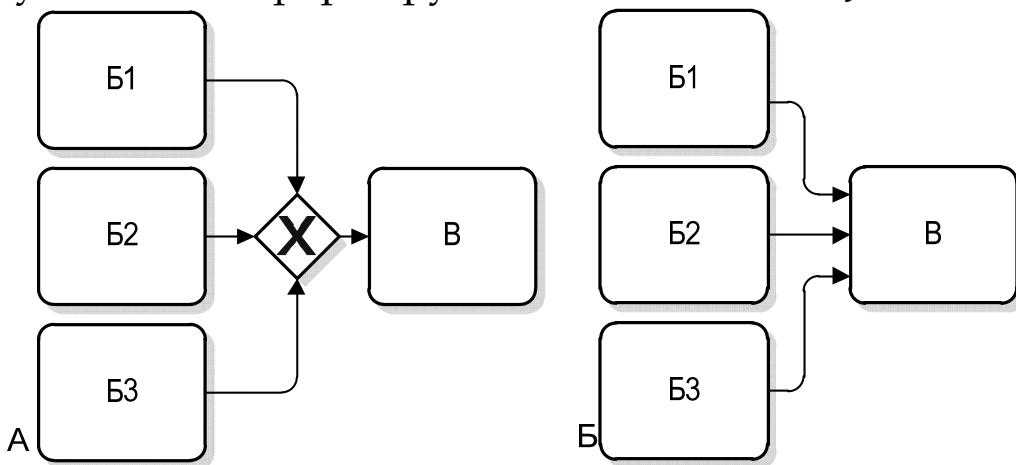


Рисунок 10-7. Парные операторы альтернативный выбор

Пример (см. Рисунок 10-8А) показывает парные ЛО ветвления и слияния «И». Узел слияния ждет поступления (синхронизации) всех входных потоков. Пример (см. Рис. Б) показывает парные ЛО ветвления и слияния «ИЛИ». Как ранее отмечалось, ЛО слияния узнает число входящих ветвей от предшествующего парного ЛО ветвления (см. п. 5.2.2). Если при ветвлении была активирована только одна ветвь, то на выходе слияния сразу возникает выходной поток, а если при ветвлении были активированы несколько потоков, то следует ждать окончания (синхронизации) последнего. В обоих случаях одному потоку на входе соответствует ровно один на выходе.

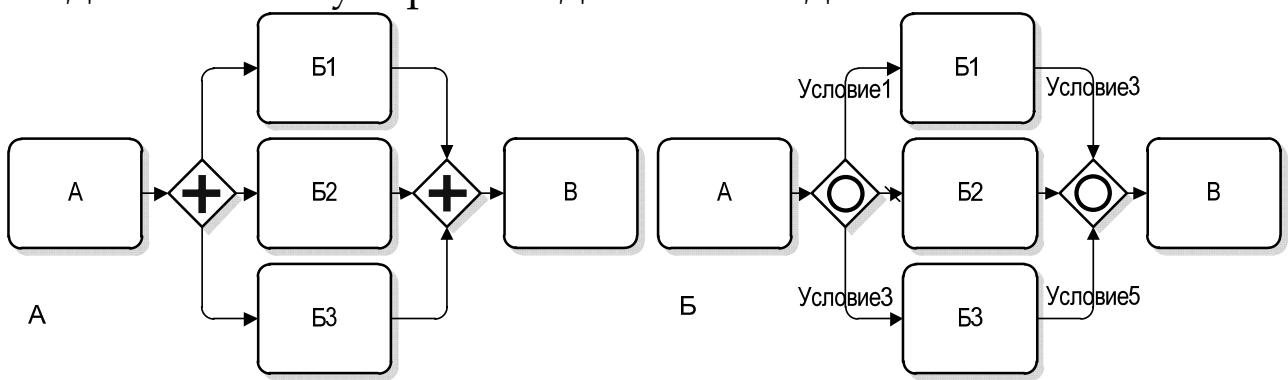


Рисунок 10-8. Структурированное слияние с синхронизацией

10.2.3. НЕСТРУКТУРИРОВАННЫЕ ПАТТЕРНЫ

Неструктурированные паттерны, где узлы ветвления и слияния используются непарно. Усложним пример из предыдуще-

го параграфа, разместим между парными узлами ветвления и слияния «И» дополнительный ЛО ветвления «исключающее ИЛИ» (см. Рисунок 10-9). В результате первого ветвления «И» возникнут два потока, направленные на операции Б1 и Б2. После завершения операции Б1 выходной поток поступает на ЛО «исключающее ИЛИ». Если выполняется Условие1, то следующей будет исполнена операция Г, т.ч. поток из этой ветви не поступит на вход узла слияния «ИЛИ». Поскольку при первом ветвлении были созданы два потока, узел слияния «И» будет ожидать появления двух ПУ. Однако после завершения операции Б1 один из потоков не поступит на узел слияния. Возникает тупиковая ситуация (DEAD LOCK).

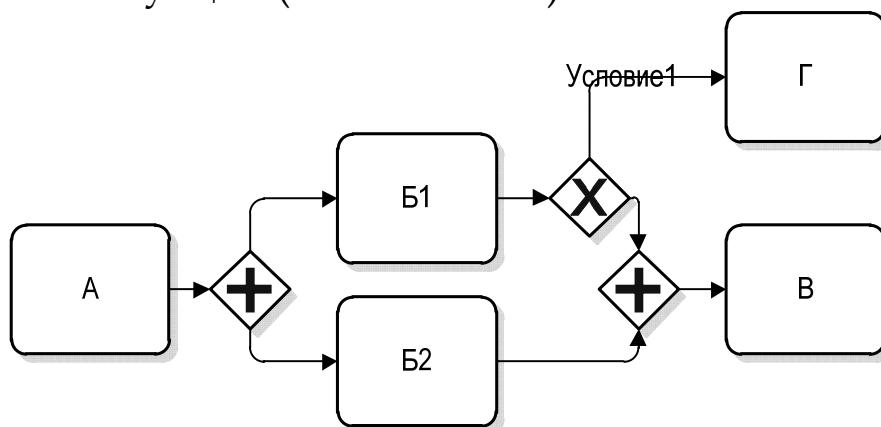


Рисунок 10-9 Неструктурированный паттерн

Но если мы добавим в схему парный узел слияния «ИЛИ», то коллизия исчезает, тупиковая ситуация оказывается невозможна (см. Рисунок 10-10).

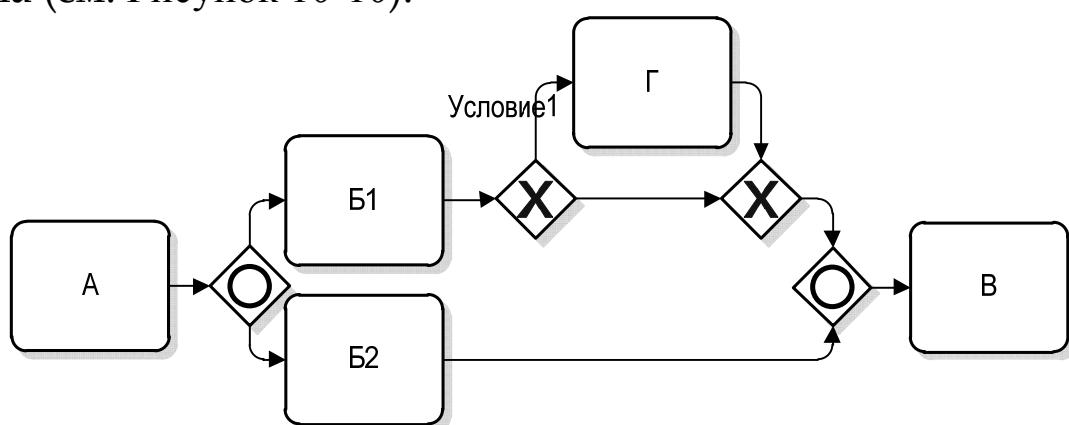


Рисунок 10-10 Структурированный паттерн

10.2.4. МНОЖЕСТВЕННОЕ СЛИЯНИЕ (CP8)

Множественное слияние демонстрирует пример непарного использования ЛО ветвления и слияния. Его особенность в том, что каждый из потоков, прибывающих по параллельным входным ветвям, формирует на выходе отдельный маркер потока управления. В результате операция, которая расположена после узла слияния, будет выполнена столько раз, сколько параллельных потоков поступило на вход узла слияния.

Пример (см. Рисунок 10-11) показывает модель, где осуществляется ветвление на параллельные потоки, а слияние «исключающее ИЛИ» пропустит на выход поток из каждой параллельной ветви. В результате, операция В будет выполнена столько раз, сколько параллельных ветвей приходит на узел слияния. Аналитик должен крайне внимательно относится к не структурированным паттернам, в том числе непарному слиянию, поскольку такое поведение может идти в разрез с первоначально планируемой логикой процесса.

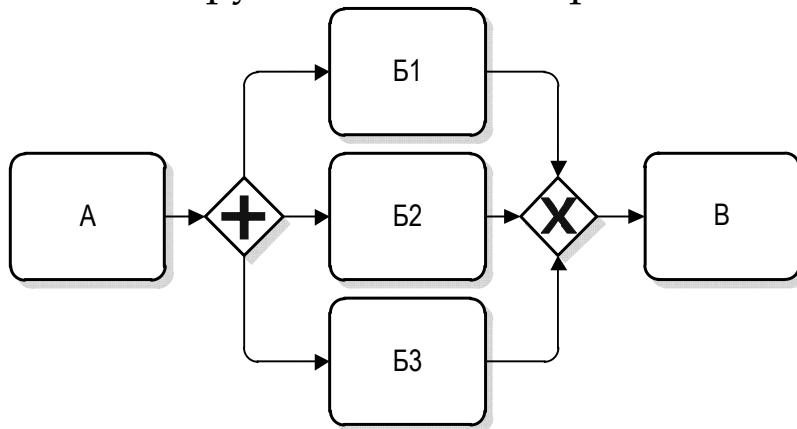


Рисунок 10-11, Непарные узлы ветвления и слияния

10.2.5. ДИСКРИМИНАТОР (CP9)

Дискриминатор описывает слияние нескольких параллельных потоков управления таким образом, что после поступления первого потока управления из параллельных входных ветвей, узел слияния вырабатывает маркер на выходе, при этом он продолжает принимать оставшиеся входящие ПУ, но игнорирует их. Общее число входящих потоков заранее известно. Только когда поступит последний параллельный ПУ, узел бу-

дет сброшен в исходное состояние, так что он будет готов снова выполнить свою функцию. Структурный дискриминатор отличается от паттерна Синхронизация, тем, что первый ждет появления первого потока управления, а второй всех входных потоков управления.

Например, мы проводим конкурс и рассылаем приглашения участникам. Число кандидатов заранее известно. Кандидаты задают вопросы и получают на них ответы, затем дают ответ на приглашение. Победителем будет объявлен первый, кто пришлет подтверждение. При этом мы продолжим отвечать на вопросы остальных участников. Конкурс будет завершен только после получения и обработки всех ответов на разосланные приглашения.

Простейший способ реализовать дискриминатор заключается в использовании ЛО комплексное условие. Пример (см. Рисунок 10-12А) изображает схему, на которой ветвление осуществляется с использованием ЛО «И», а для объединения потоков используется ЛО комплексное условие.

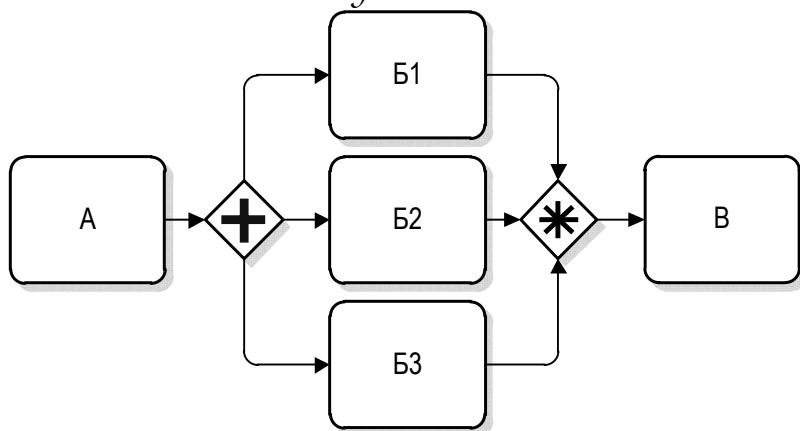


Рисунок 10-12, Дискриминатор

На исполнение параллельных ветвей следует наложить определенные ограничения. Например, они не должны создавать и не уничтожать потоки управления. Т.о. выполнение паттерна связано со следующими условиями:

- Паттерн описывает объединение потоков, которые были ранее разделены одним узлом ветвления;
- В параллельных ветвях допускаются только парные ЛО ветвления и слияния;

- Каждая из ветвей м.б. исполнена только однократно.
- Ни одна из параллельных ветвей не м.б. завершена независимо от остальных. Ветви м.б. завершены только если работа всего процесса будет прекращена.

В общем случае, паттерн *дискриминатор* может использоваться, что бы объединить и синхронизировать некоторое подмножество из всего множества входящих потоков. Например, на вход приходит M параллельных потоков, но нас интересует синхронизация только первых N из них ($N < M$). После того, как искомое число N ПУ из параллельных ветвей поступило, узел вырабатывает выходной ПУ, при этом он продолжает принимать оставшиеся входящие ($M-N$) ПУ и игнорирует их. Только когда поступит последний поток номер M из параллельной ветви, узел будет сброшен в исходное состояние, так что будет готов снова выполнить свою функцию.

Альтернативный способ реализации дискриминатора предполагает использование задачи, называемой *Счетчик*. Последний суммирует число прошедших через него маркеров потока управления, после чего вырабатывает выходной сигнал (см. Рисунок 10-13). Поскольку следующим исполняется промежуточное событие ошибки, которое вырабатывает сигнал *ошибки*, прерывающий работу всего подпроцесса. В этом случае, все еще не завершенные параллельные потоки будут остановлены. Перехватывающее прикрепленное событие-ошибки указывает на сценарий продолжения. После того, как работа подпроцесса будет завершена, начнется исполнение задачи *B*.

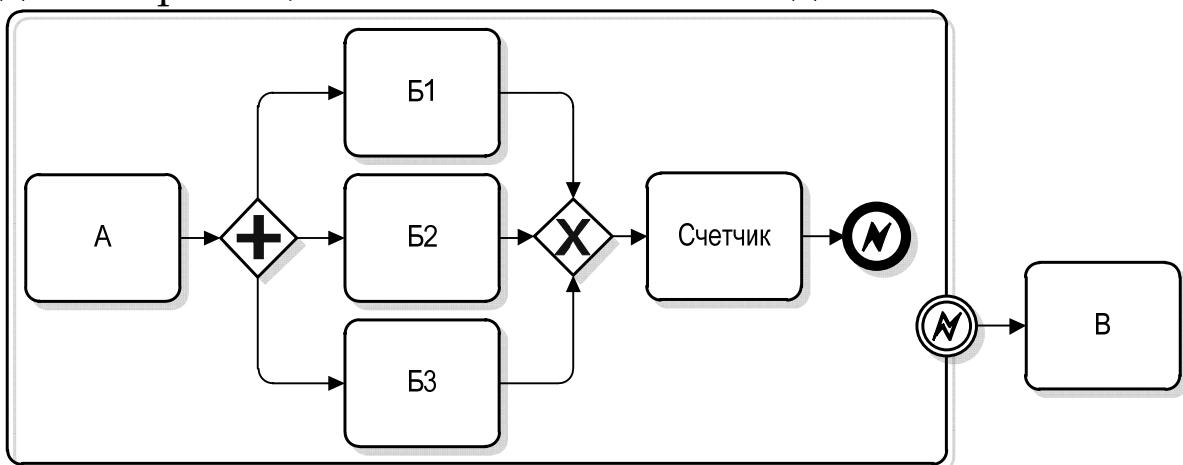


Рисунок 10-13. Дискриминатор, счетчик

Следует позаботиться, что бы ожидаемое число сигналов не превышало количества входных ветвей узла слияния, в противном случае возникнет тупиковая ситуация.

10.3. ИТЕРАЦИИ

Эта группа паттернов описывает ситуации, когда операция или подпроцесс исполняются итеративно, в цикле.

10.3.1. МНОГОКРАТНОЕ ПОВТОРЕНИЕ (CP10)

Многократное повторение описывает повторное исполнение отдельной операции или группы задач. Часто возникает ситуация, когда некоторый фрагмент процесса требуется многократно повторить. Для реализации такой конструкции можно использовать несколько вариантов записи.

Пример (см. Рисунок 10-14А) показывает реализацию цикла Do-While. Если результат обработки не достигнут, то происходит ветвление потока и повторное выполнение задачи А. Если результат достигнут, то выполняется операция Б. Например, если в результате обработки обнаружен исправимый брак, то работу следует выполнить повторно. Второй пример (см. Рис. Б) показывает реализацию цикла Repeat-Until.

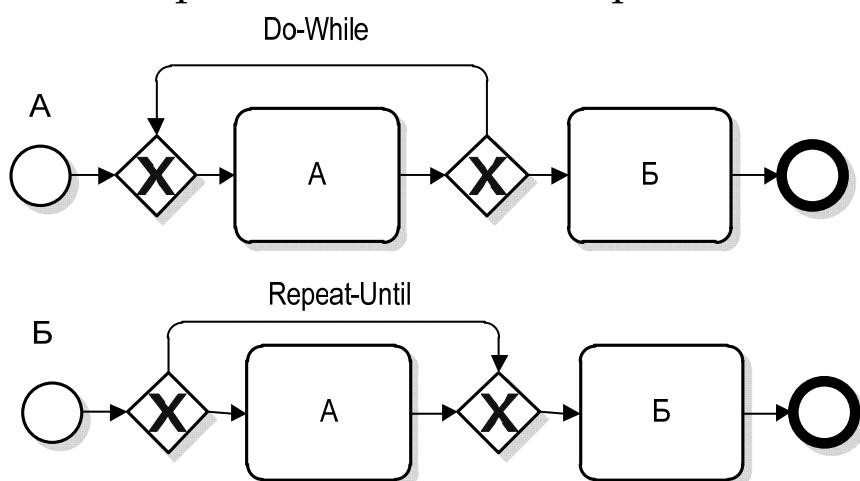


Рисунок 10-14. Многократное повторение

10.3.2. ИТЕРАЦИЯ (CP21)

Итерация описывает многократное повторение задания или подпроцесса. Пример (см. Рисунок 10-14) показывает подпро-

цесс, исполняемый итеративно столько раз, сколько указано в условии повторения. Есть возможность указать способ исполнения циклов: Do-While или Repeat-Until (см. п. 3.2.3).

