UML 2
и Унифицированный процесс

ВТОРОЕ ИЗДАНИЕ

ПРАКТИЧЕСКИЙ ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ

ДЖИМ АРЛОУ
АЙЛА НЕЙШТАДТ

СИМВОЛ

UML 2
and the Unified Process

Practical Object-Oriented
Analysis and Design

Second Edition

Jim Arlow and Ila Neustadt

Addison-Wesley

UML 2
и Унифицированный процесс

Практический объектно-ориентированный анализ и проектирование

Второе издание

Джим Арлоу и Айла Нейштадт

Символ®

Санкт-Петербург — Москва
2008

Джим Арлоу и Айла Нейштадт

UML 2 и Унифицированный процесс, 2-е издание

Практический объектно-ориентированный анализ и проектирование

Перевод Н. Шатохиной

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*А. Пальваль*
Редактор	*А. Петухов*
Корректура	*О. Макарова*
Верстка	*Д. Орлова*

Арлоу Д., Нейштадт И.
UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 624 с., ил.

ISBN-13: 978-5-93286-094-6
ISBN-10: 5-93286-094-4

Сегодня многие книги посвящены или UML, или Унифицированному процессу (Unified Process, UP), но не им обоим. Арлоу и Нейштадт заполнили этот пробел книгой, являющей собою замечательный синтез UML и UP. Здесь вы изучите методики объектно-ориентированного анализа и проектирования, синтаксис и семантику UML и соответствующие аспекты UP. Книга содержит точный и лаконичный обзор UML и UP с точки зрения ОО аналитика и проектировщика. В издании четко и понятно рассказано о практическом применении UML 2 на этапах анализа и проектирования Унифицированного процесса. Вы узнаете о роли моделирования в цикле разработки ПО, и эти знания помогут вам ответить на вопрос: как и когда использовать (или не использовать) UML, чтобы найти оптимальное решение для своего проекта. Авторы приводят множество примеров и дают рекомендации, бесценные для начинающих разработчиков моделей. Опытные ОО аналитики и проектировщики найдут в книге полезное руководство и справочник по UML 2.

ISBN-13: 978-5-93286-094-6
ISBN-10: 5-93286-094-4
ISBN 0-321-32127-8 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 01.10.2007. Формат 70x100 1/16 . Печать офсетная.
Объем 39 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Отзывы о книге

«Стандарт UML 2 группы OMG очень систематично и основательно определяет UML, но в нем не хватает описания того, как применять UML 2 в реальном проекте. Вот где пригодится «UML 2 и Унифицированный процесс», 2-е издание. В книге ясно и доходчиво рассказывается о практическом применении UML 2. Изложение сопровождается множеством примеров и рекомендаций. Книга очень полезна даже тем, кто не работает с Унифицированным процессом. «UML 2 и Унифицированный процесс», 2-е издание – обязательная книга для новичков в UML 2 и полезное руководство и справочник для опытных профессионалов».

– Роланд Лейбандгут (Roland Leibundgut),
технический директор, Zühlke Engineering Ltd.

«Авторы очень подробно описывают конструктивные элементы UML и то, как они поддерживают Унифицированный процесс. Эта книга – хорошая отправная точка для организаций и специалистов, которые переходят к UP и нуждаются в понимании того, как обеспечить визуализацию различных аспектов в соответствии с UP».

– Эрик Найбург (Eric Naiburg)
менеджер по маркетингу, Desktop Products
IBM Rational Software

«Сегодня многие книги посвящены или UML, или Унифицированному процессу (Unified Process, UP), но не им обоим. Арлоу и Нейштадт заполнили этот пробел книгой, являющей собой замечательный синтез UML и UP. Авторы предлагают богатый опыт, бесценный для начинающих разработчиков моделей и опытных OO аналитиков и проектировщиков. Логическая структура, основанная на рабочих потоках UP, и особый стиль изложения с использованием диаграмм деятельностей в начале каждой главы существенно упрощают работу с книгой. Это издание должно быть всегда под рукой и у профессионалов, и у студентов».

– Исхан Де Силва (Ishan De Silva)
разработчик программного обеспечения
Millennium Information Technologies, Шри-Ланка

«Если вы ищете книгу с рецептами, почитайте что-нибудь другое. Эта книга заставит вас думать! В ней описываются все синтаксические элементы UML, но, что более важно, она дает практический совет, как и когда использовать (или не использовать) UML. Вы научитесь думать о роли моделирования в процессе разработки. Эти знания помогут вам ответить на вопрос: как и когда использовать UML, чтобы найти оптимальное решение для своего проекта. «UML 2 и Унифицированный процесс», 2-е издание подготовит вас к успешному применению UML».

– Джос Уормер (Jos Warmer)
Ordina System Integration & Development, Нидерланды

«Авторы создали книгу, объединяющую два важных предмета, UML и Унифицированный процесс. «UML 2 и Унифицированный процесс» – превосходный справочник по UML 2. Издание рассказывает о возможностях UML и о том, как применять его в дисциплинах анализа и проектирования Унифицированного процесса. Эта книга должна быть на столе у каждого профессионала».

– Гэри Поллис (Gary Pollice)
профессор, преподаватель вычислительной техники
Вустерский политехнический институт

Благодарности

Хотелось бы поблагодарить Фабрицио Феррандина (Fabrizio Ferrandina), Вольфганга Еммериха (Wolfgang Emmerich) и наших друзей из Zühlke Engineering за то, что они сподвигли нас на создание учебного курса, который стал основой этой книги. Особая благодарность Роланду Лейбандгуту (Roland Leibundgut) из Zühlke за его комментарии к главам, посвященным прецедентам, и Джосу Уормеру (Jos Warmer) и Тому Ван Корту (Tom VanCourt) за их глубокий анализ главы по OCL. Огромное спасибо и остальным техническим редакторам: Глен Форд (Glen Ford), Бергеру Меллер-Педерсену (Birger Møller-Pedersen), Робу Петтиту (Rob Pettit), Гэри Поллису (Gary Pollice), Исхану Де Силва (Ishan De Silva) и Фреду Васкиевичу (Fred Waskiewicz). Спасибо Сью и Девиду Эпштейнам (Sue, David Epstein) за жизненно важную нетехническую поддержку в течение всего проекта. Благодарим Энди Полса (Andy Pols), поделившегося с нами своими идеями по поводу прецедентов и процесса производства программного обеспечения. Наша благодарность сотрудникам издательства Addison-Wesley Ларе Вайсонг (Lara Wysong), Мэри Лу Нор (Mary Lou Nohr) и Ким Арни Малкахи (Kim Arney Mulcahy) за их замечательную работу над текстом и нашему редактору Мэри О'Брайан (Mary O'Brien). Спасибо семейству Нейштадт (Neustadt) за их терпение и Элу Томсу (Al Toms) за огромную поддержку. И конечно же, нашим котам, Гомеру, Падди и Мег, за многие часы сна на рукописях, которые наполнили их «неописуемым качеством».

И наконец, мы должны выразить признательность «Трем амигос» – Гради Бучу (Grady Booch), Джиму Рамбо (Jim Rumbaugh) и Айвару Джекобсону (Ivar Jacobson) – за их высококлассную работу над UML и UP, которым посвящена эта книга.

Предисловие

Об этой книге

Цель этой книги – показать процесс объектно-ориентированного (ОО) анализа и проектирования с помощью Унифицированного языка моделирования (Unified Modeling Language, UML) и Унифицированного процесса (Unified Process, UP).

UML представляет собой язык визуального моделирования для ОО моделирования. UP обеспечивает каркас процесса производства программного обеспечения, указывающий, как осуществлять ОО анализ и проектирование.

О UP можно говорить много. В книге представлены только аспекты, имеющие непосредственное отношение к работе ОО аналитика/проектировщика. За подробной информацией по другим деталям UP обращайтесь к [Rumbaugh 1] и другим указанным в библиографии книгам по UP.

Здесь приведено достаточное количество информации по UML и ассоциированным с ним методикам анализа и проектирования, что обеспечивает возможность эффективно применять моделирование в реальном проекте. Согласно Стивену Меллору (Stephen Mellor) [Mellor 1], существует три способа использования UML:

- UML как эскиз – это неформальный подход к UML, при котором используется схематическое изображение диаграмм, помогающее визуализировать программную систему. Это несколько схоже с наброском идеи на обратной стороне салфетки. Эскизы не представляют практически никакой ценности кроме их исходного применения, не сохраняются и в конце концов выбрасываются. Для создания неформальных эскизов обычно используют доску или инструментальные средства рисования, такие как Visio и PowerPoint (*www.microsoft.com*).
- UML как модель – это более формальный и точный подход, при котором UML используется для подробного описания программной системы. Это как набор архитекторских планов или чертеж машины. UML-модель активно поддерживается и становится важным поставляемым артефактом проекта. Этот подход требует использования настоящего инструментального средства моделирования, такого как Rational Rose (*www.rational.com*) или MagicDraw UML (*www.magicdraw.com*).
- UML как исполняемый проект – с помощью MDA (Model Driven Architecture – архитектура, управляемая моделью) UML-модели мо-

гут использоваться как язык программирования. Создается достаточно подробная UML-модель, и система может быть скомпилирована прямо из нее. Это самое формальное и точное применение UML и, по нашему мнению, это будущее разработки программного обеспечения. При таком подходе необходим UML-инструмент, поддерживающий MDA, такой как ArcStyler (*www.arcstyler.com*). Рассмотрение MDA выходит за рамки обсуждения этой книги, хотя мы касаемся его вкратце в разделе 1.4.

Основное внимание в книге сосредоточено на UML как модели. Представленные технические приемы также подойдут и для использования UML как исполняемого проекта. Изучив UML как модель, вы свободно сможете использовать UML как эскиз в случае необходимости.

Мы попытались сделать наше представление UML и UP максимально простым и доступным.