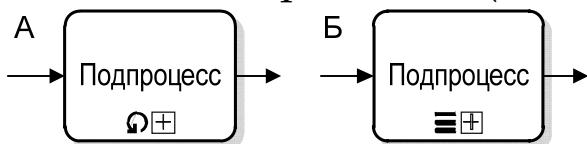


Рисунок 10-15. Итерация

Следует обратить внимание на существенные отличия в исполнении простого вложенного и глобально известного подпроцессов. Вложенный подпроцесс имеет непосредственный доступ к объектам данных процесса. Поэтому, результаты, полученные в ходе первой итерации, будут перезаписаны при втором повторении и т.д. Т.о. поле завершения нужного числа повторений сохраняется результаты только последней итерации.

Глобально известный подпроцесс не имеет доступа к данным родительского процесса, поэтому, при каждой итерации будет создаваться новая копия подпроцесса, со своим набором данных. Однако когда все итерации будут завершены, возникает задача объединить все данные разных экземпляров и вернуть результат в вызывающий родительский процесс.

10.3.3. РЕКУРСИВНОЕ ИСПОЛНЕНИЕ (CP22)

Рекурсия в программировании есть вызов функции или процедуры из неё же самой. В нотации BPMN рекурсия возможна только для повторно используемых процессов. Поскольку при каждом вызове подпроцесс порождается новая копия данных, рекурсивный вызов подпроцесса не нарушит целостности данных.

10.4. ПАРАЛЛЕЛЬНОЕ ИСПОЛНЕНИЕ

Паттерны в этом разделе описывают ситуацию, когда одна операция на модели процесса будет исполнена многократно, при этом возникнет несколько экземпляров потока управления, исполняемых одновременно. Будем различать параллель-

ное исполнение и последовательные итерации. В первом случае одна операция или группа задач исполняются последовательно требуемое число раз. При этом в любой момент времени существует только один поток управления, который обрабатывает один набор данных, так что результаты, полученные в результате первого прогона, поступят на вход второго и т.д. Во втором случае возникает много параллельных потоков управления, ветви исполняются одновременно и асинхронно друг от друга. Например, логический оператор ветвления «И» порождает несколько потоков управления («Параллельное исполнение (CP2)»). При этом возникает вопрос, работают ли параллельные экземпляры с одним набором данных или каждый со своей копией. В первом случае следует продумать ситуацию, связанную с одновременным доступом разных потоков к общим данным. При этом возможны ситуации конкуренции доступа, блокировки, гонки и т.д. Во втором случае, необходимо предусмотреть консолидацию данных из параллельных ветвей. Например, при параллельном согласовании возможна ситуация, когда в одной ветви согласование прошло, а во второй появились замечания. Теперь надо консолидировать информацию из обеих ветвей и принять согласованное решение.

10.4.1. ПАРАЛЛЕЛЬНОЕ ИСПОЛНЕНИЕ БЕЗ СИНХРОНИЗАЦИИ (CP12)

Параллельное исполнение без синхронизации моделирует параллельное исполнение некоторой операции или группы операций. В отличие от паттерна многократного повторения CP10, этот паттерн для каждой итерации порождает отдельный маркер потока управления на выходе, каждый со своим набором данных. Синхронизация (ожидание окончания последнего из них) отсутствует.

Пример (см. Рисунок 10-16) изображает задачу, исполняемую в цикле. При каждой итерации будет создан отдельный экземпляр данных, в окружении которых будет исполняться образ процесса.

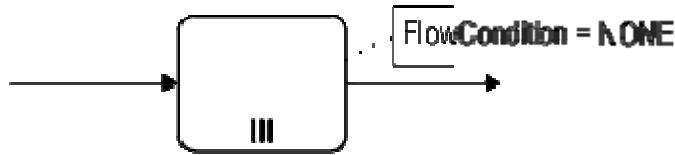


Рисунок 10-16. Параллельное исполнение без синхронизации

Внутренний атрибут **MultInstanceBehavior** определяет способ синхронизации экземпляров задачи (см. Таблица 3-4. Семантика параллельного исполнения). Значение **None**, что маркеры потока управления создаются после завершения каждого экземпляра. На выходе образуется столько маркеров потока управления, сколько параллельных экземпляров было исполнено. Значение **One** указывает, что один маркер потока управления создается после завершения первого экземпляра. Величина **All**: определяет, что будет создан только один маркер потока управления, причем после завершения всех параллельных экземпляров. Наконец, условие **Complex** задает составное, комплексное условие создания маркеров потока управления.

10.4.2. ПАРАЛЛЕЛЬНОЕ ИСПОЛНЕНИЕ С СИНХРОНИЗАЦИЕЙ, ЧИСЛО ЭКЗЕМПЛЯРОВ ИЗВЕСТНО НА ЭТАПЕ МОДЕЛИРОВАНИЯ (CP13)

Параллельное исполнение с синхронизацией (число экземпляров известно на этапе моделирования) реализуется аналогично предыдущему, при этом, число повторений задается внутренним атрибутом **loopCardinality**, который имеет значение константа. Пример (см. Рисунок 10-17) иллюстрирует паттерн параллельного исполнения.

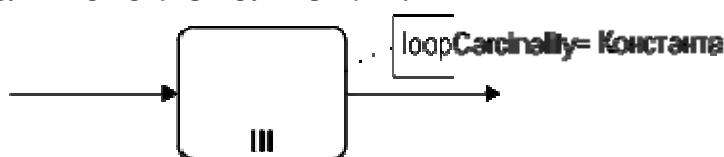


Рисунок 10-17. Параллельное исполнение с синхронизацией (число экземпляров известно на этапе моделирования)

10.4.3. ПАРАЛЛЕЛЬНОЕ ИСПОЛНЕНИЕ, ЧИСЛО ЭКЗЕМПЛЯРОВ ИЗВЕСТНО НА ЭТАПЕ ИСПОЛНЕНИЯ (CP14)

Параллельное исполнение с синхронизацией (число экземпляров известно на этапе исполнения) реализуется аналогично предыдущему, при этом, число повторений задается внутренним атрибутом loopCardinality, значение которого определяется формулой, которая вычисляется на этапе исполнения. Пример (см.Рисунок 10-18) иллюстрирует паттерн параллельного исполнения.

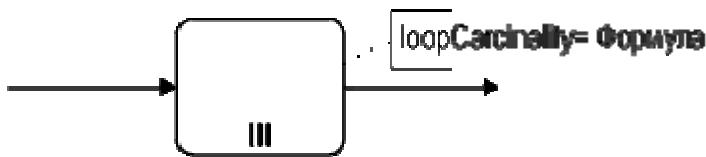


Рисунок 10-18. Параллельное исполнение с синхронизацией (число экземпляров известно на этапе исполнения)

10.5. СТАТУС ИСПОЛНЕНИЯ

Весь маршрут исполнения процесса можно разделить на этапы, каждый из которых связан с достижением определенной цели. Обычно этап завершается принятием решения, важного для выполнения всего бизнес-процесса в целом. Если ожидаемый результат достигнут, то происходит нормальное продолжение – переход на следующий этап, а ситуация, когда результат не достигнут, рассматривается как брак, происходит отказ в обслуживании или возврат на повторную обработку. Результат выполнения этапа есть показатель продукта процесса, он фиксируется в информационном объекте и определяет статус исполнения процесса. Таким образом, что бы проверить, успешно ли закончен очередной этап, следует оценить состояние соответствующего информационного объекта. Перечисленные ниже паттерны связаны с анализом состояния статуса исполнения процесса.

10.5.1. ОТЛОЖЕННЫЙ ВЫБОР (CP16)

Отложенный выбор показывает ветвление процесса в зависи-

мости от внешнего события, произошедшего в некотором другом процессе. Мы сможем узнать о наступлении события благодаря оповещению, пересылаемому в форме сообщения. Можно предложить рассмотреть несколько вариантов решения задачи (см. Рисунок 10-19).

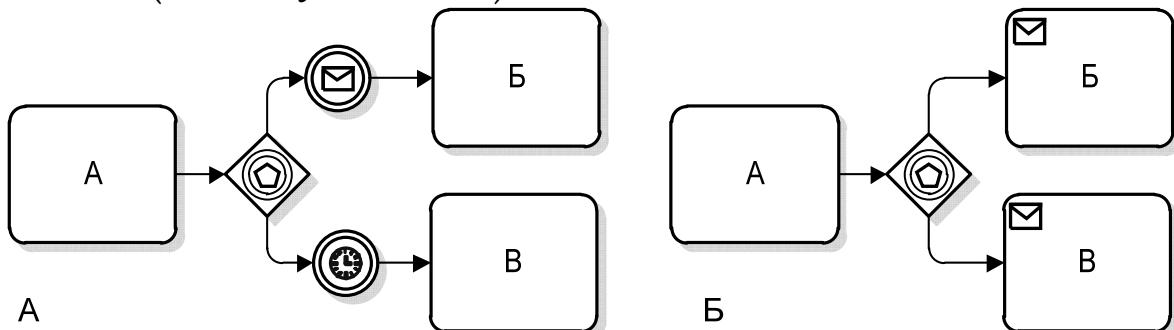


Рисунок 10-19, Отложенный выбор

В первом случае (см. Рис. А), используется событийный оператор «Исключающее ИЛИ», который моделирует решение, управляемое результатом события, причем с каждым возможным выходным маршрутом связанно отдельное перехватывающее событие. Таким образом, вариант продолжения зависит от результатов исполнения внешнего процесса.

Во втором случае (см. Рис. Рисунок 10-19Б), вместо элемента событие используются задания Отправить-Получить сообщения. Как мы знаем, семантика этих элементов не отличается.

10.5.2. ЧЕРЕДОВАНИЕ МАРШРУТОВ (СР17)

Чередование маршрутов описывает ситуацию, когда в процессе фиксируется только набор операций, а порядок их исполнения и число повторений в модели не определены. Каждый набор может включать единственную операцию или некоторую последовательность операций. Порядок, в котором исполняются наборы и число повторений, определяет пользователь в момент исполнения. Причем в любой момент времени может исполняться только один из наборов.

Для реализации паттерна следует использовать ситуационные Ad-Hoc процессы (см. п. 3.2.5). Пример (см. Рисунок 10-20А) показывает задачу с маркером Ad-Hoc, которая содержит две подзадачи А и Б. Выбор нужной подзадачи осуществля-

ляет исполнитель. Во втором случае (см. Рис. Б), операция с маркером Ad-Hoc, которая содержит две упорядоченные последовательности действий A1-A2 и Б1-Б2. Выбор варианта продолжения осуществляется пользователем. Третий вариант (см. Рис. В), предполагает, что выбор маршрута продолжения осуществляется с использованием сообщений, отправляемых извне.

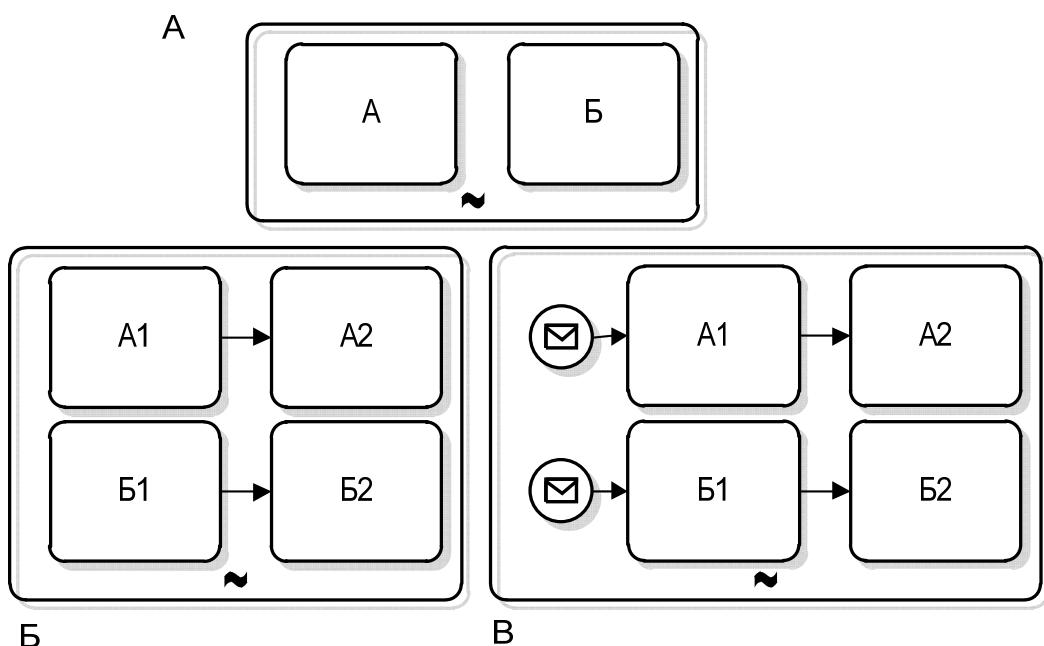


Рисунок 10-20, Чередование маршрутов

10.5.3. КООРДИНАЦИЮ ИСПОЛНЕНИЯ (CP18)

Координация исполнения описывает согласованное выполнения двух процессов или параллельных ветвей одного процесса, когда операция в одной ветви может быть исполнена, если маркер потока в другой ветви уже достиг определенного этапа. И наоборот, если маркер потока в другой ветви уже закончил некоторый этап, выполнение задачи запрещено. Это означает, что операция д.б. снята с исполнения.

Реализация этого сценария осложняется тем, что в нотации BPMN не предусмотрены средства, позволяющие управлять статусом исполнения задачи (см. п. 9.5). Хотя большинство BPMS систем включают программный интерфейс, что бы «приостановить-продолжить» исполнение задания, однако он не доступен для моделирования на схеме процесса.

Пример, (см. Рисунок 10-21) иллюстрирует ситуацию, когда операция Б может быть исполнена только параллельно с операцией А. Для моделирования этапов исполнения используются промежуточные события, размещенные в одной из ветвей. Первое разрешает исполнение, а второе запрещает. Для управления исполнением другой ветви можно использовать прикрепленные обрабатывающие прерывающие события. Недостаток этой схемы заключается в том, что если операция Б не будет завершена до окончания задачи А, она будет прервана, а результаты работы утеряны. К тому же, она не применима к параллельным ветвям одного процесса, поскольку базируется на сообщениях.

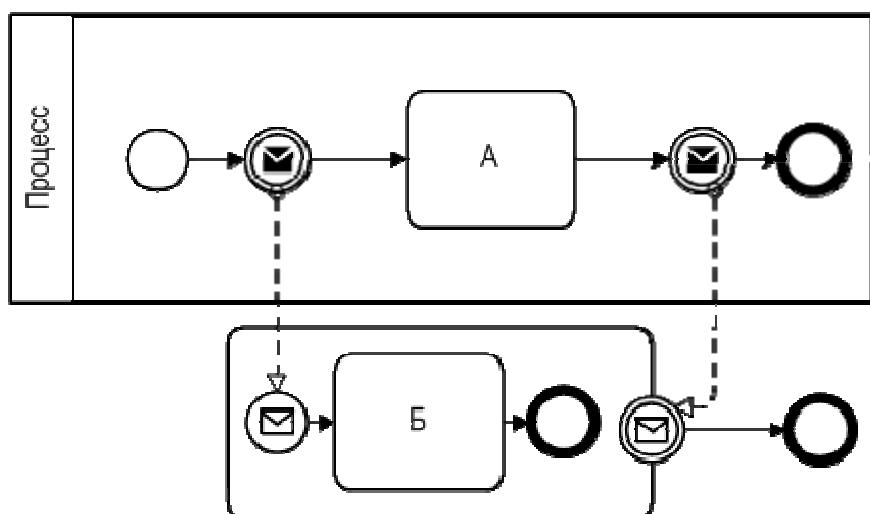


Рисунок 10-21. Координация исполнения

10.6. ПАТТЕРНЫ ЗАВЕРШЕНИЯ

Эти паттерны описывают сценарии завершения отдельной ветви процесса, подпроцесса или всего процесса целиком

10.6.1. ЯВНОЕ ЗАВЕРШЕНИЕ (CP11)

Явное завершение означает, что процесс заканчивается, когда завершится исполнение всех порожденных в данном процессе потоков управления, включая вложенные и вызываемые подпроцессы. Для этого все ветви процесса, которые не соединяются с другими ветвями, должны заканчиваться явным завершающим событием.

Пример (см. Рисунок 10-22А) показывает процесс, который имеет завершающие события в обеих параллельных ветвях. Завершение первой из них не окажет влияние на вторую, процесс полностью завершится только после окончания второй ветви. Что бы завершение ветви прекращало исполнение всего процесса, следует использовать завершающее событие отмена (см. Рис. Б).

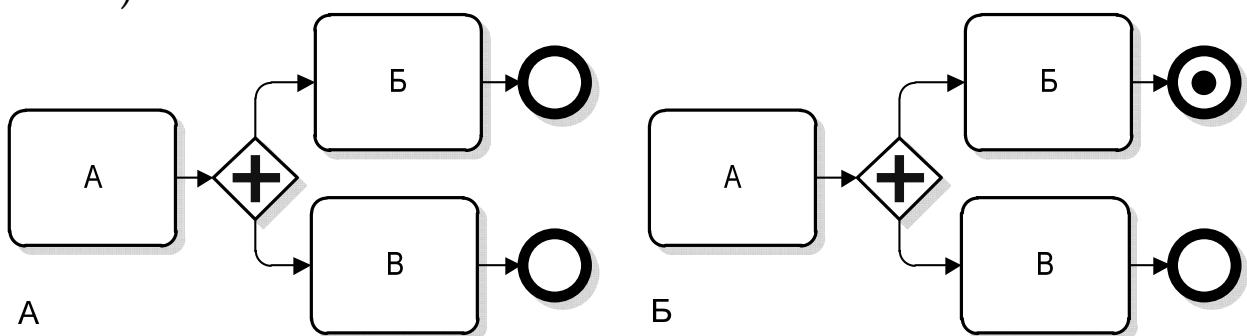


Рисунок 10-22. Явное завершение

Однако модель с явным завершением может иметь не детерминированную семантику исполнения (см. Рисунок 10-23). Если ветвь Б завершится первой, будет инициировано завершающее событие, которое прекратит исполнение во всех, еще не завершенных ветвях процесса. Т.ч. выполнение ветви В будет прервано. Однако мы не знаем, была ли завершена операция В и с каким результатом, была ли начата операция Г.