Условные обозначения

Чтобы упростить ориентирование по книге, каждая глава снабжена планом в форме диаграммы деятельностей UML. Эти диаграммы показывают деятельности чтения и порядок прочтения всех разделов. Диаграммы деятельности подробно рассматриваются в главе 14, а сейчас рис. 1 поможет разобраться с диаграммами планов глав.

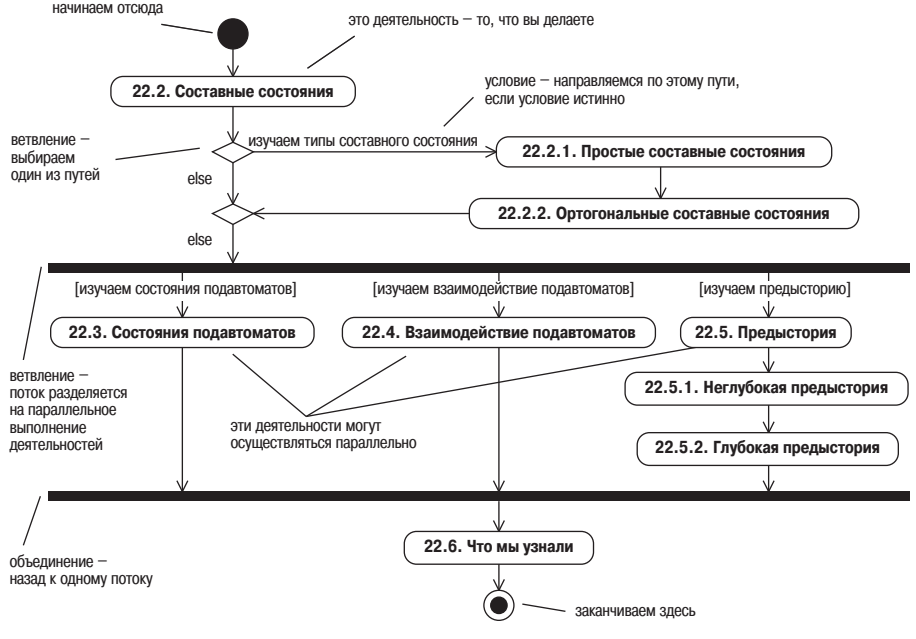


Рис. 1.

Большинство диаграмм в данной книге – это UML-диаграммы. Поясняющий текст на диаграммах не является частью синтаксиса UML.

> Важная информация оформлена в виде UML-пиктограммы примечания, т. е. заключена в прямоугольник с загнутым уголком.

В книге используются разные шрифты:

Этот шрифт применяется для элементов моделирования UML.

`Этот шрифт – для кода.`

Для кого эта книга

Мы видим следующих возможных читателей данной книги.

- Вы аналитик или проектировщик, которому необходимо научиться проводить ОО анализ и проектирование.
- Вы аналитик или проектировщик, которому необходимо научиться проводить ОО анализ и проектирование в рамках Унифицированного процесса.
- Вы студент, изучающий курс UML в университете.
- Вы разработчик программного обеспечения, которому необходима справочная информация по UML.
- Вы разработчик программного обеспечения, слушающий учебный курс по UML, и эта книга – ваш учебник.

Компания Clear View Training предлагает 4-дневный учебный курс по UML, основанный на данной книге. Этот курс читается по всей Европе нашим партнером, компанией Zuhlke Engineering (*www.zuhlke.com*), и доступен для лицензирования. Образовательные учреждения, использующие данную книгу как учебник, могут воспользоваться нашим учебным курсом бесплатно. Более подробно о коммерческом и учебном лицензировании см. по адресу *www.clearviewtraining.com*.

Как читать эту книгу

Так много книг и так мало времени, чтобы прочитать их все! Помня об этом, мы спланировали эту книгу так, что ее можно читать по-разному (в том числе и от корки до корки) соответственно вашим нуждам.

По ускоренной схеме

Выберите ускоренную схему, если хотите просто просмотреть всю книгу или отдельную главу. Кроме того, это способ получить сжатый смысл главы или книги.

- Выберите главу.
- Прочитайте план главы, чтобы знать, о чем пойдет речь.

- Просмотрите главу, останавливаясь на рисунках и примечаниях в рамках.
- Прочитайте раздел «Что мы узнали».
- Вернитесь и прочитайте любой заинтересовавший вас раздел.

Ускоренная схема – это быстрый и эффективный способ чтения этой книги. Возможно, вас приятно удивит, как много информации можно почерпнуть! Обратите внимание, что ускоренная схема эффективнее, если вы с самого начала можете четко сформулировать, какую информацию хотите получить. Например: «Я хочу понять, как осуществлять моделирование прецедентов».

Для справки

Если вам необходимо знать конкретную часть UML или изучить определенный технический прием, мы предоставили подробный индекс и оглавление, которые помогут найти необходимую информацию быстро и эффективно. Чтобы помочь в этом, в тексте используются точные перекрестные ссылки.

Просмотреть

Существует две стратегии просмотра данного текста.

- Если необходимо максимально эффективно и быстро освежить знания по UML, прочитайте краткие обзоры, приведенные в разделе «Что мы узнали» каждой главы. Если что-то непонятно, вернитесь и прочитайте соответствующий раздел.
- Если вы располагаете большим количеством времени, можно просмотреть каждую главу, изучая диаграммы и прочитывая примечания в рамках.

Пробежать глазами

Если у вас есть пара свободных минут, можно взять книгу и открыть ее на любой странице. Мы попытались сделать так, чтобы на каждой странице было что-то интересное. Если даже вы уже довольно хорошо знаете UML, все равно можно найти что-то новое.

План книги

На рис. 2 представлен план книги. Мы показали, какие главы можно читать в любом порядке, а какие можно пропустить при первом чтении, поскольку они обсуждают усовершенствованные технические приемы.

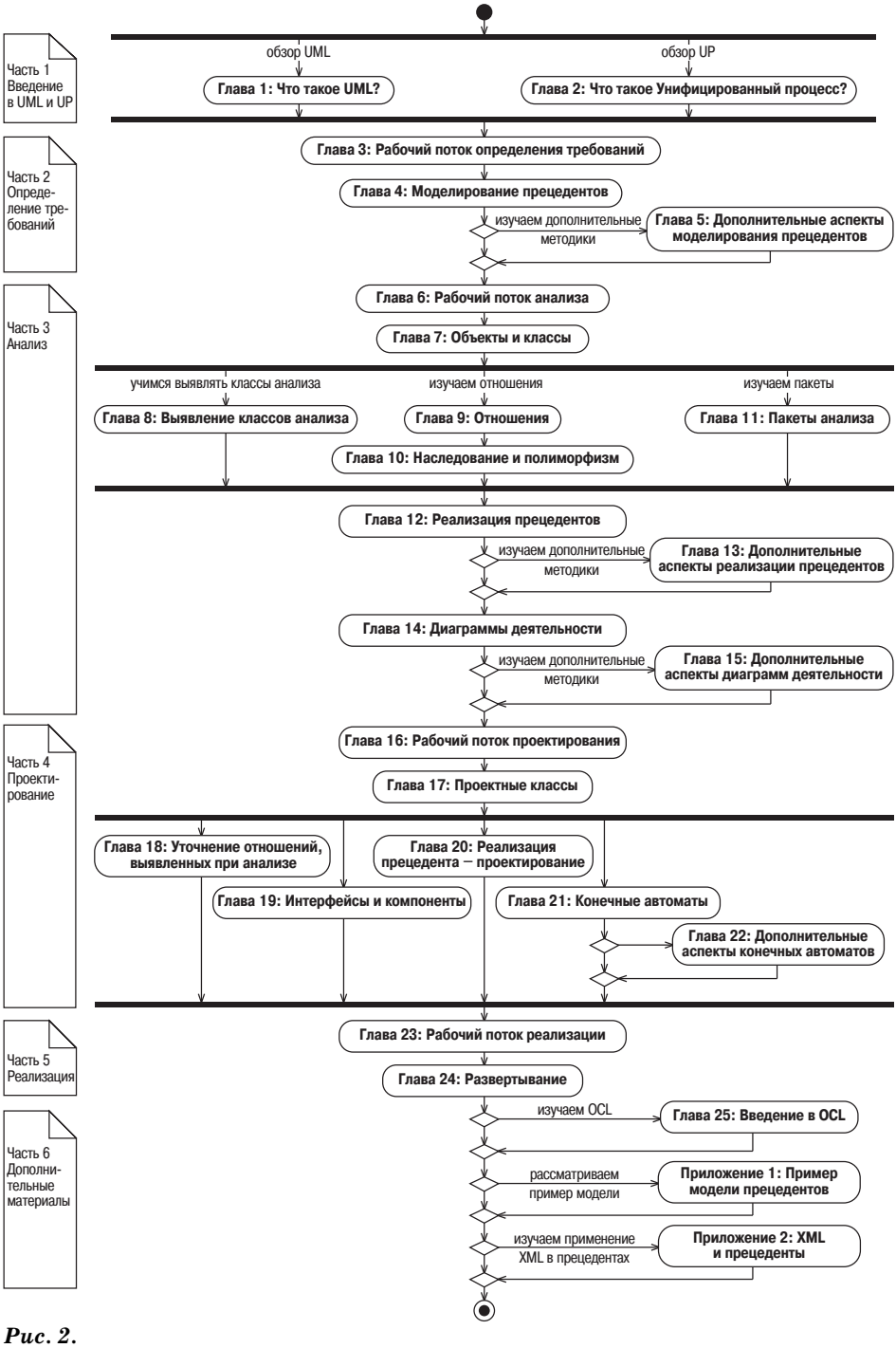

Часть 1 Введение в UML и UP
обзор UML
обзор UP
Глава 1: Что такое UML?
Глава 2: Что такое Унифицированный процесс?
Часть 2 Определение требований
Глава 3: Рабочий поток определения требований
Глава 4: Моделирование прецедентов
изучаем дополнительные методики
Глава 5: Дополнительные аспекты моделирования прецедентов
Часть 3 Анализ
Глава 6: Рабочий поток анализа
Глава 7: Объекты и классы
учимся выявлять классы анализа
изучаем отношения
изучаем пакеты
Глава 8: Выявление классов анализа
Глава 9: Отношения
Глава 11: Пакеты анализа
Глава 10: Наследование и полиморфизм
Глава 12: Реализация прецедентов
изучаем дополнительные методики
Глава 13: Дополнительные аспекты реализации прецедентов
Глава 14: Диаграммы деятельности
изучаем дополнительные методики
Глава 15: Дополнительные аспекты диаграмм деятельности
Часть 4 Проектирование
Глава 16: Рабочий поток проектирования
Глава 17: Проектные классы
Глава 18: Уточнение отношений, выявленных при анализе
Глава 20: Реализация прецедента – проектирование
Глава 19: Интерфейсы и компоненты
Глава 21: Конечные автоматы
Глава 22: Дополнительные аспекты конечных автоматов
Часть 5 Реализация
Глава 23: Рабочий поток реализации
Глава 24: Развертывание
Часть 6 Дополнительные материалы
изучаем OCL
Глава 25: Введение в OCL
рассматриваем пример модели
Приложение 1: Пример модели прецедентов
изучаем применение XML в прецедентах
Приложение 2: XML и прецеденты

Рис. 2.