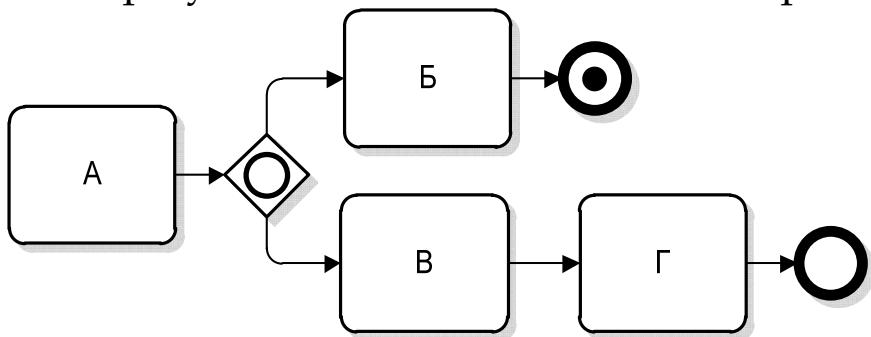


Рисунок 10-23. Явное завершение, не детерминированная семантика исполнения

10.6.2. ПРЕКРАТИТЬ ИСПОЛНЕНИЕ ЗАДАНИЯ (CP19)

Прекращение исполнения задания описывает ситуацию, когда по каким-то причинам необходимо прервать и прекратить

исполнение только одной операции процесса.

Пример (см. Рисунок 10-24) показывает задачу А с прикрепленным прерывающим событием. В случае наступления события, выполнение этой задачи будет перервано, будет вызван обработчик, который передаст управление в задачу Б. Результаты работы задачи А будут потеряны.

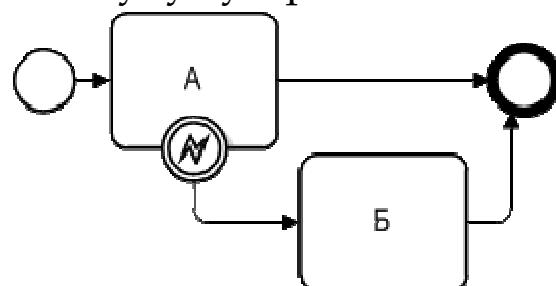


Рисунок 10-24. Прекратить исполнение задания

10.6.3. ПРЕКРАТИТЬ ИСПОЛНЕНИЕ ПРОЦЕССА (CP20)

Прекращение исполнения процесса, описывает ситуацию, когда по каким-то причинам необходимо прервать и прекратить исполнение всего процесса целиком. Результаты ранее выполненной работы будут потеряны, если они не записаны на внешних хранилищах данных. Пример (Рисунок 10-25) показывает подпроцесс А с прикрепленным прерывающим событием. В случае наступления события, работа этого подпроцесса будет перервана, будет вызван обработчик, который передаст управление а задачу Б. Результаты работы задачи А будут потеряны.

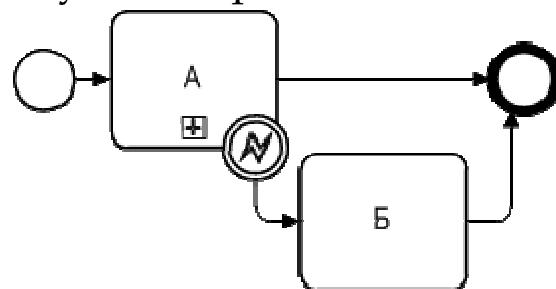


Рисунок 10-25. Прекратить исполнение процесса

10.6.4. ПРЕКРАТИТЬ ИСПОЛНЕНИЕ ГРУППЫ ОПЕРАЦИЙ (CP25)

Прекращение исполнения группы задач, описывает ситуа-

цию, когда по каким-то причинам необходимо прервать и прекратить исполнение одной или нескольких задач. Задачи не обязательно д.б. связаны между собой.

Пример (см. Рисунок 10-26) изображает группу из двух задач А и Б, к которой прикреплено прерывающее событие. В случае наступления события, работа этой группы будет прервана, будет вызван обработчик, который передаст управление а задачу Г. Результаты работы группы А и Б будут потеряны.

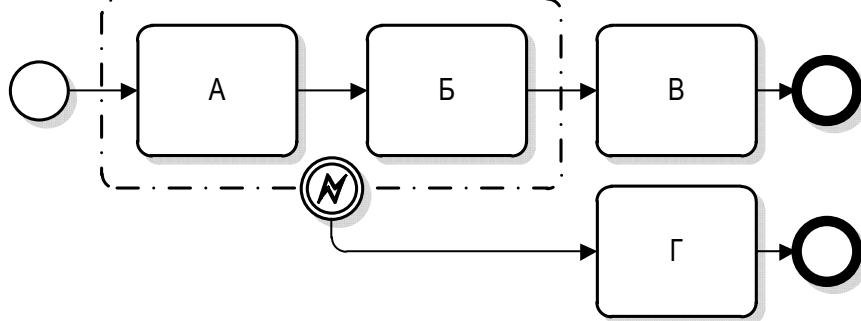


Рисунок 10-26. Прекратить выполнение группы задач

10.6.5. ПРЕКРАТИТЬ ИСПОЛНЕНИЕ МНОГОПОТОЧНОЙ ОПЕРАЦИИ (CP26)

Прекращение исполнения многопоточной операции, описывает ситуацию, когда по каким-то причинам необходимо прервать и прекратить исполнение задачи, имеющей маркер многопоточного исполнения. Выполнение многопоточной задачи м.б. прервано в любой момент времени. При этом, те экземпляры, которые еще не завершились, будут остановлены и завершены, а результаты их работы утеряны. Остановка не затронет ранее завершившиеся экземпляры операции. Этот паттерн использует особенности т.н. прерывающих событий, которые прикрепляются к соответствующей операции.

Пример (см. Рисунок 10-27) изображает прикрепленное событие, которое прерывает исполнение всех еще не завершенных экземпляров, порожденных многопоточной операцией А.

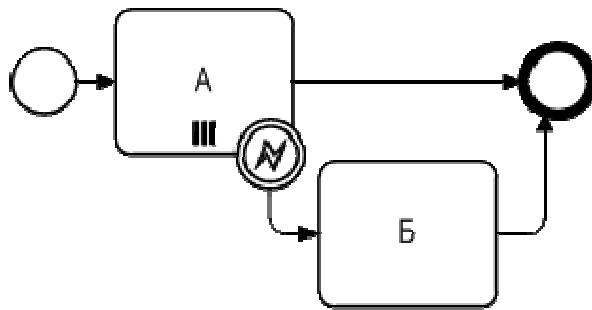


Рисунок 10-27. Прекратить исполнение многопоточной задачи

10.7. СИНХРОНИЗАЦИЯ С ПОМОЩЬЮ СОБЫТИЙ

Эти паттерны описывают взаимодействие (синхронизацию) процесса с другими процессами посредством механизма события. В параграфе (4.3) мы договорились разделять события как явления и события как программу обработчик. Схема моделируемого процесса может включать элемент событие, которое приостанавливает выполнение до тех пор, пока не будет получен сигнал из внешнего источника. Этот сигнал инициирует продолжение исполнения. Будем разделять: обработчики без запоминания события и с запоминанием. Эти паттерны находят широкое применение при оркестровке нескольких взаимодействующих процессов.

10.7.1. СИНХРОНИЗАЦИЯ БЕЗ ЗАПОМИНАНИЯ ОПОВЕЩЕНИЯ (CP23)

Синхронизация без запоминания оповещения описывает ситуацию, когда исполнение процесса останавливается после достижения элемента промежуточное событие. Исполнение будет продолжено, только после прихода оповещения. Паттерн называется обработчик без запоминания события, поскольку он сработает, только если поток управления уже находится на элементе событие и ожидает прихода оповещения. Если событие произошло, но ПУ еще не пришел до элемента промежуточное событие, то оповещение не запоминается и теряется. В качестве оповещения могут использоваться события и сигналы.

Пример (см. Рисунок 10-28) иллюстрирует Синхронизацию

без запоминания события между Процессом1 и Процессом2. После выполнения задачи В маркер потока управления доходит до промежуточного обрабатывающего элемента события и останавливается. Если из Процесса1 поступит сигнал, исполнение будет продолжено, начнет исполняться операция Г. Если оповещение поступит до того, как маркер потока управления достигнет узла промежуточное событие, оно будет утеряно.

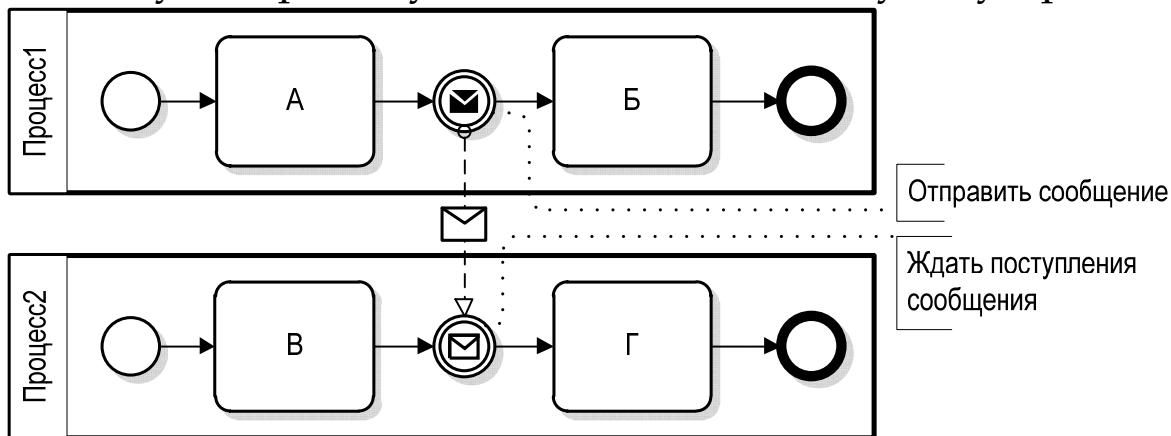


Рисунок 10-28. Синхронизация без запоминания оповещения

10.7.2. СИНХРОНИЗАЦИЯ С ЗАПОМИНАНИЕМ ОПОВЕЩЕНИЯ (SP24)

Синхронизация с запоминанием оповещения работает так же как предыдущий паттерн, но в добавок он позволяет обработать ситуацию, когда событие произошло, а маркер потока управления еще не дошел до элемента промежуточное событие. При этом оповещение запоминается и не теряется, так что позднее, когда маркер достигнет узла, он сможет обработать оповещение и продолжит исполнение без ожидания.

Нотация BPMN не поддерживает этот паттерн, однако он может реализовываться внешними средствами. Некоторые BPMS реализуют интеграцию с внешними средствами работы с очередями сообщений. Последние помогают запомнить сообщение. Для других видов оповещений такая возможность не предусмотрена.

11. ДИАГРАММЫ ВЗАИМОДЕЙСТВИЯ

Схема взаимодействия (collaboration) показывает информационный обмен сообщениями между двумя и более участниками, управляемыми по отдельности, каждый из своего единого центра. Следует говорить о взаимодействии типа B2B, осуществляемом между несколькими юридическими лицами. Т.о. можно рассматривать диаграммы взаимодействия как дальнейшее развитие публичных процессов (см. п. 2.5.1.3).

Организация участник взаимодействия на схеме моделируется пулом. Пул может однозначно идентифицировать участника по имени организации, например: «Первый народный банк», «Государственная страховая компания» и т.д., либо указывать обобщенное имя, которое характеризует группу участников, например: «Поставщик», «Покупатель», «Заказчик». Каждый пул может содержать дорожки, отображающие роли сотрудников, работающих в соответствующем юридическом лице.

Вспомним, что потоки управления не могут пересекать границы пула, т.ч. для моделирования информационного взаимодействие между процессами следует использовать потоки сообщений. Неправильно ассоциировать потоки сообщений с конкретными физическими способами передачи информации между участниками, например с электронной почтой, факсом или обычной почтой.

11.1. УРОВНИ ВЗАИМОДЕЙСТВИЯ

Описание *взаимодействия* процессов может выполнить с разной степенью детализации. На верхнем (концептуальном) уровне мы изображаем процессы в виде «черных ящиков» и специфицировать только обмен сообщениями между пулами. В этом случае, состав работ в каждом процессе не отображается, а последовательность обмена сообщениями не устанавливается. Пример (см. Рисунок 11-1) изображает схему взаимодействия заказчика и клиента. Нас пока не интересуют детали выполне-

ния процессов или порядок обмена сообщениями между двумя контрагентами, схема показывает некий диалог участников.

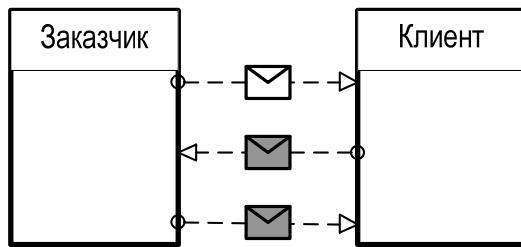


Рисунок 11-1. Межорганизационное взаимодействие, закрытый процесс

На следующем уровне детализации изображаются *открытые* процессы. В этом случае, на схеме в одном из пулов изображаются операции, отвечающие за отправку и получение сообщений, а второй пул при этом может оставаться «черным ящиком» (см. Рисунок 11-2). Применительно к нашему примеру, операции, которые выполняет заказчик, изображаются на схеме, в то время как действия клиента по-прежнему не специфицируется. Использование схем открытых процессов оправдано по нескольким причинам. Во-первых, диаграмма процесса не перегружена излишней информацией о том, как выполняется процесс внутри компании. Во-вторых, процессы зачастую составляют интеллектуальную собственность организации, и, разумеется, возникает операция защитить эту информацию.

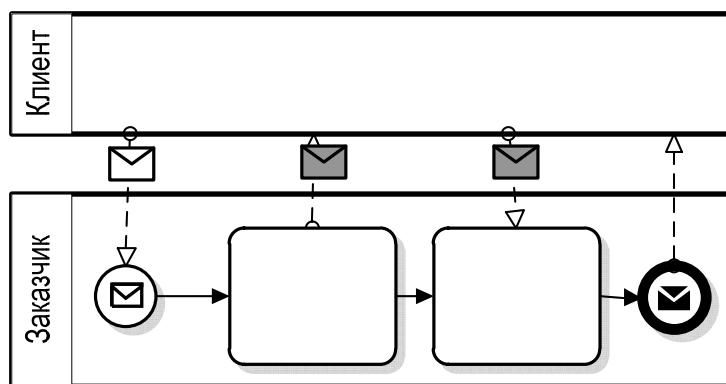


Рисунок 11-2. Межорганизационное взаимодействие, открытый процесс

На уровне «белых ящиков» действия обоих участников взаимодействия специфицируются детально. Рассмотрим моделирование открытых процессов для обоих участников взаимодействия (см. Рисунок 11-3). Здесь представлены последовательности выполнения операций, выполняемые заказчиком и

клиентом. Внутренние операции обоих участников, не связанные с межорганизационным взаимодействием, остаются за рамками диаграмм взаимодействия.

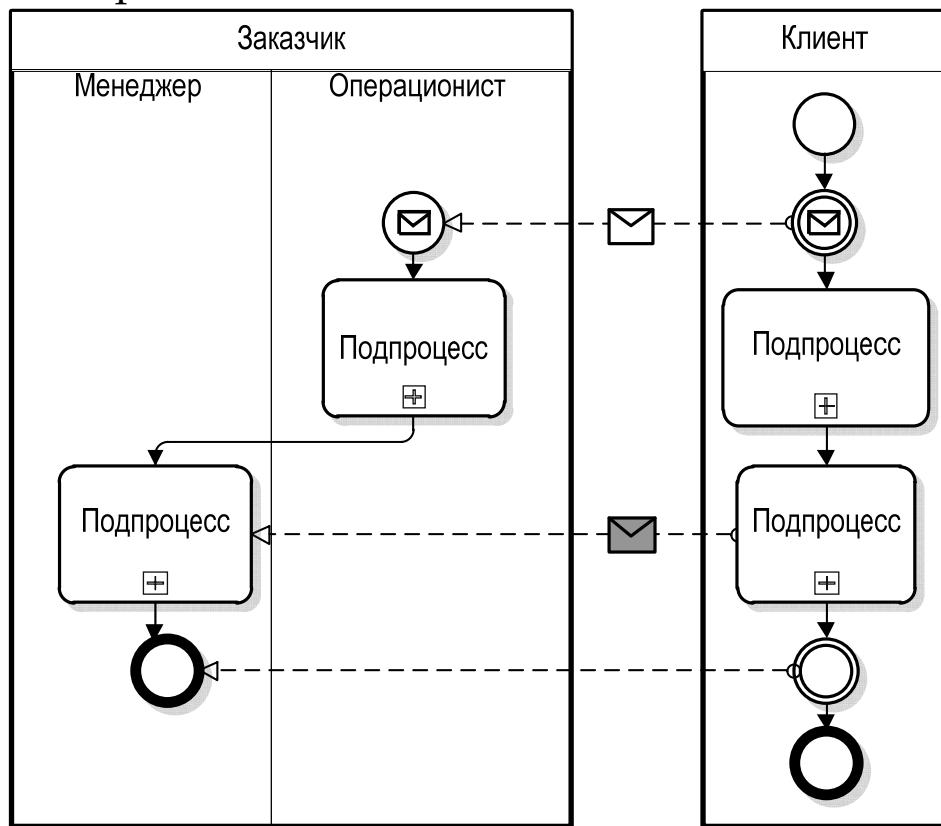


Рисунок 11-3. Детализация межорганизационного взаимодействия

11.2. МНОЖЕСТВЕННЫЕ УЧАСТНИКИ ВЗАИМОДЕЙСТВИЯ

В рассмотренном примере мы использовали сущность «Клиент», понятно, что компания одновременно может взаимодействовать не с одним клиентом, а со многими, точно так же в процессе могут участвовать несколько «Поставщиков» или «Производителей». Что бы отобразить на схеме множественного участника, знак Пул может иметь специальный маркер множественный участник. Пример (см. Рисунок 12-2) изображает взаимодействие одного заказчика (показан обычным пулом) со множеством клиентов (показаны мульти-пулами).

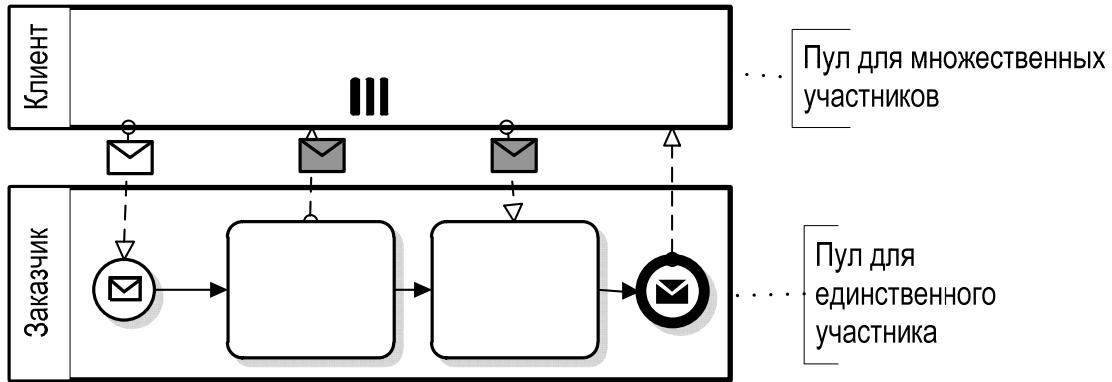


Рисунок 11-4. Мульти-пул

Графический элемент мульти-пул имеет специальные атрибуты, которые позволяют указать число множественных участников взаимодействия.

Таблица 11-1. Число участников взаимодействия

Атрибут	Пояснение
minimum	Целое число, определяет максимально допустимое число участников взаимодействия
maximum	Целое число, определяет минимально допустимое число участников взаимодействия
numParticipants	Целое число, определяет участников взаимодействия

11.3. ПАТТЕРНЫ МЕЖОРГАНИЗАЦИОННОГО ВЗАИМОДЕЙСТВИЯ

Ранее мы рассматривали приемы разработки диаграмм оркестровки. Тем не менее, существуют некоторые различия между оркестровкой процесса и описанием его взаимодействия с другими процессами. В первом случае, на схеме описывается событийная последовательность выполнения операций в процессе, управляемом из единого центра. Во втором случае на схеме изображается логическая цепочка обмена сообщениями между участниками, управляемыми по-отдельности.

Типовые конструкции (паттерны), которые рассматриваются в данном разделе, предназначены для моделирования простейших (элементарных) типов межпроцессного взаимодействия. В дальнейшем можно описать хореографию процесса, комбинируя их между собой. Кроме того, типовые конст-

рукции используются так же для верификации языков исполнения бизнес-процессов. Предложим следующую классификацию паттернов межпроцессного взаимодействия:

- По количеству вовлеченных участников.

Двустороннее взаимодействие включает всего двух участников, в то время как многостороннее - нескольких (более двух).

- По количеству переданных сообщений.

В зависимости от количества адресатов, выделяют одиночные и множественные межпроцессные взаимодействия;

Для графического изображения паттернов будем использовать нотацию BPMN. Поскольку средств BPMN не достаточно для исчерпывающего описания паттернов, будем использовать так же текстовые спецификации для каждого из них.

11.3.1. ОТПРАВКА СООБЩЕНИЯ

Паттерн описывает ситуацию односторонней передачи сообщения от отправителя к адресату. Существуют различные разновидности данного паттерна, в зависимости от момента выбора адресата сообщения: во время выполнения экземпляра процесса либо на диаграмме хореографии.

Пример (см. Рисунок 11-5) иллюстрирует использование паттерна отправки сообщения. Телефонный провайдер уведомляет заказчика о необходимости оплаты задолженности. Графический элемент пул используется на диаграмме для отображения участников взаимодействия.

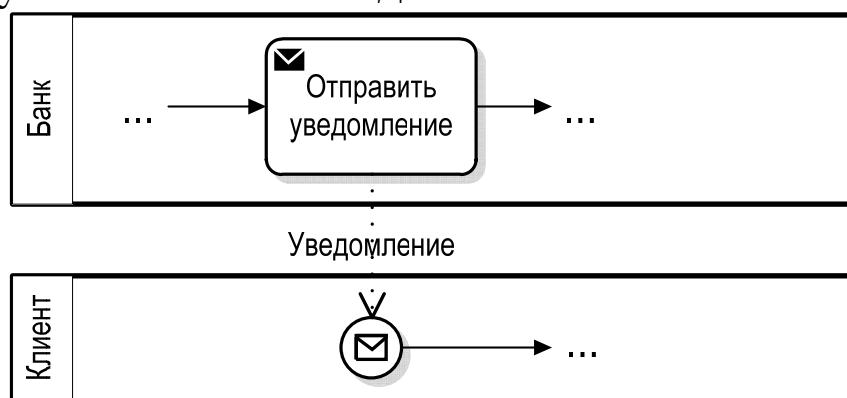


Рисунок 11-5. Паттерн отправка сообщения

11.3.2. ПОЛУЧЕНИЕ СООБЩЕНИЯ

Паттерн описывает ситуацию односторонней передачи сообщения от отправителя к получателю, но теперь уже с позиции адресата. Различают два способа использования паттерна в зависимости от наличия или отсутствия буфера сообщений у адресата. В первом случае накапливается очередь сообщений, которые будут обработаны получателем по мере возможности. Во втором случае, сообщение отменяется, если по какой-либо причине адресат не может него обработать моментально. Пример (см. Рисунок 11-6) иллюстрирует использование паттерна получения сообщения адресатом. Отправленное сообщение инициирует новый процесс, в рамках которого проверяются операции, произведенные по счету.

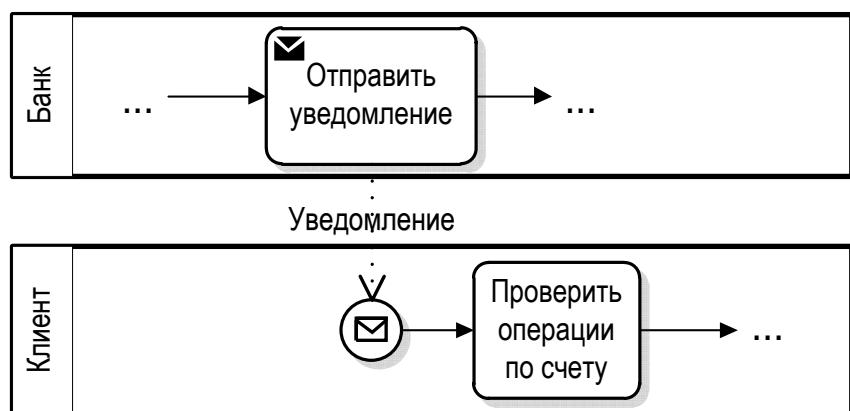


Рисунок 11-6. Паттерн получение сообщения

11.3.3. ОТПРАВКА / ПОЛУЧЕНИЕ СООБЩЕНИЯ

Паттерн описывает ситуацию, когда один адресант отправляет сообщение другому адресату, а обратно получает от него ответ. Оба сообщения пересылаются между одними и теми же участниками. В случае, если в процессе подразумевается использовать параллельный обмен сообщениями между различными участниками одного типа, то тогда необходимо связать между собой соответствующие пары запрос/ответ.

Рисунок 11-7 иллюстрирует описанный выше паттерн. Отдел закупок посылает запрос поставщику, а в ответ получает коммерческое предложение. В случае, если поставщиков несколько, то ответ должен приходить в тот экземпляр процесса,

сколько, то ответ должен приходить в тот экземпляр процесса, в котором покупатель сформировал соответствующий запрос. Для этого, сообщения с запросом и ответом должны содержать один и тот же уникальный ключ для идентификации нужного экземпляра процесса.

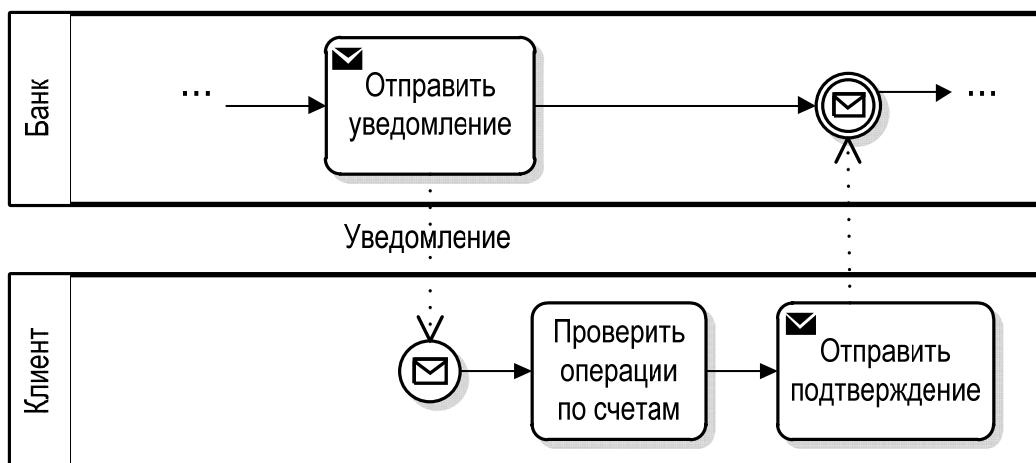


Рисунок 11-7. Паттерн отправка и получение сообщения

11.3.4. ОЧЕРЕДЬ ВХОДЯЩИХ СООБЩЕНИЙ

Паттерн описывает ситуацию, когда адресат ожидает несколько входящих сообщений от различных адресантов, но обрабатывать будет только то сообщение, которое пришло первым, по принципу FIFO - «первый пришел, первый ушел». Данная ситуация достаточно распространена при моделировании бизнес-процессов. Способ обработки очереди может зависеть от типа сообщения, от отправителя и т.д. Полученные сообщения в очереди могут быть либо сохранены для последующей обработки либо аннулированы. Паттерн очереди сообщений не покрывает описание этого паттерна (см. Рисунок 5.25).

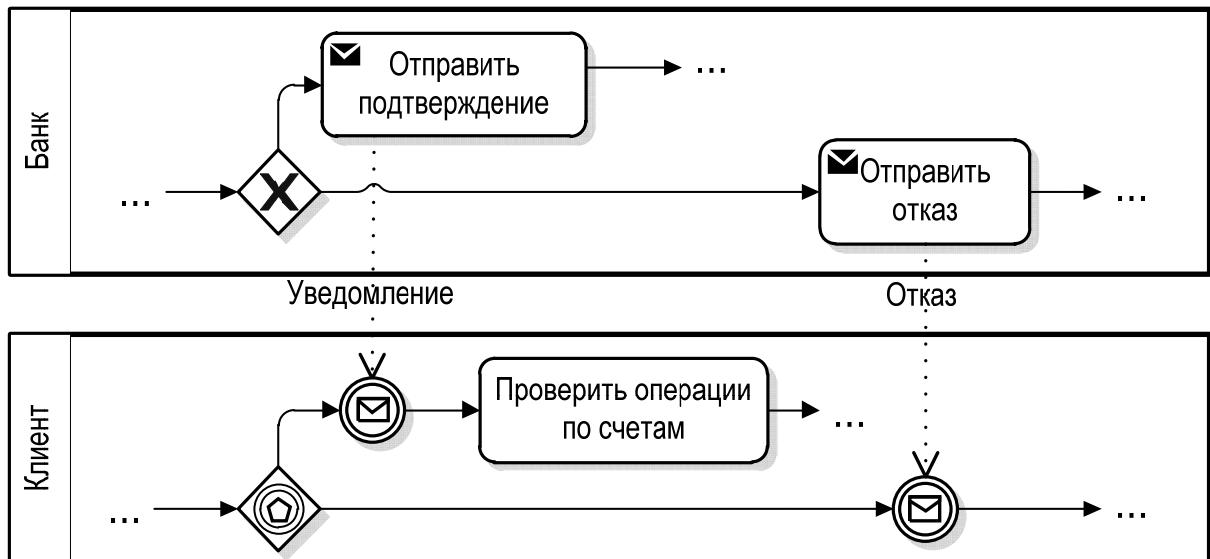


Рисунок 11-8. Очередь входящих сообщений

11.3.5. ВЕЕРНАЯ РАССЫЛКА СООБЩЕНИЙ

Участник может разослать сообщение одновременно нескольким адресатам. Список получателей может быть определен как в схеме хореографии процесса, так и непосредственно в момент рассылки сообщения, т.е. вычисляется в момент выполнении экземпляра процесса. Пример (см. Рисунок 11-9) иллюстрирует веерную рассылку. За четыре недели до начала парламентских выборов происходит рассылка уведомлений жителям, зарегистрированным в автоматизированной избирательной системе.

В нотации BPMN, данная конструкция может быть описана с помощью графического элемента - операции с пометкой маркера параллельных множественных экземпляров. Жители, которые представлены на схеме в виде пула с маркером множественности, получают сообщение и могут его обработать.

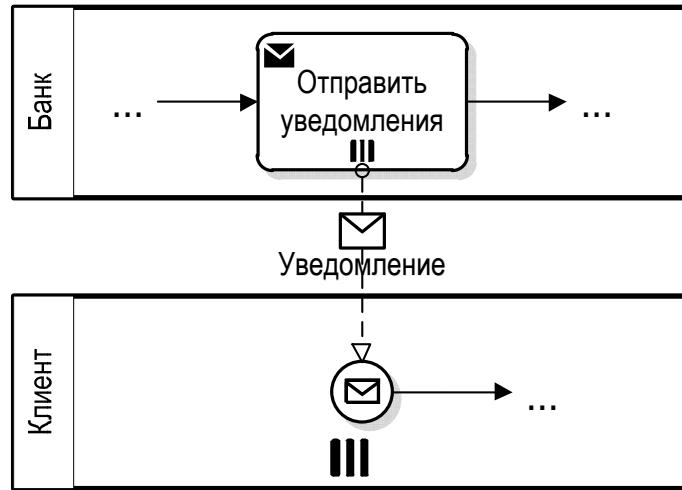


Рисунок 11-9. Веерная рассылка сообщений

11.3.6. СПИСОК ПОЛУЧЕННЫХ СООБЩЕНИЙ

Паттерн моделирует ситуацию, при которой адресат получает одновременно сообщения от разных отправителей. Каждый участник может отправить не более одного сообщения одному адресату.

Обычно, адресат не знает заранее количество сообщений, которые он получит. В связи с этим необходимо задать условие, при достижении которого прием сообщений завершится: максимально допустимое количество запросов, временное ограничение получения.

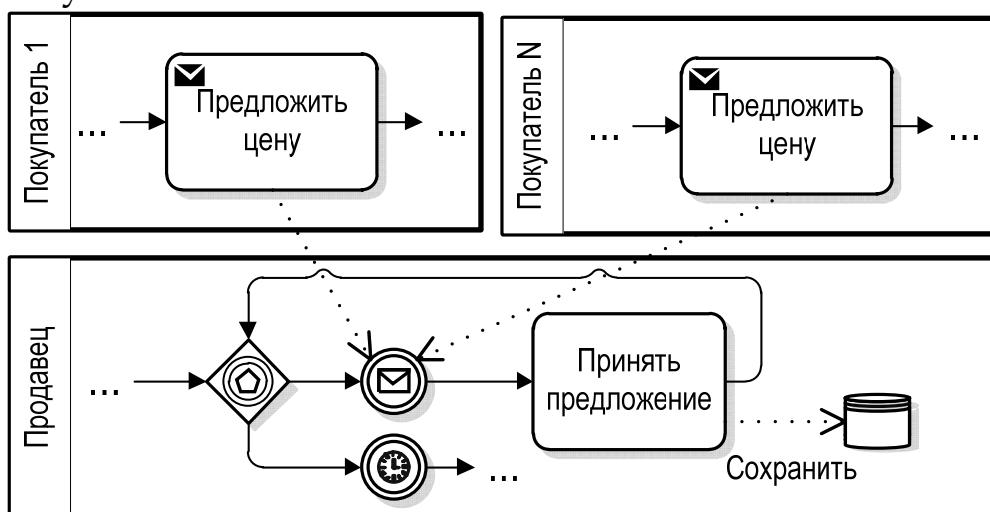


Рисунок 11-10. Список полученных сообщений

Для иллюстрации паттерна представим процесс аукционных продаж (Рисунок 11-10). Продавец выставляет на торги предложение. В течение заранее определенного срока, любой

из покупателей может сделать предложение, которое сохраняется в списке. После завершения аукциона выбирается наиболее подходящее предложение.

11.3.7. РАССЫЛКА СООБЩЕНИЙ И ОБРАБОТКА ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Паттерн моделирует ситуацию, при которой адресант одновременно рассыпает сообщение нескольким адресатам и ожидает от них ответа. Обычно, сроки получения результатов рассылки ограничиваются заранее необходимым количеством сообщений, либо временными рамками.

Приведем пример для иллюстрации паттерна (см. Рисунок 11-11). Туристическое агентство ищет лучшее предложение среди различных авиакомпаний. Для этого агентство рассыпает авиакомпаниям запрос и те в ответ могут прислать свое коммерческое предложение.