14

Диаграммы деятельности

14.1. План главы

Диаграммы деятельности – это «ОО блок-схемы». Они позволяют моделировать процесс как деятельность, состоящую из коллекции соединенных ребрами узлов. UML 2 вводит новую семантику диаграмм деятельности, которая обеспечивает им намного большую мощь и гибкость, чем ранее. В этой главе рассматриваются основы диаграмм деятельности в объеме, достаточном для моделирования деятельности. Более глубокие вопросы рассматриваются в следующей главе.

14.2. Что такое диаграммы деятельности

Диаграммы деятельности часто называют «ОО блок-схемами». Они позволяют моделировать процесс как деятельность, которая состоит из коллекции соединенных ребрами узлов.

В UML 1 диаграммы деятельности фактически были лишь особым случаем диаграмм состояний (глава 21), где у каждого состояния было входное действие, которое определяло некоторый процесс или функцию, имеющие место при входе в состояние. В UML 2 диаграммы деятельности имеют совершенно новую семантику, базирующуюся на технологии сетей Петри (Petri Nets). В использовании этой технологии есть два преимущества:

1. Формализм сети Петри обеспечивает большую гибкость при моделировании различных типов потока.
2. В UML теперь есть четкое разделение между диаграммами деятельности и диаграммами состояний.

> Диаграммы деятельности – это ОО блок-схемы.

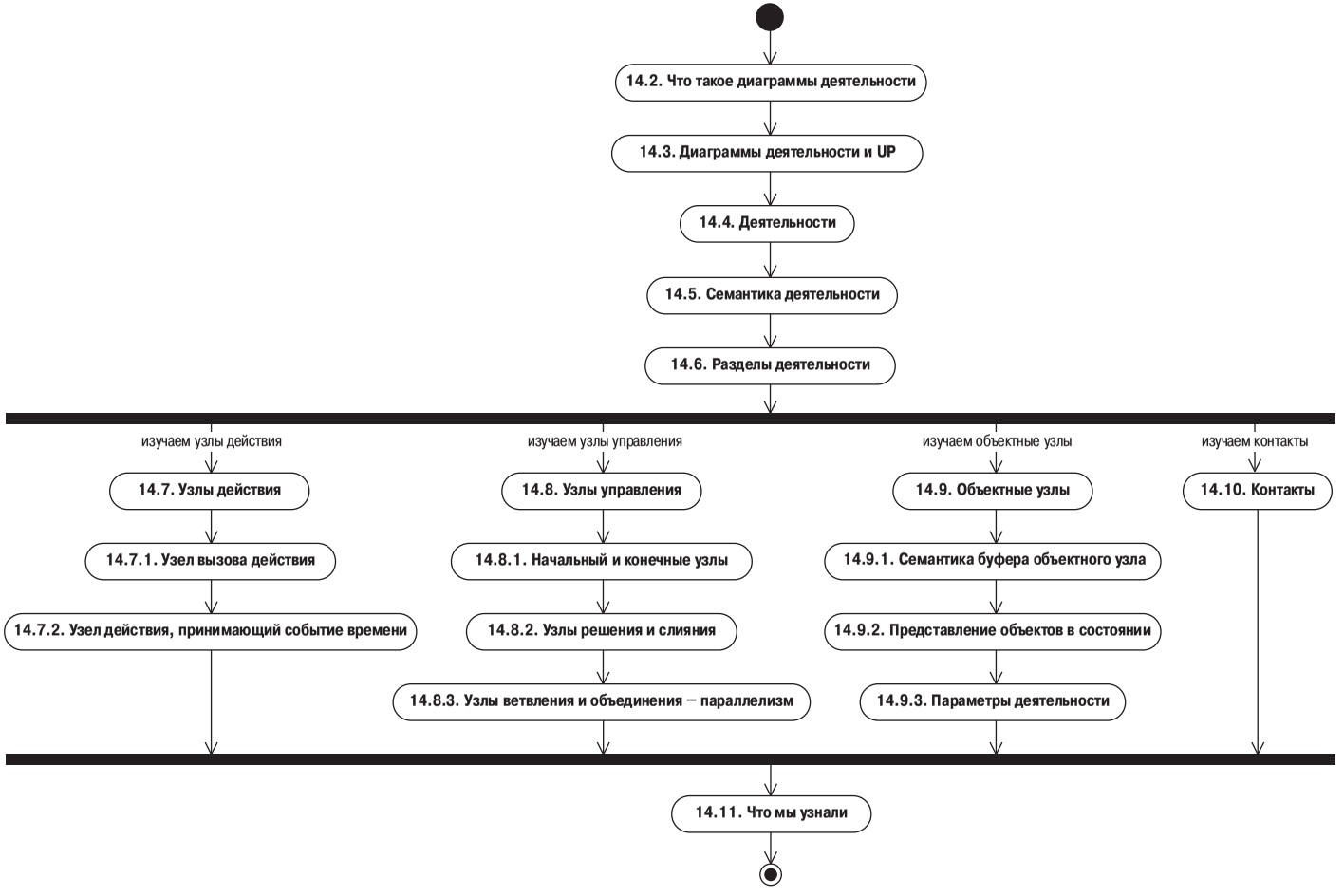


Рис. 14.1. *План главы*

Деятельность может быть добавлена к *любому* элементу модели с целью моделирования его поведения. Элемент обеспечивает контекст для деятельности, и деятельность может использовать возможности своего контекста. Деятельности обычно добавляются к:

- прецедентам;
- классам;
- интерфейсам;
- компонентам;
- кооперациям;
- операциям.

Диаграммы деятельности также могут использоваться для моделирования бизнес-процессов и рабочих потоков. Мы кратко покажем, как это делать, но более сложные аспекты выходят за рамки этой книги.

Хотя диаграммы деятельности обычно используются как блок-схемы операций, следует отметить, что исходный код операции в виде кода или псевдокода, возможно, является лучшим и более кратким их представлением! Так что о каждом случае необходимо судить отдельно.

Хорошая диаграмма деятельности сосредоточена на отражении лишь одного определенного аспекта динамического поведения системы. Таким образом, она должна находиться на соответствующем уровне абстракции, чтобы донести эту идею до целевой аудитории, и содержать минимум необходимой информации. Диаграммы деятельности можно дополнять состояниями и потоками объектов, но необходимо постоянно спрашивать себя, проясняют ли эти элементы диаграмму или делают ее еще более запутанной? Как обычно, лучше придерживаться максимальной простоты.

14.3. Диаграммы деятельности и UP

> Диаграммы деятельности могут использоваться во многих рабочих потоках UP.

Благодаря своей гибкости диаграммы деятельности не имеют одного определенного назначения в UP. Они обеспечивают универсальный механизм моделирования поведения и могут использоваться везде, где возникает необходимость в их применении. Мы обсуждаем их в рабочем потоке анализа, потому что чаще всего эти диаграммы используются при анализе.

Уникальная способность диаграмм деятельности в том, что они позволяют моделировать процесс *без* необходимости определения статической структуры классов и объектов, реализующих процесс. Несомненно, это очень полезно на ранних этапах анализа при попытках выяснить, что представляет собой конкретный процесс.

Из собственного опыта можем сказать, что диаграммы деятельности чаще всего используются в следующих случаях.

- В процессе анализа:
  - для графического моделирования потока прецедента. Такое представление является более понятным для заинтересованных сторон;
  - для моделирования потока между прецедентами. При этом используется особая форма диаграммы деятельности – диаграмма обзора взаимодействий (раздел 15.12).
- При проектировании:
  - для моделирования деталей операции;
  - для моделирования деталей алгоритма.
- При моделировании деловой активности:
  - для моделирования бизнес-процесса.

Обычно заказчикам проще понимать диаграммы деятельности, поскольку большинство из них имеет представление о блок-схемах в той или иной форме. Следовательно, диаграммы деятельности могут быть замечательным средством общения, если *они просты*.

Как будет видно в этой и следующей главах, UML 2 представляет много мощных нововведений в синтаксисе и семантике диаграмм деятельности. Важно не слишком сильно увлекаться всем этим. При создании *любой* UML-диаграммы всегда нужно помнить о целевой аудитории и использовать возможности UML соответственно. Нет смысла применять все самые последние нововведения, если никто не поймет диаграмму.

14.4. Деятельности

Деятельности – это системы *узлов* (*nodes*), соединенных *ребрами* (*edges*). Существует три категории узлов:

1. Узлы действия (action nodes) – представляют отдельные единицы работы, элементарные *в рамках деятельности*;
2. Узлы управления (control nodes) – управляют потоком деятельности;
3. Объектные узлы (object nodes) – представляют объекты, используемые в деятельности.