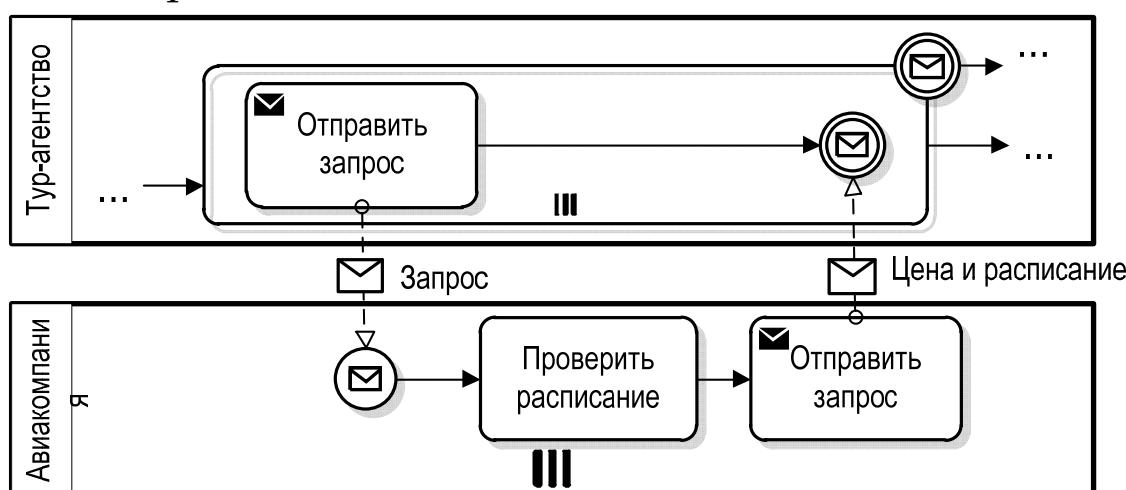


Рисунок 11-11. Рассылка сообщений и обработка результатов

11.3.8. МНОЖЕСТВЕННЫЙ ОТВЕТ

Паттерн моделирует ситуацию, при которой один участник отправляет другому запрос, а в ответ получает несколько сообщений. Главный вопрос в данной ситуации заключается в том, каким образом адресант узнает о количестве сообщений, которые он получит в ответ в результате своего запроса. Можно предложить несколько различных сценариев решения данной операции. Во-первых, каждый ответ может содержать в себе

информацию о том, ожидаются ли другие сообщения. Вторых, последнее в цепочке ответов сообщение может иметь характерный признак. Наконец, для адресанта можно определить временной промежуток, в течение которого он будет ожидать ответа от адресата своего запроса.

11.3.9. НЕГАРАНТИРОВАННЫЙ ОТВЕТ НА ЗАПРОС

Паттерн иллюстрирует ситуацию, при которой участник отправляет запрос адресату и ожидает его ответа. В случае, если адресат не ответил вовремя, адресант отправляет этот же запрос другому адресату, ожидая ответа от него. Ситуация повторяется до тех пор, пока адресант не получит необходимого ему ответа. В случае, если адресанту поступили «просроченные» ответы, он вправе решать как поступить с этими сообщениями - либо обработать их, либо аннулировать.

Пример (см. Рисунок 11-12), иллюстрирует паттерн *негарантированный ответ на запрос*. Сотрудник делегирует выполнение операции подчиненному и ожидает подтверждения. Если тот не отвечает вовремя, операция делегируется другому сотруднику и так далее, пока не будет назначен новый исполнитель задания.

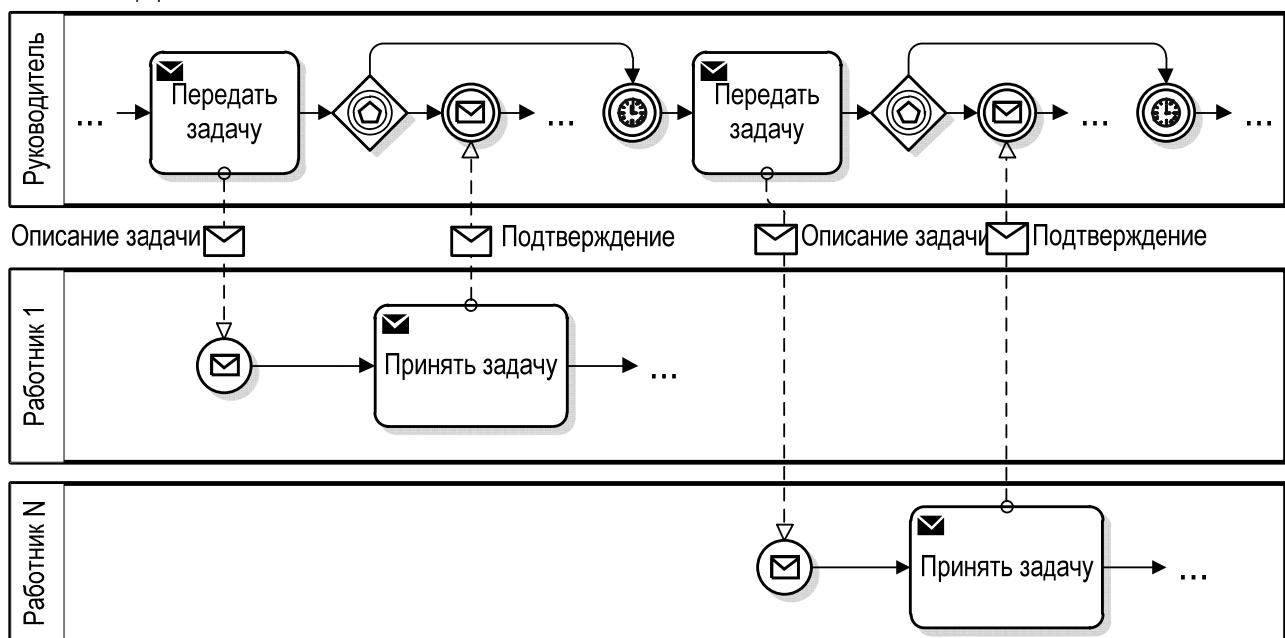


Рисунок 11-12. Негарантированный ответ на запрос

11.3.10. ШИРОКОВЕЩАТЕЛЬНАЯ РАССЫЛКА

Паттерн моделирует ситуацию, при которой участник посыпает сигнал всем доступным агентам. В зависимости от условий операции, в качестве принимающей стороны может выступать один, группа или все множество агентов.

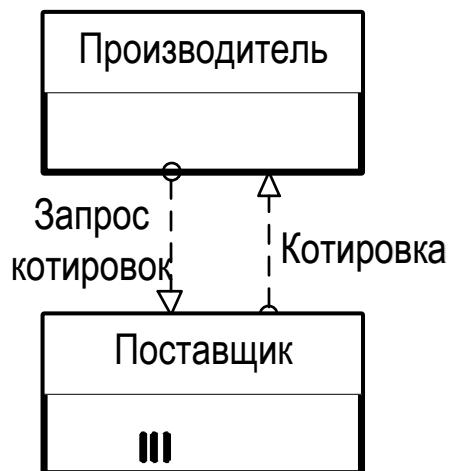


Рисунок 11-13. Широковещательная рассылка

11.3.11. ЗАПРОС С ПЕРЕНАПРАВЛЕНИЕМ

Возможность совершения запроса с перенаправлением достаточно важна в сервис-ориентированной среде, где реестр сервисов может динамически изменяться, поэтому в момент моделирования реальный адрес м.б. не известен. Например, данный паттерн может использоваться в случае, если участник А отправляет сообщение участнику В, в котором содержится просьба продолжить диалог с третьим участником С. Теперь участник В может взаимодействовать с участником С, без посредничества участника А.

Для иллюстрации рассмотрим ситуацию, которую моделирует данный паттерн (см. Рисунок 11-14). Заказчик покупает несколько книг через интернет магазин. Книжный магазин перенаправляет запрос покупателя адрес платежной системы для оплаты заказа. Это означает, что покупатель совершает покупку с использованием сервиса, о котором не подозревал в начале.

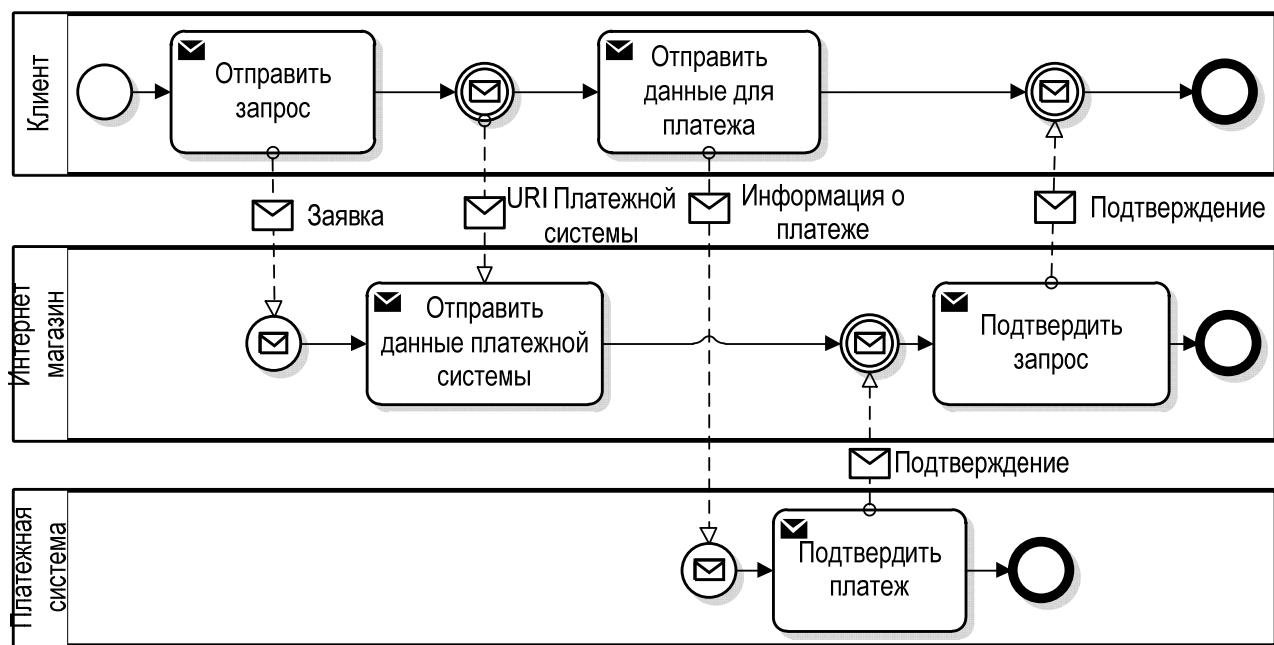


Рисунок 11-14. Запрос с перенаправлением

12. СХЕМЫ ДИАЛОГОВ

Схемы диалогов используются для моделирования межпроцессного взаимодействия на концептуальном уровне, без детализации способов передачи сообщений. Данный вид диаграмм позволяет отобразить участников, между которыми происходит взаимодействие и тематику передаваемых сообщений и, таким образом, отвечает на вопрос кто с кем общается и по какому вопросу.

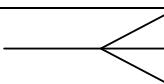
Поскольку термин диалог является достаточно общим, BPMN предлагает более точное описание: **диалог** - это набор логически связанных информационных сообщений, посвященные одной теме. Например, организация покупатель посыпает запрос на поставку товара. С одним поставщиком у неё обсуждается одновременно несколько контрактов, так что каждая сделка может рассматриваться как отдельный диалог. Диалоги м.б. иерархически вложенными. Например, один контакт может включать обсуждения: номенклатуры товаров, способов оплаты и доставки, сроки поставки и т.д. Каждая тема может рассматриваться как отдельный диалог. Тематика обмена определяется автором. Технически каждой теме соответствует отдельный корреляционный ключ (см. п. 6.5.6). Таким образом, автор диаграммы может произвольно определить тематику диалогов по своему желанию.

Схема диалогов является высокоуровневым способом отображения взаимодействия участников. Для этого, логически связанные сообщения, изображаемые на схеме взаимодействия независимо друг от друга, на схеме диалога группируются по темам. Принцип группировки может основываться, например, на работе с одним бизнес объектом, например, «Заказом». В этом случае, диалог объединяет отдельные сообщения: «Отправить Заказ», «Подтвердить Заказ», «Ответ на Заказ». Элементы Схемы диалогов

Схемы диалогов состоят из ограниченного набора графических элементов. Таблица 12-1 показывает основные элементы. Ключевым элементом такой диаграммы является **диалог**. Спе-

цификация определяет четыре типа диалогов:

Таблица 12-1. Элементы схемы диалогов.

Знак	Название	Комментарий
	Обычный диалог	группа логически связанных потоков сообщений
	Комплексный диалог	Диалог, который м.б. иерархически декомпозирован на вложенные суб-диалоги
	Глобальный диалог	Вызов глобально определенного, повторно используемого диалога
	Комплексный глобальный диалог	Вызов глобально определенного, повторно используемого суб-диалога
	Пул	Пул (Pool) представляет собой графическое изображение Участника (Participant) Взаимодействия. Ограничивает процесс, выполняемый одним из участников.
	Мульти-пул	Процесс, у которого есть много участников, для каждого исполняется свой экземпляр процесса
	Линия коммуникации	Связывает участников, осуществляющих диалог
	Мульти линия коммуникации	Поток диалога связывает с мультипулом, которому соответствует несколько участников

Для моделирования участников используются **пулы**. Один пул может отображать множественных участников. Например, в диалоге покупателя с поставщиком, в роли поставщика могут выступать много компаний. Если число участников не известно на этапе моделирования, то используется элемент мульти-пул. Тема обсуждения моделируется с помощью элемента диалог, она м.б. составной – состоять из нескольких субтем, в этом случае **суб-диалог** помечается маркером «+», он может быть декомпозирован на отдельном листе диаграммы, где информационное взаимодействие участников диалога отображается на более детализированном уровне. Для соединения диалогов с участниками используются графический элемент

«линия коммуникации». Если линия направлена на мульти-пул, то используется **мульти-линия**.

Рисунок 12-1 иллюстрирует схему взаимодействия (А) и соответствующую ей схему диалога (Б).

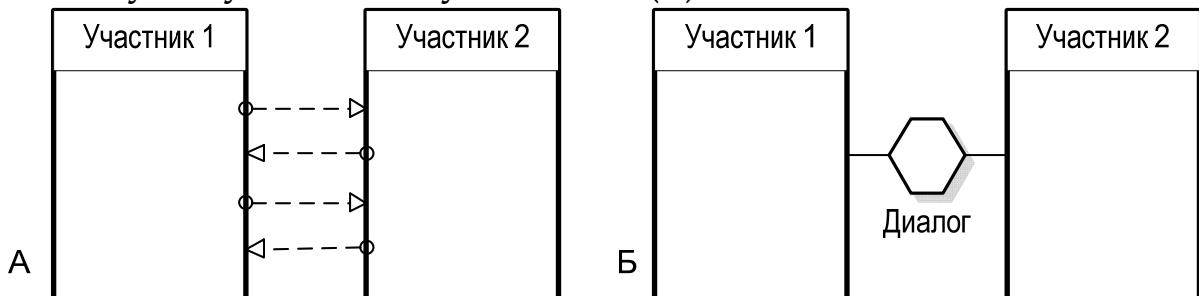


Рисунок 12-1. Схемы взаимодействия (А) и диалога (Б)

12.1. МУЛЬТИ-ПУЛЫ

Проиллюстрируем использование схемы диалогов на примере аукционных торгов, изображающей обмен сообщениями между Поставщиками, Посредником и Заказчиками (см. Рисунок 12-2). В роли поставщиков и Заказчиков могут выступать сразу несколько компаний, поэтому они моделируются мульти-пулами. С Поставщиком ведется диалог о «Закупке товара», а с Заказчиком о «Продаже товара».



Рисунок 12-2. Диалог участников, мульти-пул

Диалог м.б. повторно используемым, в этом случае, для его обозначения используется элемент **глобальный диалог**. Рассмотрим пример (см. Рисунок 12-3), пусть банк взаимодействует с агентством кредитных историй. Диалог является повторно используемым, поэтому он моделируется с помощью глобального диалога. В этом случае линия коммуникации содержит надпись, указывающую участников диалога.

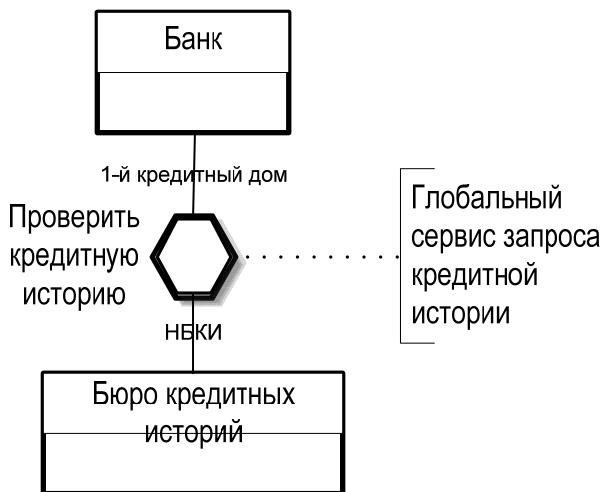


Рисунок 12-3. Глобальный диалог

12.2. КОМПЛЕКСНЫЕ ДИАЛОГИ

Диаграмма диалогов не показывает мелких деталей, поэтому она позволяет охватить комплексное взаимодействие многих участников со многими. Пример (см. Рисунок 12-4) показывает взаимодействие Покупателя и Поставщика. Схема верхнего уровня (Рисунок А) показывает один комплексный диалог о поставке. Следующая схема (Рисунок Б) показывает декомпозицию комплексного диалога на два простых, посвященных условиям оплаты и условиям поставки.

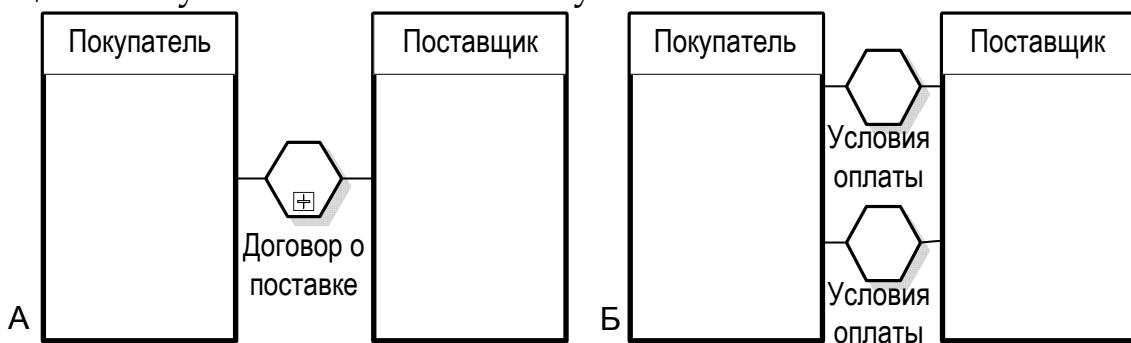


Рисунок 12-4. Комплексный диалог

Следующая схема (см. Рисунок 12-5А) показывает декомпозицию на отдельные сообщения, а затем (рисунок Б) соответствующую диаграмму публичного процесса.

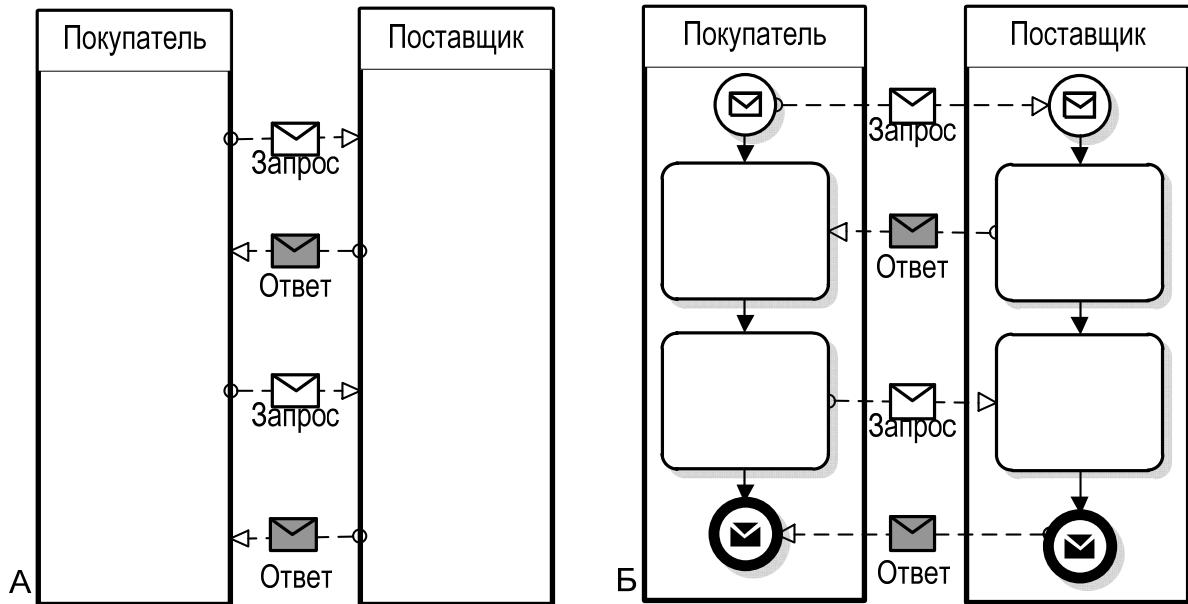


Рисунок 12-5. Декомпозиция комплексного диалога

Диаграмма публичного взаимодействия может появиться на одном листе модели вместе с диаграммой взаимодействия (см. Рисунок 12-6).

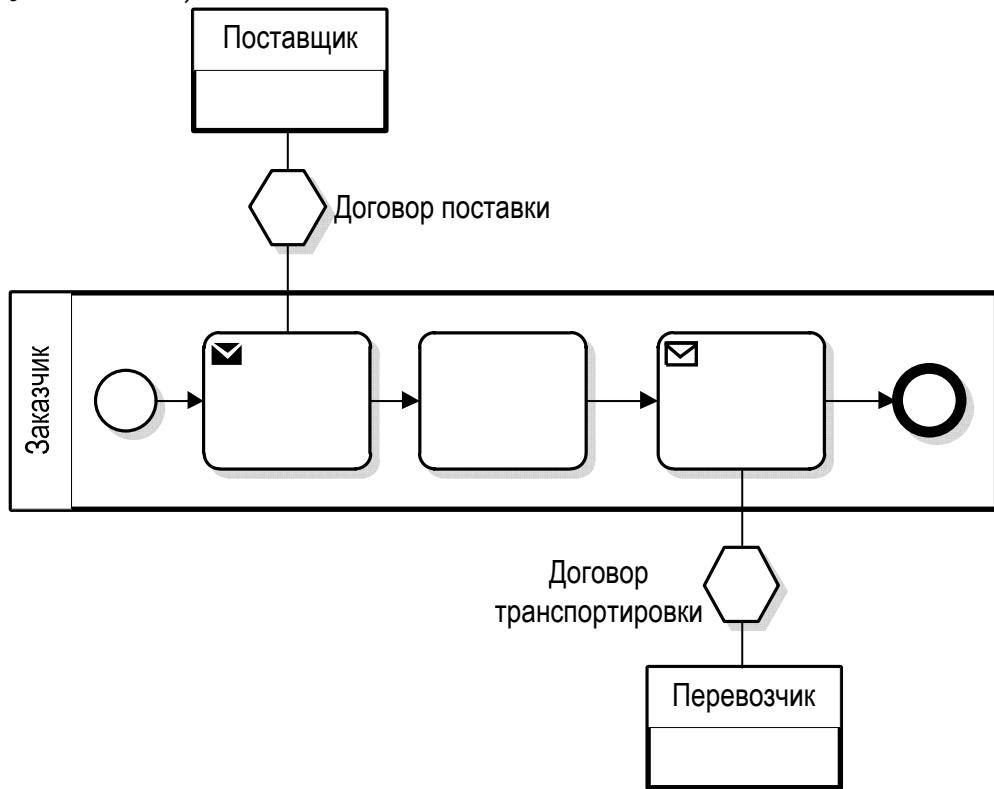


Рисунок 12-6. Комплексная диаграмма взаимодействия

13. СХЕМЫ ХОРЕОГРАФИИ

Схема Хореографии предназначена для формализации порядка межорганизационного взаимодействия, основанного на обмене информационными сообщениями между сторонами, участвующими во взаимодействии. Она описывает формальный договор между источником и получателем информационной посылки.

Поскольку техническим каналом обмена сообщениями обычно является протокол WebService, типовые сценарии взаимодействия описываются с помощью Web Services Message Exchange Patterns (см. <http://www.w3.org/2002/ws/cg/2/07/meps.html>). Что бы обмен не носил хаотический характер, разработаны типовые последовательности информационного обмена, которые можно разделить на односторонние, когда запрос не предполагает определенного ответа и двухсторонние, с ответом. Не следует сводить эти последовательности только к тривиальным случаям, поскольку они описывают так же информирование о произошедших ошибках, сбоях и т.д.

13.1. ГРАФИЧЕСКИЕ ЭЛЕМЕНТЫ ХОРЕОГРАФИИ

Схемы хореографии используют следующие графические элементы нотации BPMN:

- Задачи хореографии;
- Потоки сообщений;
- Переходы управления;
- Логические элементы;
- События (подмножество элементов оркестровки).

Основной элемент, используемый на диаграмме хореографии, есть задача хореографии. Она показывает взаимодействие адресанта (инициатора) и адресата (получателя), имеет индивидуальное имя. Следует обратить внимание, что адресант и адресат различаются цветом соответствующих полей, а не положением этих полей внутри задачи хореографии. Схема

может показывать конверты сообщений, содержимое посылки на этой схеме не моделируется. Инициирующее диалог сообщение и ответное различаются цветом. Пример (см. Рисунок 13-1А) показывает двусторонний обмен типа «запрос-ответ», там же (рисунок Б и В) задача хореографии изображена средствами диаграммы оркестровки и взаимодействия.

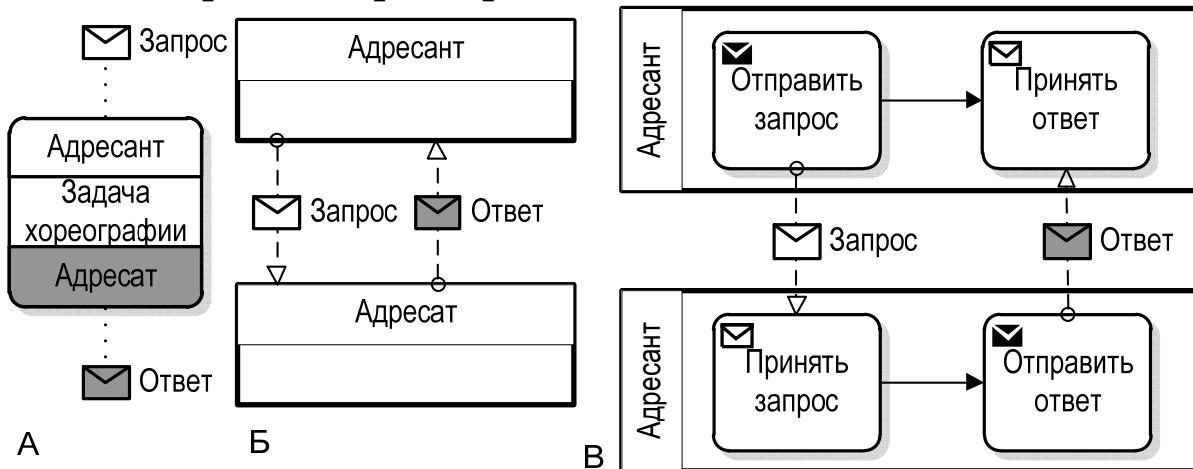


Рисунок 13-1. Обмен сообщениями, оркестровка, хореография

Допускаются т.н. односторонние диалоги, когда инициатор отправляет сообщение, но не ожидает на него ответа. Пример (см. Рисунок 13-2А) показывает одностороннюю задачу хореографии, там же (рис. Б) – соответствующая схема оркестровки.

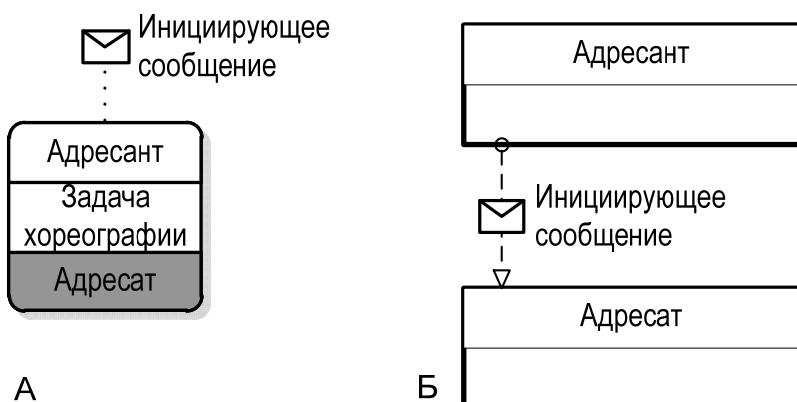


Рисунок 13-2. Однонаправленная задача хореографии

В одной задаче хореографии могут участвовать сразу несколько адресатов. Представим себе, что заказчик отправляет запрос на поставку товара двум поставщикам. С обоими он одновременно ведет одинаковые переговоры. Пример (см. Рисунок 13-3А) показывает задачу хореографии, имеющую двух ад-

ресатов, там же (рис. Б) – соответствующая схема оркестровки.

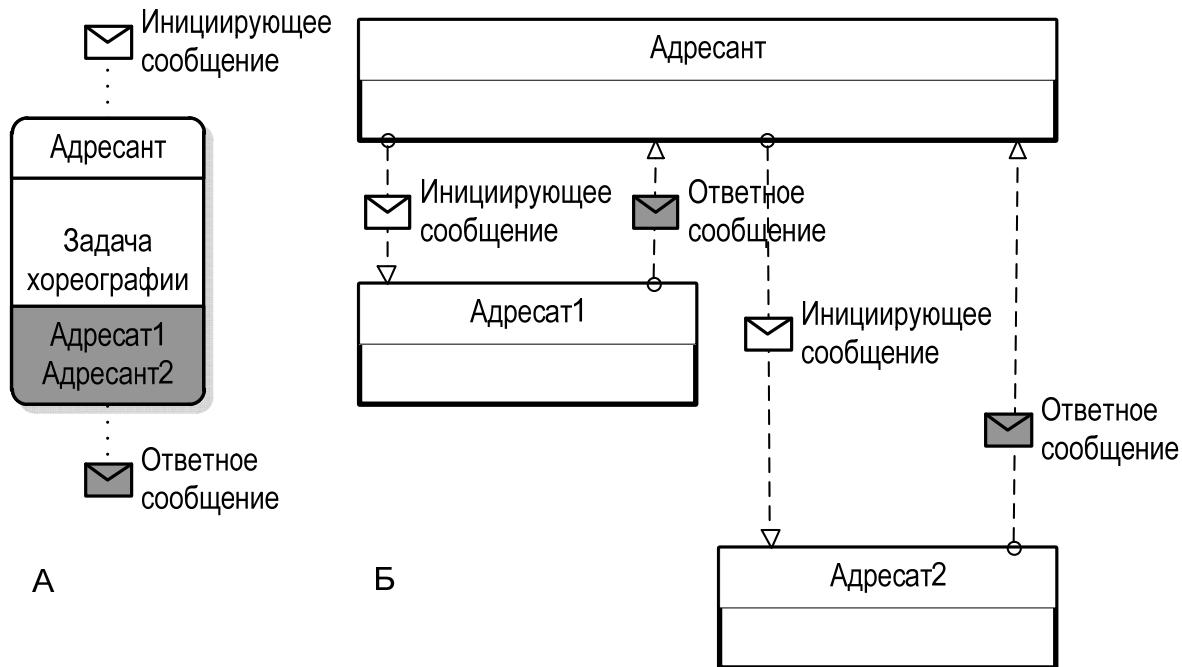


Рисунок 13-3. Задача хореографии с двумя адресатами

Возможна ситуация, когда число партнеров не было известно на этапе моделирования, но станет известно в момент исполнения. Например, запрос направляется всем поставщикам, их список зависит от типа требуемой поставки. В этом случае принято говорить о множественных участниках, на диаграмме оркестровки они отображаются с помощью мультипулов. Пример (см. Рисунок 13-4А) иллюстрирует веерную рассылку запроса многим участникам, там же (рис. Б) – соответствующая схема оркестровки.

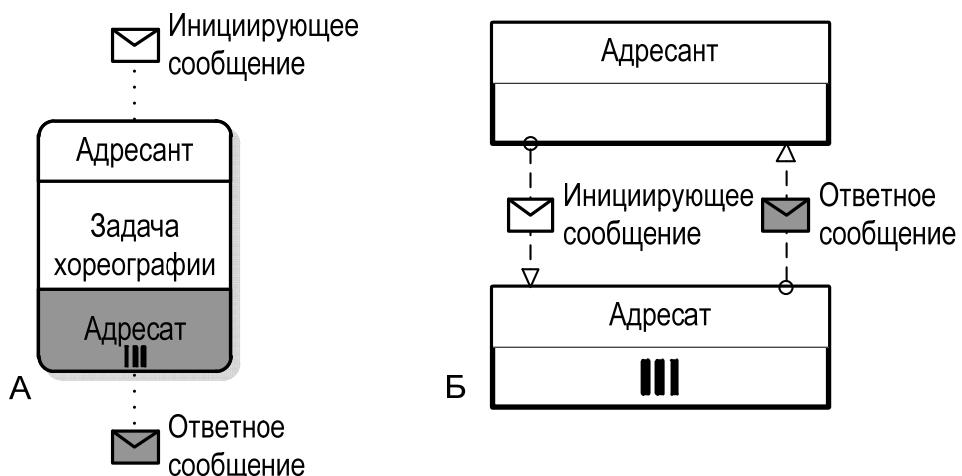


Рисунок 13-4. Веерная рассылка запроса многим участникам

Задача хореографии м.б. повторяющейся, представим себе, что инициатор циклически повторяет рассылку. Пример (см. Рисунок 13-5А) показывает циклическую рассылку, там же (рис. Б) – соответствующая схема оркестровки.

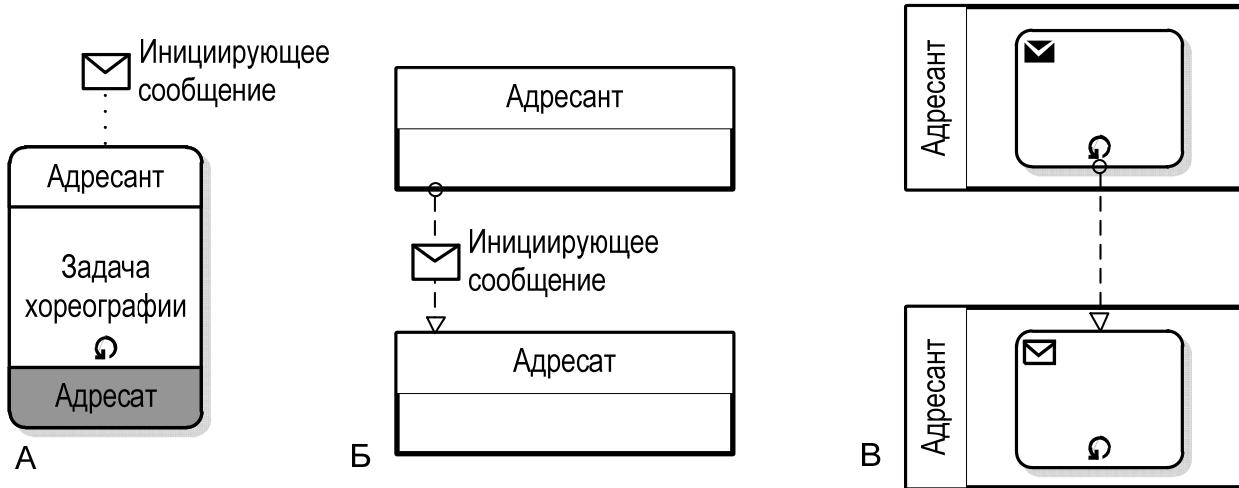


Рисунок 13-5. Циклическая рассылка запроса

Представим себе ситуацию, когда два участника обмениваются большим числом посылок. Пример (см. Рисунок 13-6) показывает диаграмму взаимодействия двух участников, которая содержит несколько пар диалогов.