Ребра представляют потоки деятельности. Существует два типа ребер:

1. Ребра потоков управления (control flows) – представляют поток управления деятельности;

> Деятельности – это системы узлов, соединенных ребрами.

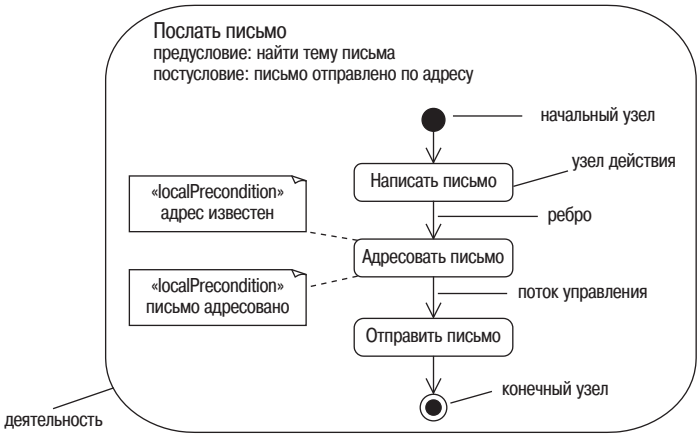


Рис. 14.2. *Диаграмма деятельности бизнес-процесса Послать письмо*

2. Ребра потоков объектов (object flows) – представляют поток объектов деятельности.

Все типы узлов и ребер будут подробно изучены в последующих разделах.

Давайте рассмотрим пример. На рис. 14.2 показана простая диаграмма деятельности бизнес-процесса Послать письмо. Обратите внимание, что деятельности могут иметь предусловия и постусловия, как и прецеденты. Предусловия – это условия, которые должны быть истинными, чтобы деятельность могла начаться, а постусловия – это условия, которые будут истинными по завершении деятельности. Действия внутри деятельности тоже могут иметь собственные локальные предусловия и постусловия, как показано на рис. 14.2.

Деятельности обычно начинаются с одного узла управления, начального. Он обозначает начало исполнения при вызове деятельности. Один или более конечных узлов показывают места завершения деятельности.

В примере, изображенном на рис. 14.2, деятельность начинается в начальном узле. Затем управление переходит вдоль ребра к узлу действия Написать письмо. Этот узел обозначает элементарную часть работы или поведения, поскольку содержит описываемую деятельность. Поток проходит через узлы Адресовать письмо, Отправить письмо и затем к конечному узлу, в котором деятельность завершается.

Обычно диаграммы деятельности используются для моделирования прецедента в виде последовательностей действий. На рис. 14.3 показан прецедент PaySalesTax (выплата налога с оборота) из главы 4. Этот прецедент может быть представлен в виде диаграммы деятельности, как показано на рис. 14.4.

Прецедент: PaySalesTax
ID: 1
Краткое описание: Выплата налога с оборота в налоговое управление по окончанию налогового периода.
Главные актеры: Time
Второстепенные актеры: TaxAuthority
Предусловия: 1. Конец налогового периода.
Основной поток: 1. Прецедент начинается в конце налогового периода. 2. Система определяет сумму налога с оборота, которую необходимо выплатить TaxAuthority. 3. Система посылает электронный платеж в TaxAuthority.
Постусловия: 1. TaxAuthority получает соответствующую сумму налога с оборота.
Альтернативные потоки: Нет.

Рис. 14.3. *Спецификация прецедента PaySalesTax*

Обратите внимание, что диаграмма деятельности – это более компактная и графическая форма прецедента. Диаграмма деятельности представляет прецедент как два действия: Вычисление налога с оборота и Отправка электронного платежа. Каждое из этих действий могло бы быть представлено в виде отдельной диаграммы деятельности. Вероятно, так и произойдет при проектировании, когда понадобится показать способ реализации этих действий. Актер и его взаимодействие с системой – это структурные элементы, поэтому их нет на этой диаграмме.

Прецеденты представляют поведение системы как взаимодействие актеров и системы, тогда как диаграммы деятельности отображают его в виде последовательности действий. Они являются дополнительными представлениями этого поведения.

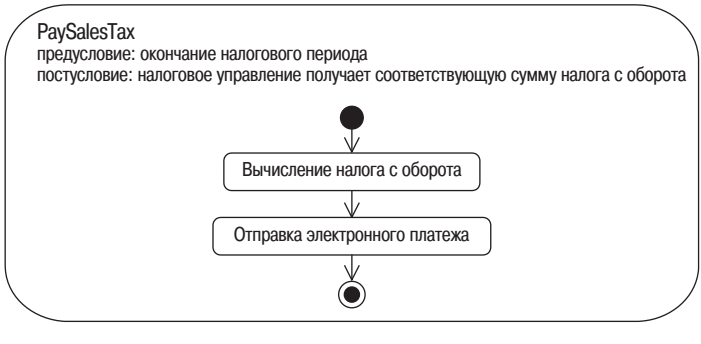


Рис. 14.4. *Диаграмма деятельности прецедента PaySalesTax*

14.5. Семантика деятельности

> Диаграммы деятельности основаны на технологии сетей Петри.

Семантика диаграмм деятельности, как следует из предыдущего изложения, интуитивно довольно проста. Данный раздел посвящен подробному описанию семантики деятельности.

Диаграммы деятельности UML 2 основаны на технологиях сетей Петри. Сети Петри выходят за рамки нашей книги. Более подробную информацию о них можно найти по адресу *www.daimi.au.dk/PetriNets/*.

Диаграммы деятельности моделируют поведение с помощью *«игры» маркеров* (*token game*). Эта игра описывает поток маркеров, движущийся по сети узлов и ребер согласно определенным правилам. Маркеры на диаграммах деятельности UML могут представлять:

- поток управления;
- объект;
- некоторые данные.

Состояние системы в любой момент времени определяется расположением ее маркеров.

В примере на рис. 14.2 маркер – это поток управления, поскольку в рассматриваемом случае между узлами не происходит передачи объектов или данных.

Маркеры перемещаются вдоль ребра (edge) от начального узла (source node) к целевому узлу (target node). Перемещение маркера происходит только при выполнении *всех* необходимых условий. Условия меняются в зависимости от типа узла. Для узлов, представленных на рис. 14.5 (узлы действия), этими условиями являются:

- постусловия начального узла;
- сторожевые условия ребра;
- предусловия целевого узла.

Условия существуют не только для узлов действия, но и для узлов управления и объектных узлов. Узлы управления имеют особую семантику, которая управляет тем, как передаются маркеры от входных ребер к выходным. Например, начальный узел начинает деятельность, конечный узел (final node) завершает деятельность, а узел объединения будет предлагать маркер на своем единственном исходящем ребре только в случае, если маркеры поступили на все его входные ребра. Объектные узлы представляют объекты, существующие в системе. Узлы управления и объектные узлы подробно обсуждаются в разделах 14.8 и 14.9 соответственно.

Рассмотрим «игру» маркеров для деятельности, показанной на рис. 14.5. С началом выполнения деятельности в начальном узле стартует поток

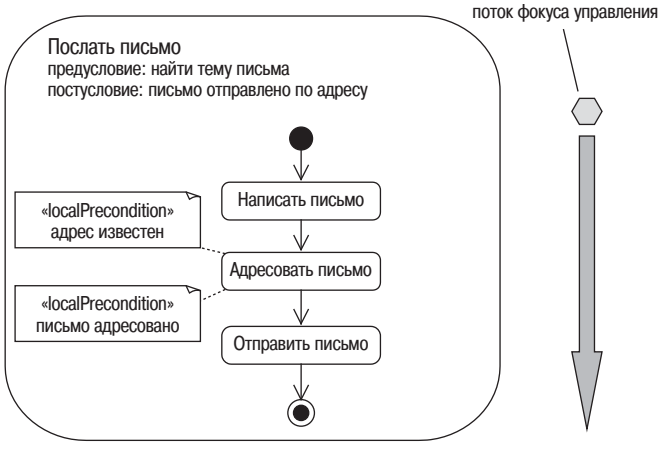


Рис. 14.5. *Диаграмма деятельности Послать письмо*

маркера управления. Ни на начальный узел, ни на его выходное ребро, ни на целевой узел не наложено никаких условий, поэтому маркер автоматически проходит по выходному ребру к целевому узлу Написать письмо. Это инициирует выполнение действия, определенного узлом действия Написать письмо. По завершении действия Написать письмо поток маркера управления переходит в узел действия Адресовать письмо тогда и только тогда, когда удовлетворено его предусловие адрес известен. После выполнения постусловия письмо адресовано управление переходит от Адресовать письмо к Послать письмо. И наконец, поскольку условий, препятствующих выходу потока из Послать письмо, нет, поток управления прибывает в конечное состояние и деятельность завершается.

В этом простом примере поток управления проходит по всем узлам действия по очереди, обусловливая их выполнение. Это основная семантика деятельности.

Как уже упоминалось, состояние исполняющейся системы в любой момент времени может быть представлено расположением ее маркеров. Например, когда маркер находится в узле Написать письмо, можно сказать, что система находится в состоянии Написание письма. Однако не каждое выполнение действия или передача маркера создает заметное изменение в состоянии системы с точки зрения ее конечных автоматов (глава 21). Тем не менее расположение маркеров обеспечивает связь между диаграммами деятельности и диаграммами состояний, которые должны быть гарантированно согласованными для конкретного элемента модели.

Хотя семантика деятельностей UML 2 *описывается* «игрой» маркеров, они едва ли когда-нибудь реализуются таким образом. По сути, деятельность – это всего лишь описание, для которого возможно мно-

жество реализаций. Например, на рис. 14.5 описывается простой бизнес-процесс, а не программная система, и реализации этого процесса вообще *не* используют передачу маркеров!

14.6. Разделы деятельности

> Каждый раздел деятельности представляет высокоуровневую группировку взаимосвязанных действий.

Чтобы облегчить чтение диаграмм деятельности, можно разбить деятельности на разделы с помощью вертикальных, горизонтальных или кривых линий. Каждый раздел деятельности – это группа взаимосвязанных действий с высоким уровнем вложенности. Иногда разделы деятельности называют плавательными дорожками (swimlanes). Разбиение на разделы – мощный метод. При правильном использовании он может существенно упростить понимание диаграмм деятельности.

В UML 2 семантику разделов деятельности определяет разработчик модели. Разделы не имеют собственной семантики, поэтому могут использоваться для разбиения диаграмм деятельности любым удобным способом! Разделы деятельности обычно применяются для представления:

- прецедентов;
- классов;
- компонентов;
- организационных единиц (в бизнес-моделировании);
- ролей (в моделировании рабочих потоков).

Но этим не ограничиваются. Например, в проектных моделях распределенных систем разделы деятельности могут использоваться даже для моделирования распределения процессов на физических машинах.

У каждого множества разделов должно быть единственное измерение, описывающее его базовую семантику. В рамках этого измерения разделы могут быть иерархично вложенными. На рис. 14.6 показана деятельность, имеющая иерархично вложенное множество разделов деятельности.

Местонахождение – это измерение, и в рамках этого измерения существует иерархия разделов, как показано на рис. 14.7. Эта диаграмма моделирует бизнес-процесс создания курса нашей партнерской компании Zuhlke Engineering AG. Многие из их курсов разрабатываются нами в Лондоне.

Часто разделы деятельности и параллельные потоки управления взаимосвязаны. Моделирование параллелизма на диаграммах деятельности подробно рассматривается в разделе 14.8.3. Например, обычно разные подразделения или организационные единицы работают параллельно, затем в некоторый момент происходит их синхронизация.

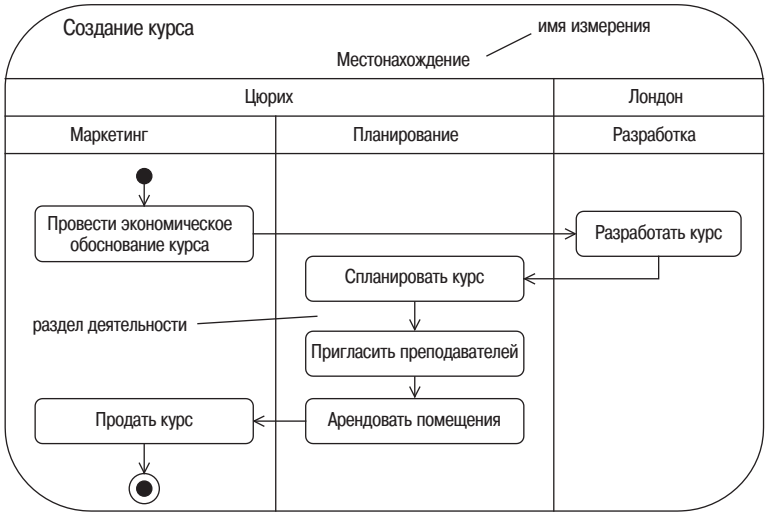


Рис. 14.6. *Деятельность с иерархично вложенным множеством разделов*

Диаграммы деятельности с разделами деятельности превосходно подходят для моделирования подобного процесса.

Иногда абсолютно нереально сгруппировать узлы в вертикальные или горизонтальные разделы, не ухудшив при этом читаемость диаграммы. В этом случае можно создавать разделы неправильной формы с помощью кривых или обозначать разделы с помощью текста. В UML существует текстовая нотация для разделов деятельности. Пример текстовой нотации приведен на рис. 14.8. Однако обычно к ней прибегают в самом крайнем случае, потому что графическое представление, как правило, намного нагляднее.

Местоположение действия в иерархии раздела можно задать с помощью разделенного двойными двоеточиями пути (pathname); путь указывается в скобках над именем действия. Если действие встречается в нескольких разделах, через запятую перечисляют имена путей каждого из разделов (рис. 14.8).

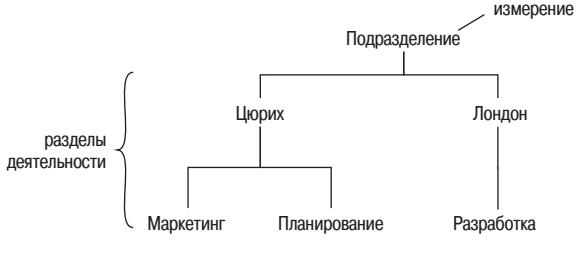


Рис. 14.7. *Иерархия разделов в рамках измерения*

Рис. 14.8. *Примеры текстовой нотации для разделов деятельности*

> Разделы, обозначенные стереотипом **«external»**, не являются частью системы.

Порой необходимо показать на диаграмме деятельности поведение, находящееся, строго говоря, вне области действия системы, например взаимодействие системы с некоторой внешней системой. Это можно представить, указав на диаграмме деятельности стереотип «external» прямо над именем раздела. Обратите внимание, что внешний раздел не является частью системы и, следовательно, не может входить в какую-либо иерархию разделов модели.

Тщательно выбирая измерения и разделы деятельности, можно, безусловно, добавить на диаграмму деятельности массу полезной информации. Однако эти возможности могут и усложнить диаграмму, особенно при наличии нескольких измерений и сложной иерархии разделов! На практике необходимо стремиться к использованию на одной диаграмме не более трех уровней иерархии (включая измерение) и не более двух измерений.

Всегда опирайтесь на собственное мнение и применяйте разделы деятельности только в случае, если они действительно повышают ценность модели.

14.7. Узлы действия

Узлы действия (action node) исполняются в следующих случаях:

- маркеры одновременно поступили на все входящие ребра
- и входящие маркеры удовлетворяют всем локальным предусловиям узла действия.

Это проиллюстрировано на рис. 14.9.

Узлы действия осуществляют операцию логическое И над своими входящими маркерами – узел не готов к исполнению до тех пор, пока маркеры не будут присутствовать на *всех* входящих ребрах. Даже если все необходимые маркеры присутствуют, узел будет исполняться только после удовлетворения его локального предусловия.

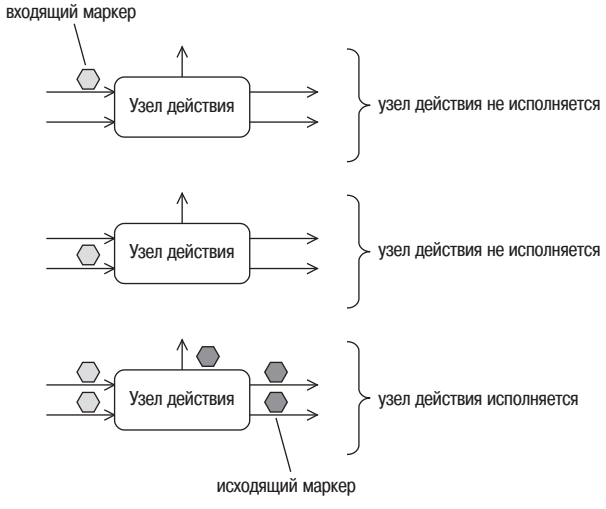


***Рис. 14.9.** Условия исполнения узлов действия*

> Узлы действия предлагают управляющие маркеры на *всех* своих исходящих ребрах – неявное ветвление.

По завершении выполнения узла действия проверяется локальное постусловие. Если оно удовлетворено, узел одновременно предлагает маркеры на *всех* своих исходящих ребрах. Это ветвление является неявным, поскольку один узел действия может породить множество потоков. В отличие от обычных блок-схем, диаграммы деятельности по сути своей параллельны.

Поскольку в узлах действия производятся некоторые действия, обычно их имена являются глаголами или глагольными группами. Спецификация UML не дает никаких рекомендаций по присваиванию имен узлам действия. Мы пользуемся соглашением, по которому имя узла начинается с большой буквы, все остальные слова, входящие в имя, пишутся с маленькой буквы через пробелы. Единственное исключение из этого правила: узел действия содержит ссылку на другой элемент модели. В этом случае имя элемента модели всегда используется как есть, *без* изменения регистра или добавления пробелов. На рис. 14.10 показано два примера. В верхнем примере происходит ссылка на что-то

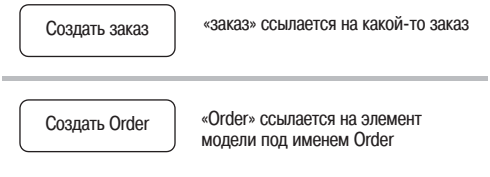


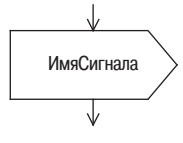
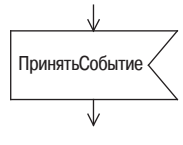
***Рис. 14.10.** Примеры имен узлов действия*

под названием «заказ», тогда как нижний пример явно ссылается на класс `Order`, который можно найти где-то в модели.

Детали действия фиксируются в описании узла действия. Часто это обычное текстовое описание, такое как «Написать письмо», но на этапе проектирования оно превращается в структурированный текст, псевдокод или реальный код. При моделировании прецедента диаграмма деятельности могла бы опережать поток прецедента на два-три шага. Но это может вызвать трудности, поскольку придется постоянно синхронизировать прецедент и соответствующую диаграмму деятельности.

Существует четыре типа узлов действия; они перечислены в табл. 14.1. Подробное обсуждение типов узлов действия можно найти в разделах, указанных в таблице.