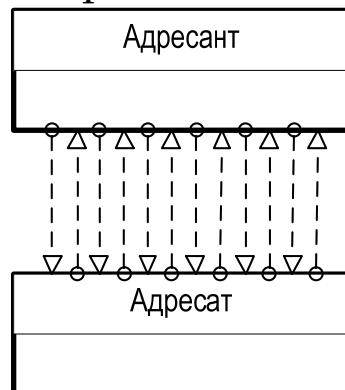


Рисунок 13-6. Диаграмму взаимодействия, несколько пар диалогов

Что бы обмен не выглядел хаотическим, можно заменить этот обмен на комплексную задачу, которая включала бы в себя несколько диалогов

Пример (см. Рисунок 13-7А) показывает комплексную задачу хореографии, в которой первый участник осуществляет два диалога со вторым и третьим участниками. Там же (рис. Б) показана развернутая задача хореографии, она показывает два поддиалога, схема взаимодействия (рис В) показывает те же

диалоги.

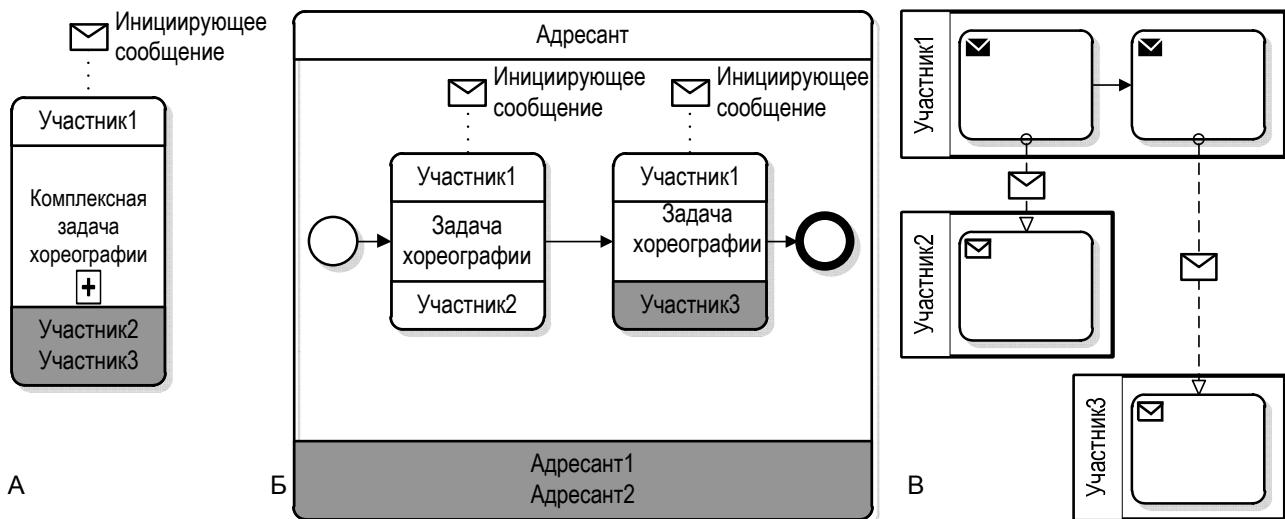


Рисунок 13-7. Комплексная задача хореографии

Комплексная задача хореографии может быть повторно используемой. Как и в схемах оркестровки для вызова внешнего подпроцесса предусмотрена глобальная задача. По внешнему виду она отличается рамкой, изображаемой более толстой линией. Глобальная задача м.б. свернутой, адресую схему на отдельной странице, или развернутой, показывая нужные диалоги на том же листе. Пример (см. Рисунок 13-8А) показывает свернутую глобальную задачу, там же (см. рис. Б) показана развернутая задача хореографии.

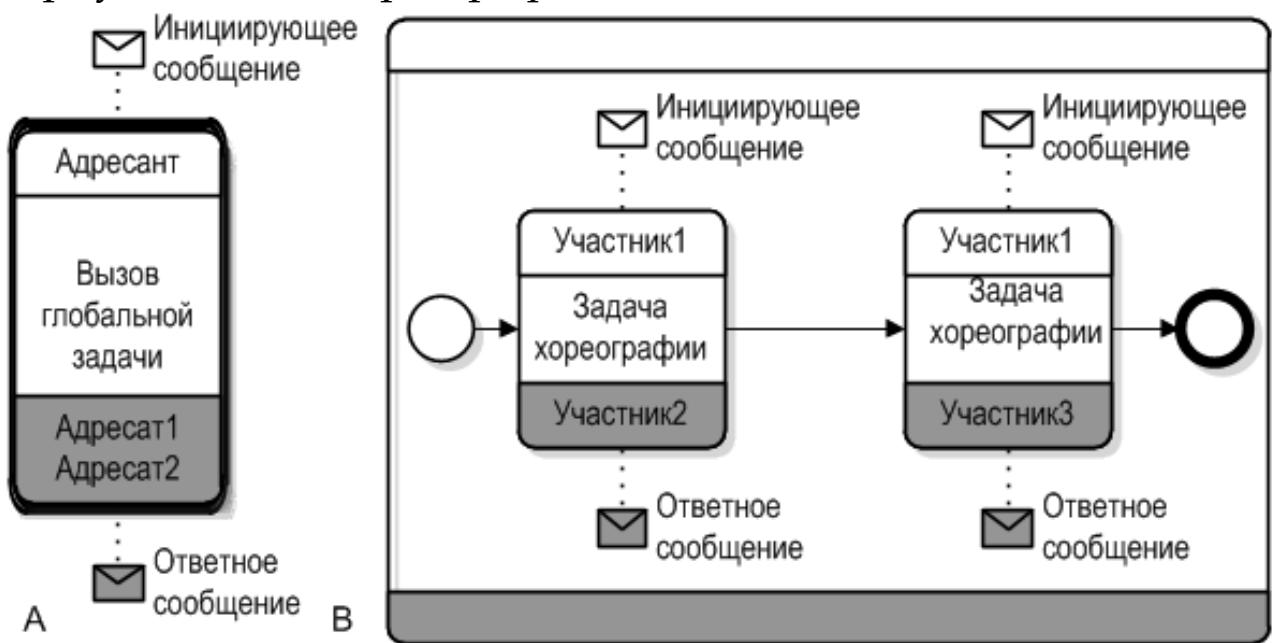


Рисунок 13-8. Глобальная задача хореографии

13.2. ИСПОЛЬЗОВАНИЕ СОБЫТИЙ В ХОРЕОГРАФИИ

Диаграммы Хореографии могут использовать ограниченный набор событий, что позволяет описать реакцию процесса на внешние изменения.

13.2.1. НАЧАЛЬНЫЕ СОБЫТИЯ

Таблица 13-1. Обзор начальных событий показывает краткий обзор используемых в хореографии начальных событий.

Таблица 13-1. Обзор начальных событий

Название	Описание	Знак
Простое	Не типизированное событие, показывает старт процесса хореографии.	(○)
Сообщение	Нет	
Таймер	Создает экземпляр процесса в назначенное время, в цикле, по расписанию. Все участники должны иметь общее представление о времени.	(🕒)
Эскалация	Нет	
Условное	Проверяется внутренне условие одного из участников.	(☰)
Ошибка	Нет	
Компенсация	Нет	
Сигнал	Создает экземпляр процесса после получения сигнала.	(△)
Составное	Для создания ЭП необходимо и достаточно совершения только одного из перечисленных событий. Разрешены только сообщения и таймер	(◇)
Параллельное составное	Нет	(+)

13.2.2. ЗАВЕРШАЮЩИЕ СОБЫТИЯ

Таблица 13-2. Обзор завершающих событий содержит краткий обзор используемых в хореографии завершающих событий.

Таблица 13-2. Обзор завершающих событий

Название	Описание	Знак
Простое	Другие участники не знают про завершение потока.	
Сообщение, Ошибка, Эскалация, Отмена, Компенсация, Сигнал		Нет
Прекращение	Не приводит к остановке исполнения других участников. Остальные участники не смогут узнать о завершении потока иначе, чем через полученные сообщения.	

13.3. ПРОМЕЖУТОЧНЫЕ СОБЫТИЯ

Таблица 13-3 содержит краткий обзор используемых в хореографии промежуточных событий.

Таблица 13-3. Обработчики промежуточных событий.

Название	Описание	Знак
Простое	Обработчик отсутствует. Используется для отображения статуса исполнения.	
Сообщение	Нет	
Таймер	При установке абсолютных дат все участники знают календарное время. Только участники задали хореографии непосредственно предшествовавшей установке таймера знают значение задержки срабатывания.	
Ошибка, Эскалация, Компенсация		Нет
Условное	Работает как элемент задержки, ожидает, когда условие станет истинно. Данные, входящие в проверяемое условие д.б. видимы всем участникам	
Ссылка	Связывает две разные части процесса и подразумевает неявную передачу потока управления от одной операции - другой.	
Сигнал	Используется для широковещательной связи между участниками. Работает как элемент задержки, ожидающий поступления сигнала.	
Составное	Обработка одного события из множества или генерация всех определенных событий.	

Таблица 13-4 содержит краткий обзор используемых в хореографии промежуточных прикрепляемых событий.

Таблица 13-4. Обработчики событий, прикрепляемые к задаче хореографии

Наименование	Описание	Знак
Сообщение	Срабатывает при получении сообщения.	
Таймер	Срабатывает один раз в определенный момент времени либо несколько раз по расписанию.	
Эскалация	Изменяет уровень принятия решения, может быть только генерирующими.	
Ошибка	Перехватывает именованные и безымянные ошибки, которые могут произойти во время выполнения операции. Прерывает выполнение задачи, к которой прикреплен обработчик.	
Отмена	Срабатывает, если задача хореографии заканчивается завершающим событием отмена.	
Компенсация	Срабатывает, если задача хореографии заканчивается завершающим событием отмена.	
Условное	Проверяет истинность условия. Условие может быть любым, в том числе составным и содержать в себе другие обработчики событий, которые не могут быть визуализированы на схеме процесса.	
Сигнал	Принимает широковещательное оповещение. Отличается от Ошибки более широким диапазоном использования, например, позволяет уведомить об успешном завершении операции и т.д.	
Составное взаимоисключающее	Перехватывает сложные составные события, при этом достаточно, чтобы произошло хотя бы одно событие из списка.	

13.4. ИСПОЛЬЗОВАНИЕ ЛОГИЧЕСКИХ ОПЕРАТОРОВ В ХОРЕОГРАФИИ

Диаграммы Хореографии могут использовать логические операторы, что позволяет описать не только последовательность обменов, но так же логику взаимодействия. Следует учитывать, что участники оценивают ход процесса только по полученным сообщениям. Представим ситуацию, когда покупатель и продавец провели переговоры о цене, пришли к соглашению, покупатель перевел деньги в банк. Понятно, что продавец отпустит товар только после подтверждения от банка о поступлении денег.

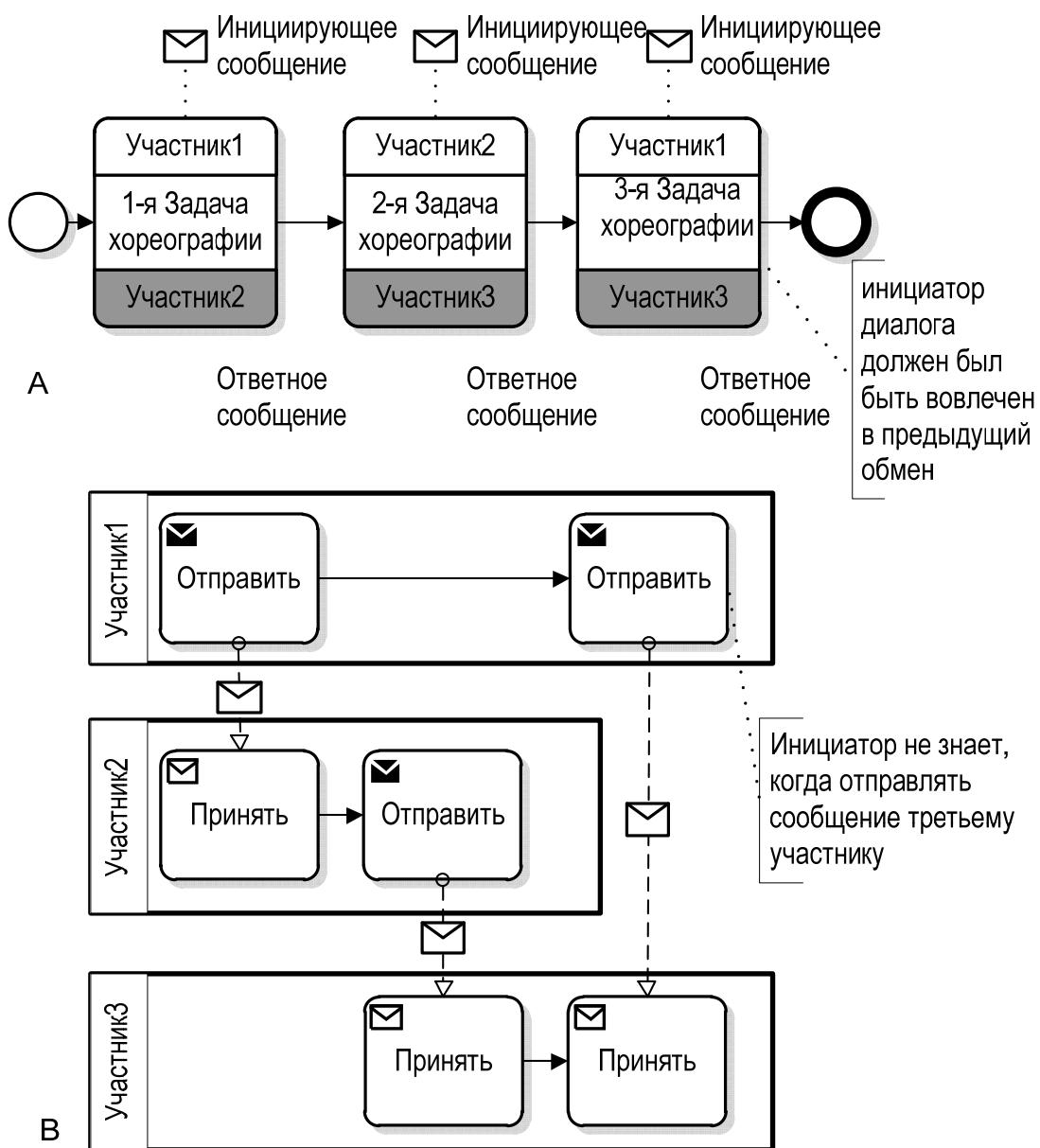


Рисунок 13-9. Пример ошибочной схемы хореографии

Этот пример иллюстрирует важное правило диаграмм хореографии: инициатор диалога должен был быть вовлечен в предыдущий обмен, иначе он может знать его результат (это правило не относится к первой задаче в цепочке). Пример (см. Рисунок 13-9) показывает ошибочную схему хореографии, участник 1 не вовлечен в предыдущий диалог и не знает, когда отправлять сообщение.

Использование логических операторов на схеме хореографии вызывает некоторые затруднения. Это связано с отсутствием в хореографии центрального потока управления и единого объекта управления. Например, логический оператор «ИЛИ управляемый данными» предполагает анализ некоторого информационного объекта. Прежде всего, следует иметь в виду, что анализируемые данные должны поступить в процесс до момента принятия решения в виде информационного сообщения. Но не обязательно именно в предыдущей задаче.

Пример (см. Рисунок 13-10) изображает схему хореографии, которая состоит из трех задач и логического оператора «исключающее ИЛИ управляемое данными», осуществляющего ветвление в зависимости от значения данных процесса. Заказчик запрашивает коммерческое предложение от двух поставщиков, получает их ответы, после чего изучает их (эта задача отсутствует на схеме хореографии, так как она является внутренней для компании и моделируется на схеме потоков работ) и делает выбор в пользу одного из них. Там же (рис. Б) – соответствующая схема оркестровки.

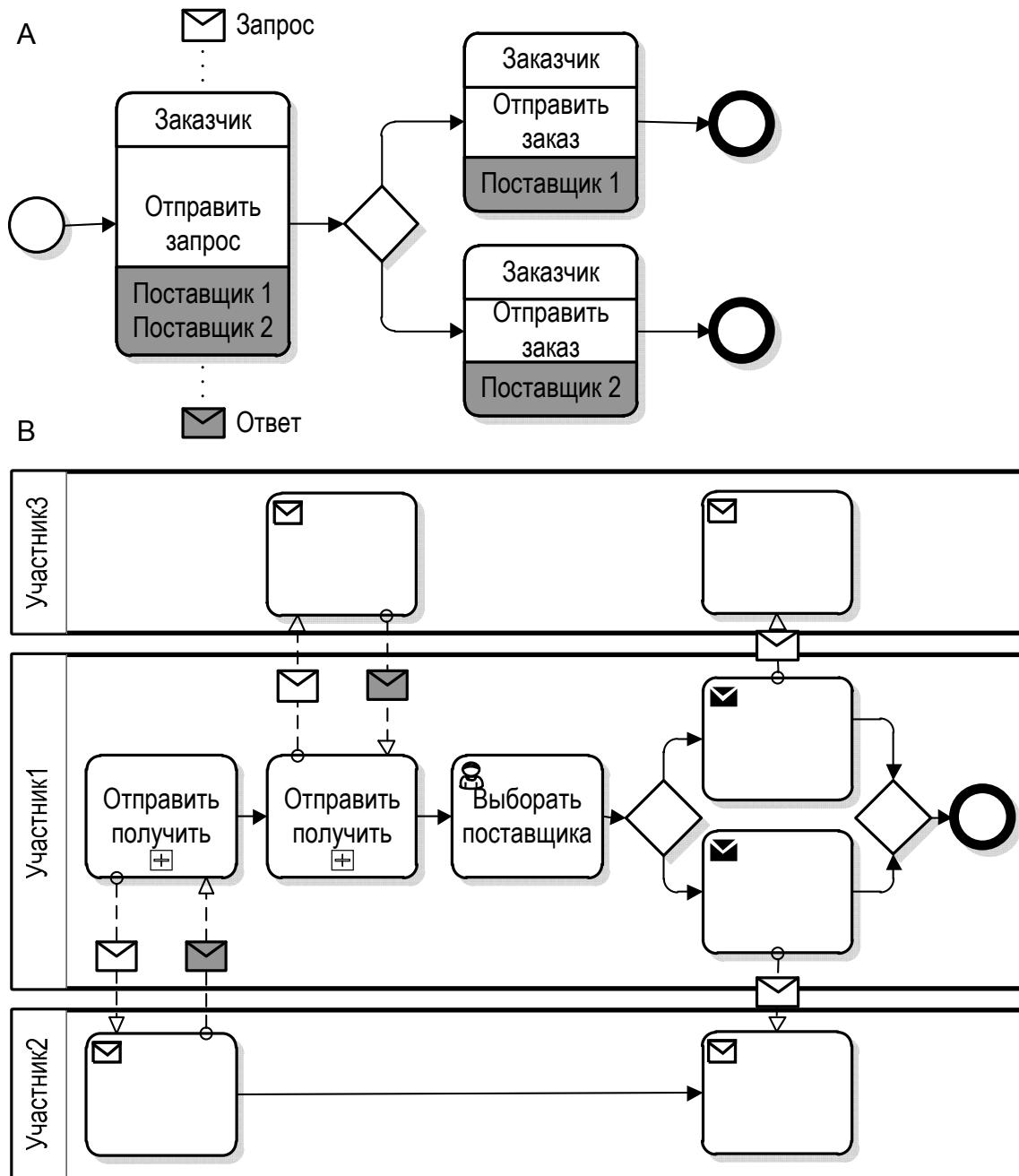


Рисунок 13-10. Использование логических операторов на схеме хореографии

Логика принимаемого решения должна управляться одним участником взаимодействия, он принимает решение и извещает о нем остальных посредством сообщения. Это означает, что все участники хореографии должны иметь единое представление о данных, на основании которых принимается решение. Для этого служат общие словари, дающие единое толкование бизнес терминов. После того как соответствующее сообщение, содержащее данные отправлено, участник уже не может в одностороннем порядке изменить их значения. Таким

образом, участник, который оповещает остальных о решении, считается лицом принимающим решение.