Таблица 14.1

Синтаксис	Имя	Семантика	Раздел
	Узел вызова действия	Инициирует деятельность, поведение или операцию.	14.7.1
	Посылка сигнала	Действие посылки сигнала – посылает сигнал асинхронно (отправитель *не* ожидает подтверждения получения сигнала). Для создания сигнала может принимать входные параметры.	15.6
	Узел действия, принимающий событие	Принимает событие – ожидает события, установленного объектом-владельцем, и выдает событие на выходе. Активируется при получении маркера по входящему ребру. Если *нет* входящего ребра, запускается при запуске включающей его деятельности и всегда является активированным.	15.6
	Узел действия, принимающий событие времени	Принимает событие времени – отвечает на определенное значение времени. Генерирует события времени соответственно своему временному выражению.	14.7.2

14.7.1. Узел вызова действия

Самыми распространенными узлами действий являются *узлы вызова действия* (*call action node*). Этот тип узлов может инициировать:

- деятельность;
- поведение;
- операцию.

> Узел вызова действия может инициировать деятельность, поведение или операцию.

Некоторые примеры синтаксиса узла вызова действия приведены на рис. 14.11. Как видно из рисунка, синтаксис очень гибок!

- Посредством специального символа «грабли» в нижнем правом углу пиктограммы узла можно указать, что действие вызывает другое действие. Имя узла соответствует имени вызываемой им деятельности.
- Можно вызвать поведение – это прямой вызов поведения контекста деятельности *без* указания какой-либо конкретной операции.
- Можно вызвать операцию, используя стандартный синтаксис операции, описанный в разделе 7.5.3.
- Можно вызвать операцию, описывая ее детали на конкретном языке программирования. Это может быть особенно полезным при использовании инструментального средства UML, позволяющего генерировать код из диаграмм деятельности (например, iUML от компании Kennedy Carter, *www.kc.com*).
- С помощью ключевого слова self можно использовать возможности контекста деятельности.

Узлы вызова действия, используемые в диаграммах деятельности на уровне анализа, обычно вызывают поведение. Узлы вызова действия, инициирующие конкретные операции, как правило, используются для более детального моделирования деятельности при проектировании.

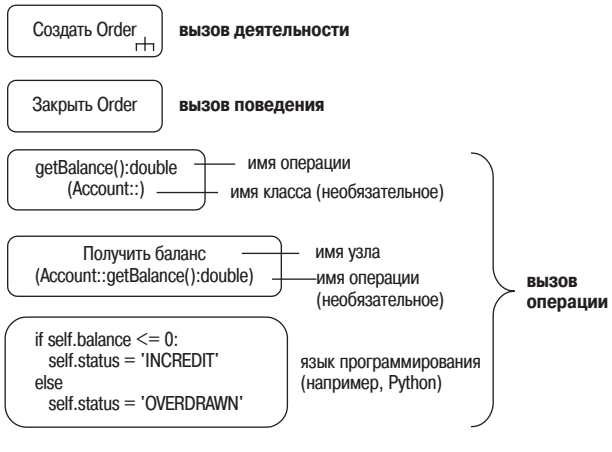


Рис. 14.11. *Примеры синтаксиса узла вызова действия*

14.7.2. Узел действия, принимающий событие времени

> Узел действия, принимающий события времени, реагирует на время.

Узел действия, принимающий события времени, реагирует на время. Этот тип узла имеет временное выражение и генерирует событие времени, когда это выражение становится истинным. Поведение такого узла зависит от наличия входящего ребра.

Например, на рис. 14.12 показан узел действия, принимающий события времени, без входящего ребра. Этот узел станет активным и будет генерировать событие времени после запуска его деятельности-владельца, когда его временное выражение становится истинным. В приведенном примере событие времени генерируется в конце каждого финансового года и инициирует деятельность Отправить налоговую декларацию компании.

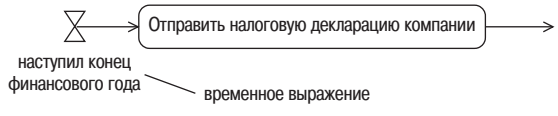


***Рис. 14.12.** Узел действия, принимающий событие времени, без входящего ребра*

Однако в примере на рис. 14.13 действие, принимающее событие времени, имеет входящее ребро и станет активным только после получения маркера по этому ребру. Этот пример является фрагментом системы управления лифтом. Первое действие открывает дверь лифта и запускает действие, принимающее событие времени. Это действие ожидает десять секунд и затем передает маркер действию Закрыть дверь.

***Рис. 14.13.** Узел действия, принимающий событие времени, с входящим ребром*

Обратите внимание, что временное выражение может указывать на:

- некоторое событие (например, конец финансового года);
- конкретный момент времени (например, 11/03/1960);
- временной интервал (например, ожидать 10 секунд).

14.8. Узлы управления

Узлы управления контролируют поток управления деятельности. В табл. 14.2 представлены все узлы управления UML 2; их подробное обсуждение см. в следующих разделах.

Таблица 14.2

Синтаксис	Имя	Семантика		Раздел
	Начальный узел	Указывает, где начинается поток при вызове деятельности.		14.8.1
	Конечный узел деятельности	Завершает деятельность.	Конечные узлы	14.8.1
	Конечный узел потока	Завершает определенный поток деятельности – другие потоки не затрагиваются.		14.8.1
	Узел решения	Поток проходит по исходящему ребру, сторожевое условие которого истинно. Может иметь входные данные (необязательно).		14.8.2
	Узел слияния	Копирует входные маркеры в единственное выходное ребро.		14.8.2
	Узел ветвления	Разделяет поток на несколько параллельных потоков.		14.8.3
	Узел объединения	Синхронизирует несколько параллельных потоков. Может иметь описание объединения (не обязательно) для изменения его семантики.		14.8.3

14.8.1. Начальный и конечный узлы

> Начальный узел показывает, где начинается деятельность.

Как уже говорилось в разделе 14.4, начальный узел (initial node) – это точка, в которой начинается поток при вызове деятельности. У деятельности может быть более одного начального узла. В этом случае потоки запускаются во всех начальных узлах одновременно и выполняются параллельно.

> Конечный узел деятельности завершает все потоки деятельности.

Деятельность также может быть инициирована действием принятия события (раздел 15.6) или узлом, являющимся параметром (раздел 14.9.3). Таким образом, начальные узлы *не* являются обязательными, поскольку есть другие способы запуска деятельности.

> Конечный узел потока завершает один из потоков деятельности.

Конечный узел (final node) деятельности завершает *все* потоки деятельности. Конечных узлов деятельности может быть много, и тот, который будет активирован первым, завершит все остальные потоки и саму деятельность.

Конечный узел потока просто останавливает *один* из потоков деятельности, остальные потоки продолжают выполнение. Пример приведен на рис. 15.10.

14.8.2. Узлы решения и слияния

Узел решения имеет одно входящее ребро и два и более альтернативных исходящих ребер. Маркер, поступающий по входящему ребру, будет предложен *всем* исходящим ребрам, но пройдет только по *одному* из них. Узел решения – это перекресток потоков, на котором маркер должен выбрать только один путь.

> Узел решения передает маркер на то выходное ребро, для которого выполняется сторожевое условие.

Каждое выходное ребро защищено *сторожевым условием* (*guard condition*), которое означает, что ребро примет маркер только в случае выполнения сторожевого условия. Важно, чтобы сторожевые условия были гарантированно взаимоисключающими, т. е. чтобы в любой момент времени истинным могло быть только *одно* из них. В противном случае согласно спецификации UML 2 поведение узла решения формально является неопределенным!

Для задания ребра, по которому пройдет поток управления в случае невыполнения всех сторожевых условий, может использоваться ключевое слово else.

На рис. 14.14 показан простой пример узла решения. После действия Получить корреспонденцию поток управления попадает в узел решения. Если выполняется условие [это мусор], почта отправляется в мусорную корзину, в противном случае (else) почтовое сообщение открывается.

Узел, отмеченный стереотипом «decisionInput» (входные данные решения), представляет условие принятия решения. Его результат используется сторожевыми условиями на исходящих ребрах. Пример фрагмента деятельности показан на рис. 14.15. Здесь условие принятия решения сравнивает запрашиваемую для снятия сумму с балансом счета. Если баланс больше или равен запрашиваемой сумме, условие принимает значение истина и поток переходит к действию Снять сумму. В противном случае регистрируется неплатежеспособность.

На рис. 14.14 показан узел слияния (merge node). В узлах слияния сходятся два или более входящих ребра и выходит одно исходящее. Они объединяют все входящие потоки в один исходящий. Семантика слияния очень проста: все маркеры, предлагаемые на входящих ребрах, предлагаются на исходящем ребре. Маркеры и поток не изменяются.

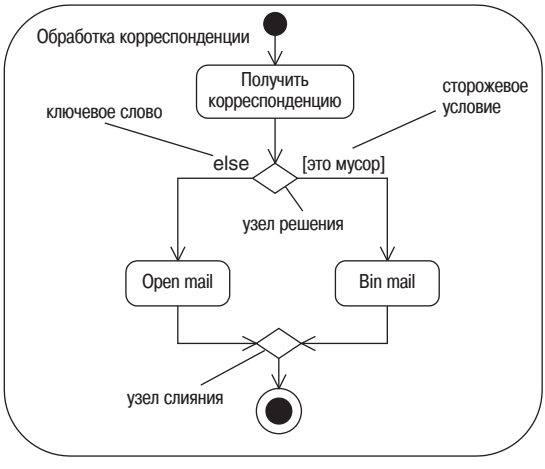


Рис. 14.14. *Пример узла решения и узла слияния*

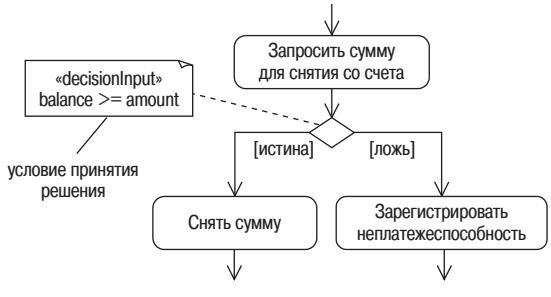


Рис. 14.15. *Фрагмент деятельности с узлом, помеченным стереотипом «decisioninput»*

Узел слияния и непосредственно следующий за ним узел решения могут быть объединены в один символ, как показано на рис. 14.16. Однако мы не рекомендуем применять такую сокращенную нотацию, поскольку изображение узлов по отдельности придает диаграмме большую наглядность.

14.8.3. Узлы ветвления и объединения – параллелизм

> Узел ветвления разделяет поток на несколько параллельных потоков.

Параллельные потоки деятельности можно создать путем разделения одного потока с помощью узла ветвления. Хотя обычно параллелизм является решением, принимаемым во время проектирования, нередко необходимо показать параллельные деятельности при моделировании бизнес-процессов. По этой причине мы будем часто использовать узлы ветвления и объединения как при анализе, так и при проектировании.

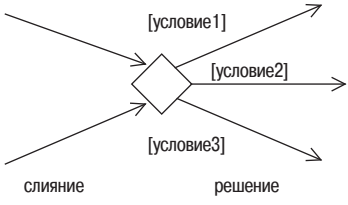


***Рис. 14.16.** Узел слияния и узел решения могут быть объединены в один символ*

Узел ветвления имеет одно входящее и два или более исходящих ребер. Маркеры, поступающие по входящим ребрам, дублируются и предлагаются на *всех* исходящих ребрах одновременно. Тем самым единственный входящий поток разделяется на несколько параллельных исходящих потоков. У каждого исходящего ребра может быть сторожевое условие, и маркер, как и в узлах решения, может передаваться по исходящему ребру только в случае выполнения сторожевого условия.

> Узел объединения синхронизирует и объединяет несколько входящих потоков в единственный исходящий.

В узле объединения несколько входящих ребер встречаются и объединяются в одно исходящее. Эти узлы синхронизируют потоки: маркер на их единственном исходящем ребре предлагается только после того, как поступили маркеры *всех* входящих потоков. Они осуществляют операцию логического И над всеми своими входящими ребрами.

На рис. 14.17 показан простой пример Процесс производства продукта, в котором используются узлы ветвления и объединения. В этом примере:

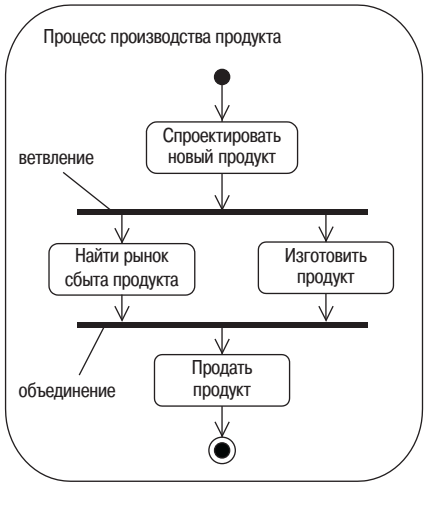


***Рис. 14.17.** Деятельность Процесс производства продукта включает **у**злы ветвления и объединения*

- продукт сначала разрабатывается;
- поиск рынка сбыта и изготовление продукта осуществляются параллельно;
- продукт реализуется *только* после завершения обоих процессов – поиска рынка сбыта и изготовления.

На рис. 14.17 деятельность Процесс производства продукта начинается с действия Спроектировать новый продукт. После этого узел ветвления разделяет единый поток на два параллельных. В одном из этих потоков ведется поиск рынка сбыта продукта (Найти рынок сбыта продукта), в другом – продукт изготавливается (Изготовить продукт). Узел объединения синхронизирует эти два параллельных потока, поскольку ожидает маркер от каждого из параллельных действий. Получив маркер от каждого действия, он предлагает маркер на своем выходном ребре, и поток переходит к действию Продать продукт.

При моделировании узлов объединения важно гарантировать получение маркера всеми входными ребрами. Например, на рис. 14.17 узел объединения никогда не смог бы получить подходящие маркеры для активации, если бы на исходящие потоки ветвления были наложены взаимоисключающие сторожевые условия. Это привело бы к «зависанию» деятельности.

14.9. Объектные узлы

> Объектные узлы показывают, что экземпляры классификатора доступны.

Объектные узлы – это специальные узлы, показывающие, что экземпляры конкретного классификатора доступны в данной точке деятельности. Они обозначены именем классификатора и представляют его экземпляры или подклассы. Фрагмент деятельности на рис. 14.18 показывает объектный узел, представляющий экземпляры классификатора Order или подклассы Order.

> Потоки объектов представляют движение объектов в деятельности.

Входящие и исходящие ребра объектных узлов называют *потоками объектов* (*object flows*). Это особые типы потоков, представляющие

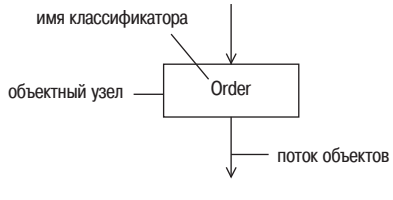


***Рис. 14.18.** Объектный узел*

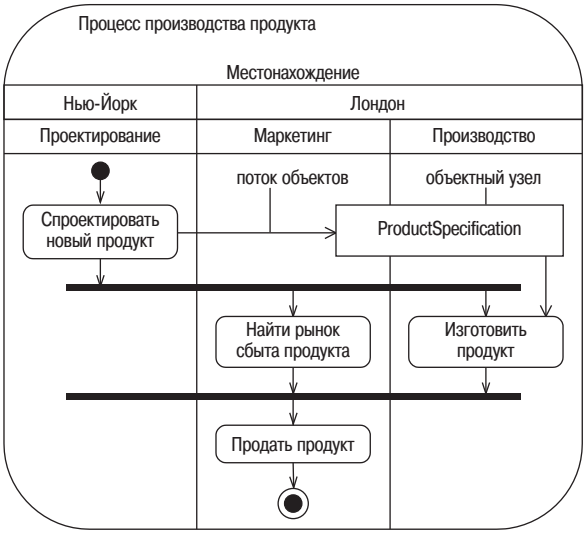


Рис. 14.19. *Деятельность Процесс производства продукта дополнена разделами, действием Спроектировать новый продукт и объектом ProductSpecification*

движение объектов в деятельности. Сами объекты создаются и используются узлами действия.

На рис. 14.19 показана деятельность Процесс производства продукта, впервые представленная на рис. 14.17. Она была дополнена: включены разделы и действием Спроектировать новый продукт создается объект ProductSpecification (спецификация продукта), который используется действием Изготовить продукт для описания производственного процесса.

> Выходные ребра объектного узла конкурируют за каждый выходной маркер.

Когда объектный узел получает объектный маркер по одному из своих входных ребер, он предлагает его всем выходным ребрам одновременно, и эти ребра *конкурируют* за этот маркер. Главное то, что маркер всего один – он *не* тиражируется на все ребра! Этот маркер получает поток, готовый первым принять его.

14.9.1. Семантика буфера объектного узла

Объектные узлы имеют очень интересную семантику. Они действуют как буферы – участки деятельности, где могут находиться объектные маркеры в ожидании принятия другими узлами.

> Объектные узлы выступают в роли буферов.

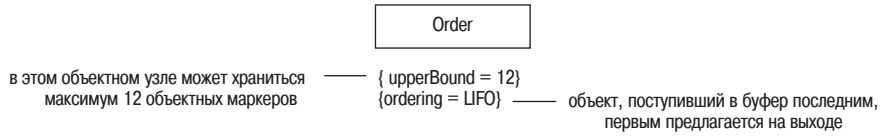


Рис. 14.20. *Объектный узел с верхней границей*

По умолчанию каждый объектный узел может удерживать бесконечное число объектных маркеров. Однако иногда необходимо ограничить размер буфера. Для этого задают *верхнюю границу* (*upper bound*) объектного узла. Она показывает максимальное число маркеров, которые могут удерживаться в узле в любой момент времени. Узел принимает объектные маркеры до тех пор, пока не заполнится. Пример объектного узла с заданной верхней границей приведен на рис. 14.20.

Для объектных узлов можно задать два аспекта семантики буфера.

- У объектных узлов есть *порядок расположения* (*ordering*) (рис. 14.20), определяющий поведение буфера. Применяемым по умолчанию порядком является FIFO (first-in, first-out – первым вошел, первым вышел). Это означает, что объект, первым поступивший в буфер, первым предлагается его выходным ребрам. Существует обратный порядок расположения – LIFO (last-in, first-out – последним вошел, первым вышел).
- Объектные узлы могут обладать *селективным поведением* (*selection behavior*). Это закрепленное за узлом поведение, по которому объекты из входных потоков выбираются согласно некоторому критерию, определенному разработчиком модели. Критерий задается примечанием со стереотипом «selection» (выбор), как показано на рис. 14.21. В данном примере объектный узел выбирает только те объекты Order, которые были созданы в декабре, и предлагает их своим выходным потокам в применяемом по умолчанию порядке (FIFO).

Объектный узел может использоваться для сбора объектов из нескольких входящих объектных потоков или для распределения объектов по нескольким исходящим объектным потокам. В этих случаях узел используется исключительно из-за его буферной семантики. Таким образом, чтобы подчеркнуть этот факт, узел можно обозначить стереотипом «centralBuffer» (центральный буфер).

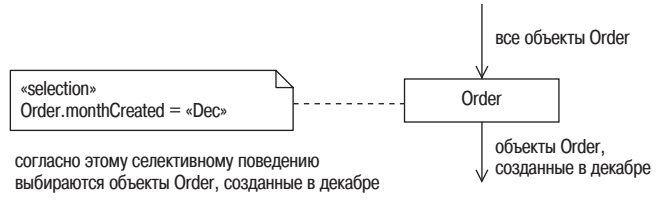


Рис. 14.21. *Объектный узел с селективным поведением*

Наряду с отдельными объектами объектные узлы могут буферизовать *множества* (*sets*) объектов. Множество – это коллекция объектов, в которой нет дублирования, т. е. каждый объект имеет уникальный идентификатор. Чтобы показать это, перед именем классификатора просто указывают Set of (множество). Пример приведен на рис. 14.23.