Реализуем теперь тот же сценарий, используя на схеме хореографии логический оператор «событийное исключающее ИЛИ» (см. Рисунок 13-11А), там же (см. рис Б) изображена соответствующая диаграмма взаимодействия.

Первая задача хореографии рассыпает запросы поставщикам и принимает от них ответ. Приняв решение, заказчик направляет сообщение тому поставщику, который выиграл конкурс. Событийный логический оператор работает как «исключающее ИЛИ», пропуская оповещение только одному поставщику. Что бы второй поставщик не застрял в ожидании подтверждения, которое ему уже никогда не поступит, используется событие таймер, которое устанавливает максимальный допустимый период ожидания. Т.о., если оповещение не поступит в течение нормативного времени, срабатывает таймер и процесс поставщика завершается.

Данный пример наглядно демонстрирует альтернативное использование логических операторов «исключающее ИЛИ», основанных на данных, либо на событии. В случае использования логического оператора, основанного на данных, на схему хореографии необходимо добавить задачу информирования поставщиков о принятом решении. В случае использования логического оператора, основанного на событии, размещать такую задачу на хореографии уже не обязательно.

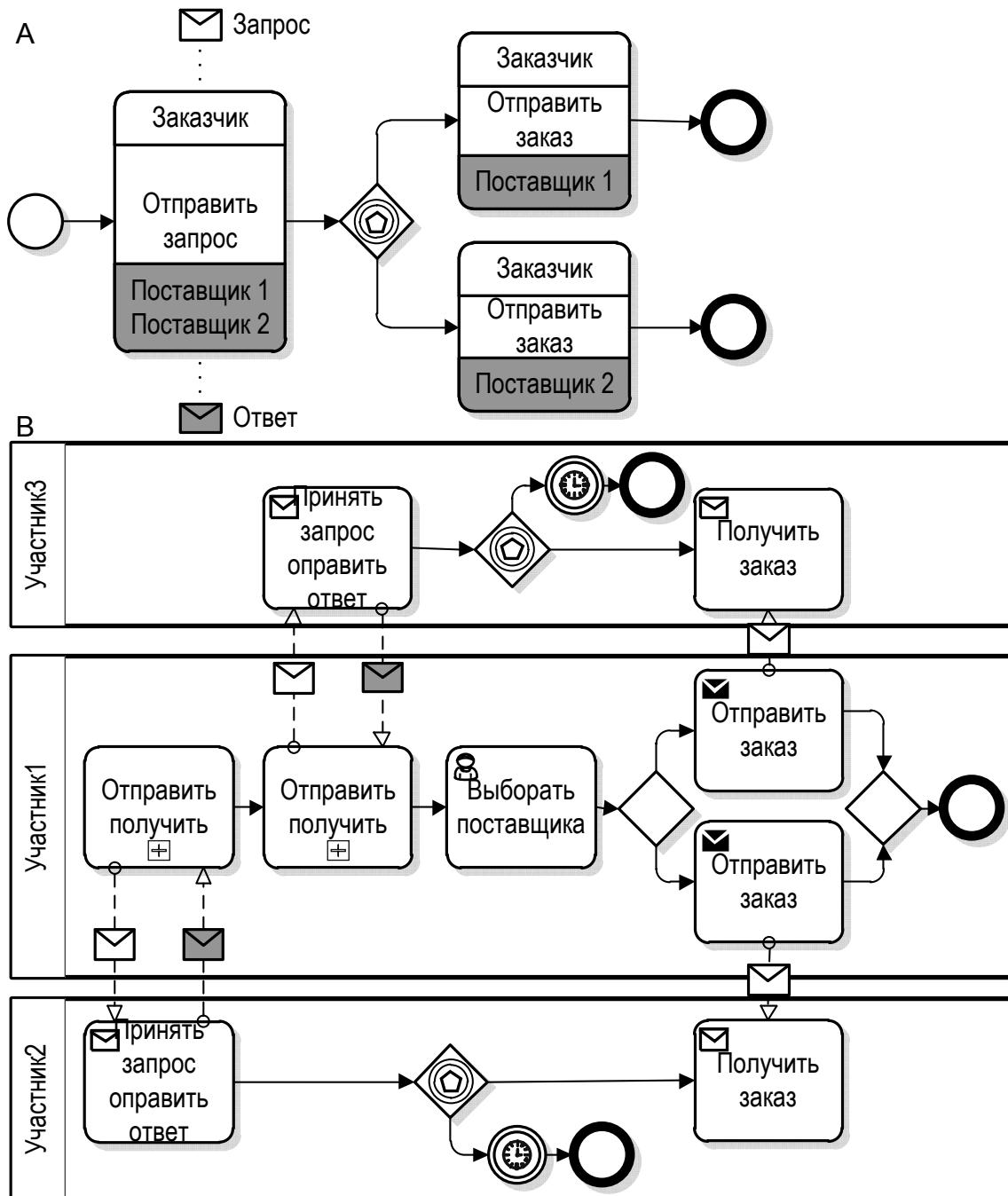


Рисунок 13-11. Использование логических операторов на схеме хореографии

Схема хореографии так же допускает использование логических операторов «И». В обоих случаях, инициатор хореографии должны участвовать в выполнении задач, в которой принимается решение. Только после того, как он будет уверены в успешном завершении этой задачи, он может инициировать передачу сообщения в логический оператор.

Пример (см. Рисунок 13-12) показывает использование опе-

ратора «И», поставщик одновременно рассыпает заказы сразу обоим поставщикам.

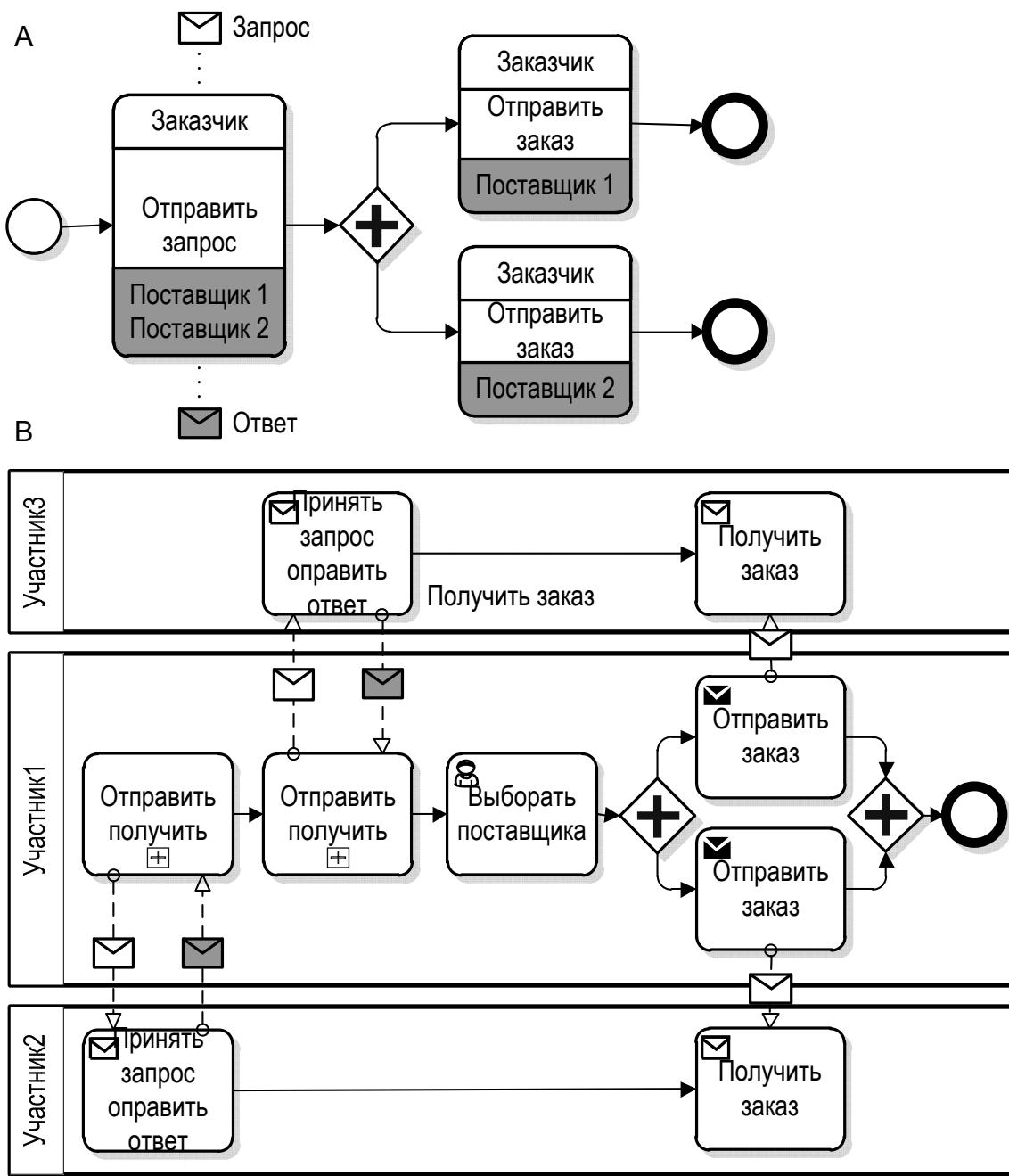


Рисунок 13-12. Использование логического оператора «И» на схеме хреографии

14. ЗАКЛЮЧЕНИЕ

В данной работе была сделана попытка дать систематическое описание нотации BPMN 2.0, описать синтаксис использования элементов, показать семантику их выполнения. Изложение снабжено многочисленными примерами. Хочется надеяться, что удалось достичнуть поставленной цели.

Вместе с тем хочется отметить, часто методологию моделирования подменяют нотацией, а это не одно и то же. Методология определяет подходы, приемы и методы моделированию, а нотация помогает зафиксировать результаты. Напрашивается аналогия с музыкой, где музыкальная композиция и нотная грамота суть разные вещи.

В качестве примера рассмотрим нотацию IDEF0, неразрывно связанную с методологией SADT. Будет ошибкой называть нотацию IDEF0 методологией или сказать, что SADT есть нотация. Аналитики, которые при анализе процесса не опираются на соответствующую методологию, рисуют допустить ошибки моделирования. Вспомним, модели IDEF0 принято называть функциональными, однако многие используют в нотацию для описания процессов. Не может ли оказаться, что некоторые модели, которые мы используем для описания бизнес-процессов в отрыве от методологии, по сути, являются функциональными? Наше утверждение: не нотация, а методология определяет, является ли модель функциональной или процессной.

В рамках данной работы не предполагалось описать методологию моделирования бизнес-процессов. Это отдельная тема, требующая серьезной проработки. Вместе с тем, без подобной методологии труд бизнес-аналитика будет напоминать ремесло, тогда как нам нужен инженерный подход, позволяющий превратить ремесло в технологию. Хочется надеяться, что следующая работа сможет стать руководящим документом бизнес-аналитиков, вооружит их методологией моделирования бизнес-процессов.

ПЕРЕКРЕСТНЫЕ ССЫЛ

Абстрактная (нетипизированная) операция.....	59
Автоматизированные процессы.....	18
Автоматическая операция	57
Автоматические процессы.....	18
Альтернатива.....	187
бизнес-процесс.....	4
Версионность исполняемой	29
Вложенный подпроцесс	68
Внутриорганизационные процессы	17
Вспомогательные (обеспечивающие) бизнес-процессы	14
Входные и выходные данные процесса.....	167
Вызывающая операция	72
Генерирующее событие.....	100
Группа операций	66
Действие	5
Диаграмма закрытого процесса	44
Диаграмма оркестровки	44
Диаграмма открытого процесса	45
Диаграмма приватного взаимодействия	45
Диаграмма публичного взаимодействия	47
Диалог	109
Дискриминатор	195
Жизненный цикл объекта данных.....	162
Завершающее событие-прекращение	122
Завершающее событие-сообщение	124
Интегрированная модель бизнес-процесса	27
Интерактивная операция	56
Исполнение с синхронизацией (число экземпляров известно на этапе исполнения)	202
Исполняемая модель	46
Исполняемая модель бизнес-процесса	25
Итерация.....	199
Коллекция объектов данных	165
Конечное событие	100

Координация исполнения	205
Корреляция	114
Логический оператор	38
Логический оператор «ИЛИ» управляемый данными	87
Логический оператор «Исключающее ИЛИ» управляемый данными	90
Логический оператор комплексное условие	93
Маркер параллельного выполнения	61
Маркер подпроцесса	60
Маркер последовательного выполнения	62
Маркер потока управления процесса	21
Маркер цикла	60
Межорганизационные процессы	17
Методология структурного анализа и проектирования	24
Многократное повторение	198
Многопоточные операции	22
Множественное слияние	194
Множественный выбор	190
Направленные ассоциации	39
Начальное событие	100 116
Независимые события	101
Неправильные ассоциации	38
Непрерывающие события	102
Неструктурированные паттерны	193
Обрабатывающее событие	100
Объектом управления	20
Объекты данных	39
Оперативное управление	8
Операция	5 38
Операция бизнес-правило	57
Операция для случая (ad-hoc)	63
Операция компенсация	63
Операция сценарий	57
Основные процессы	14
Отложенный выбор	203
Параллельное исполнение	186
Параллельное исполнение без синхронизации	201
Параллельное исполнение с синхронизацией (число экземпляров известно на этапе моделирования)	202
Параллельные потоки управления процесса	21

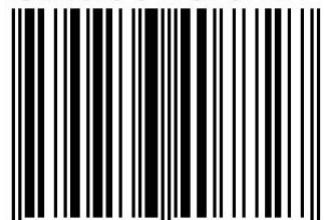
Повторно используемый (внешний) подпроцесс	70
Подпроцесс.....	67
Подпроцесс для конкретного случая (Ad-Hoc)	73
Последовательное исполнение	185
Поток обработки исключительной ситуации	137
Поток управления процесса	21
Потоки сообщений	107
Потоки управления.....	38
Прекращение исполнения задания.....	208
Прекращение исполнения многопоточной операции	210
Прекращение исполнения процесса.....	209
Прерывающее событие	102
Приватный процесс поддерживающий публичный	50
Прикрепленное непрерывающее граничное событие-таймер	138
Прикрепленное непрерывающее обрабатывающее граничное событие...	137
Прикрепленное прерывающее граничное событие-таймер	138
Прикрепленное прерывающее обрабатывающее граничное событие.....	137
Прикрепленные события	101
Проект	4
Промежуточное событие	100
Промежуточное событие-таймер	131
Промежуточное событие-условие	132
Промежуточное составное параллельное событие	133
Промежуточное составное событие	132
Промежуточные события отправить и принять сообщения	131
Простое завершающее событие	121
Простое промежуточное событие	130
Простое слияние параллельных ветвей	188
Процесс.....	66
Рекурсия.....	200
Роль.....	172
Ручная операция	56
Семантика нотации BPMN	34
Сигнал	104
Синхронизация	187
Синхронизация без запоминания оповещения.....	211
Система базирующаяся на процессах	10
Система ориентированная на процессы	10
Системы управления бизнес-процессами.....	12

Системы управляемые моделью	11
Сквозные бизнес-процессы	14
События	38
Событие принимающее сигнал	104
Событие генерирующее сигнал	104
Событие-компенсация	127
Событие-отмена	126
Событие-ошибка	125
Событие-ссылка	134
Событие-эскалация	126
Событийный логический оператор «Исключающее ИЛИ»	118
Событийный оператор «Исключающее ИЛИ»	94 96
Событийный подпроцесс	75
События прикрепляемые к границам операций	134
Сообщения	107
Составное взаимоисключающее стартовое событие	117
Составное параллельное стартовое событие	118
Статус обработки объекта данных	164
Стратегическое управление	8
Структурированная декомпозиция работ	24
Структурированное слияние с синхронизацией	192
Схема взаимодействия	213
Схема хореографии	51
Схемы взаимодействия	50
Тактическое управление	8
Транзакционный подпроцесс	78
Функциональная модель	25
Хранилище данных	166
Чередование маршрутов	204

СПИСОК ЛИТЕРАТУРЫ:

1. OMG Business Process Model and Notation (BPMN) Version 2.0 OMG Document Number: formal/2011-01-03 Standard document URL:
<http://www.omg.org/spec/BPMN/2.0>
2. B. Silver. BPMN Method & Style Cody-Cassidy Press 2011
3. S.White D. Miers BPMN Modeling and Reference Guide Future Strategies Inc. 2008
4. M.Weske Business Process Management: Concepts Languages Architectures Springer-Verlag Berlin Heidelberg 2012
5. А.Белайчук «Главное не результат главное процесс» www.mainthing.ru
6. А.Белайчук «Учим BPMN» www.bpmntraining.ru
7. B.Silver «BPMS Watch» www.brsilver.com

ISBN 978-5-7764-0772-7



9 785776 407727