Немного подробнее стереотипы «selection» и «centralBuffer» рассматриваются в разделах 15.8.2 и 15.11.

14.9.2. Представление объектов в состоянии

> Объектные узлы могут представлять объекты, находящиеся в определенном состоянии.

Объектные узлы могут представлять объекты, находящиеся в определенном состоянии. Например, на рис. 14.22 показан фрагмент деятельности обработки заказа, которая принимает объекты Order, находящиеся в состоянии Открытый, и отправляет их по назначению. Состояния объектов, на которые ссылаются объектные узлы, могут быть смоделированы с помощью конечных автоматов (см. главу 21).

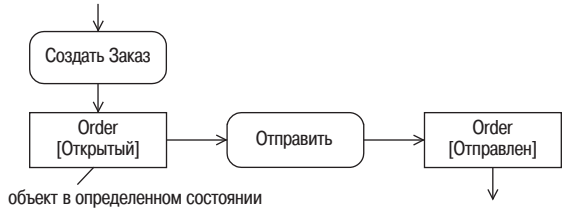


***Рис. 14.22.** Объект Order находится в состоянии Открытый*

14.9.3. Параметры деятельности

> Параметры деятельности – это объектные узлы, поступающие в или исходящие из деятельности.

Объектные узлы могут использоваться для обеспечения входных и выходных данных деятельности, как показано на рис. 14.23. Входящие и исходящие объектные узлы должны перекрывать рамку деятельности. Входящие объектные узлы связаны с деятельностью одним или более исходящими ребрами, а у *исходящих* объектных узлов – одно или более *входящих* ребер, поступающих из деятельности.

На рис. 14.23 деятельность Производство продукта на заказ имеет три входных параметра: CustomerRequest (запрос клиента), Set of BusinessConstraint (множество бизнес-ограничений) и Order, а также один выходной параметр, Order. Узел Set of BusinessConstraint содержит множество объектов BusinessConstraint.

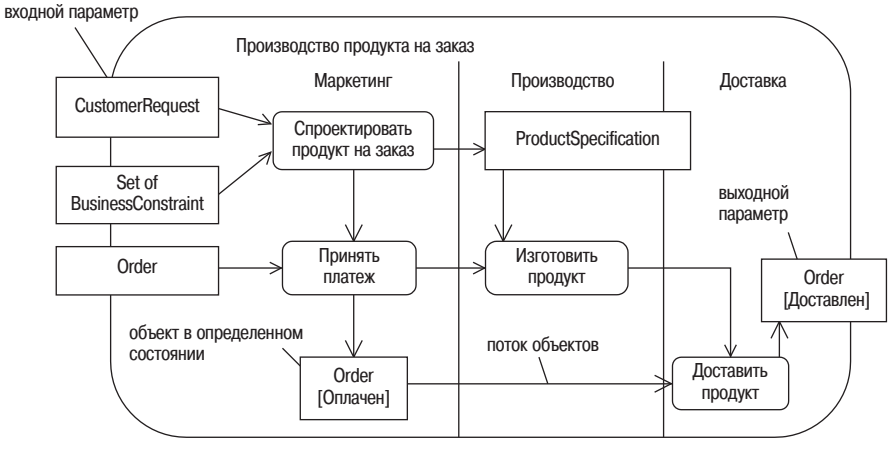


Рис. 14.23. *Деятельность Производство продукта на заказ имеет три входных параметра и один выходной*

К этому процессу предъявляется несколько бизнес-требований:

- Продукты проектируются на основании CustomerRequest. При этом создается ProductSpecification (спецификация продукта).
- При проектировании продукта учитываются все BusinessConstraint.
- Оплата производится только после завершения проектирования продукта.
- Изготовление продукта не может быть начато до тех пор, пока не будет получен платеж и создана спецификация продукта (объект ProductSpecification).
- Доставка не может осуществляться до тех пор, пока продукт не будет изготовлен.

Проанализируем данную деятельность.

1. Деятельность начинается, когда по входным потокам объектов действия Спроектировать продукт на заказ поступают CustomerRequest и множество BusinessConstraint. Действие принимает входящие объекты и выдает объект ProductSpecification.
2. Действие Принять платеж выполняется, когда получает управляющий маркер от Спроектировать продукт на заказ и объект Order по входному потоку объектов. Оно меняет состояние объекта Order на Оплачен и выдает его на свой единственный выходной поток объектов.
3. Затем поток управления переходит к действию Изготовить продукт. Оно принимает объект ProductSpecification, производимый в действии Спроектировать продукт на заказ, и предлагает маркер управления действию Доставить продукт.
4. Доставить продукт выполняется, когда от Изготовить продукт поступает маркер управления и объект Order находится в состоянии Оплачен.

Результатом действия будет объект Order в состоянии Доставлен. Этот объект Order является выходным параметром деятельности.

Как видите, мы довольно легко смогли выполнить бизнес-требования.

- Мы не будем пытаться Спроектировать продукт на заказ до тех пор, пока клиент не сделает запрос (CustomerRequest) и не будет получено множество бизнес-ограничений (Set of BusinessConstraint).
- Мы не можем Принять платеж, пока не будет сделан заказ (объект Order) и не будет завершено действие Спроектировать продукт на заказ.
- Мы не можем Изготовить продукт до тех пор, пока нет спецификации продукта (ProductSpecification) и не завершено действие Принять платеж (иначе говоря, пока продукт не оплачен!).
- Мы не можем Доставить продукт, пока он не изготовлен (не завершилось действие Изготовить продукт) и не оплачен (пока объект Order не перейдет в состояние Оплачен).

Этот пример иллюстрирует мощь диаграмм деятельности. Они могут кратко и точно моделировать сложные процессы.

14.10. Контакты

Деятельность, в которой много потоков объектов, может стать очень запутанной. Чтобы немного прояснить ситуацию, используйте контакты!

> Контакт – это объектный узел, представляющий один вход или выход из действия.

Контакт – это просто объектный узел, представляющий один вход или выход из действия. У входных контактов только одно входное ребро, а у выходных – только одно выходное ребро. Во всем остальном их семантика и синтаксис аналогичны объектным узлам. Однако из-за их небольшого размера всю информацию, например имя классификатора, приходится указывать вне контакта, но как можно ближе к нему.

На рис. 14.24 показана деятельность Войти в систему, имеющая два потока объектов. Деятельность начинается с действия Получить UserName. Его результатом является допустимый (valid) объект UserName. Следующее действие – Получить Password, на выходе которого получается допустимый объект Password. Деятельность Аутентифицировать объект User начинает выполнение, когда получает действительные объекты UserName и Password по своим входным потокам объектов. Осуществляется аутентификация пользователя, и деятельность завершается.

На рис. 14.25 показана та же деятельность, но представленная с использованием контактов. Как видите, нотация контактов выглядит более компактно, и диаграмма становится более аккуратной.

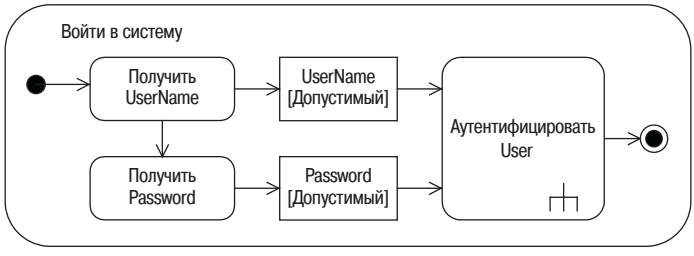


Рис. 14.24. *Деятельность с двумя потоками объектов*

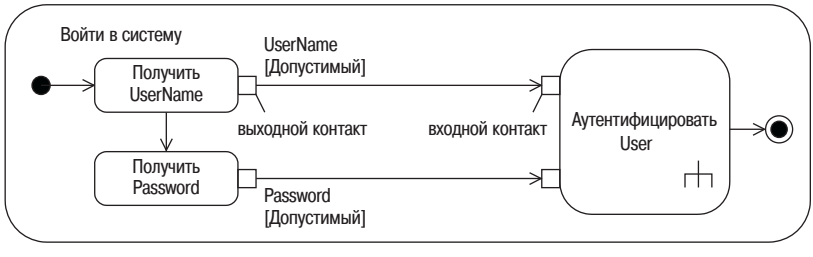


Рис. 14.25. *Та же деятельность с применением контактов*

Сравнивая рис. 14.24 и рис. 14.25, можно заметить, что объектный узел UserName эквивалентен комбинации выходного контакта действия Получить UserName и входного контакта действия Аутентифицировать объект User. Поэтому объектные узлы иногда называют *автономным контактом* (*stand-alone style pin*).

14.11. Что мы узнали

В этой главе было показано, что диаграммы деятельности можно использовать для моделирования множества различных процессов. Мы узнали следующее:

- Диаграммы деятельности – это ОО блок-схемы:
  - они используются для моделирования всех типов процессов;
  - диаграммы деятельности можно создать для *любого* элемента модели для описания его поведения;
  - хорошая диаграмма деятельности описывает один конкретный аспект поведения системы;
  - в UML 2 диаграммы деятельности имеют семантику технологии сетей Петри.
- Деятельности – это схемы, состоящие из узлов, соединенных ребрами.
  - Категории узлов:
    - узлы действия – элементарные единицы работы в рамках деятельности;

    - узлы управления – управляют потоком деятельности;
    - объектные узлы – представляют объекты, используемые в деятельности.
  - Категории ребер:
    - потоки управления – представляют поток управления деятельности;
    - потоки объектов – представляют поток объектов деятельности.
  - Маркеры перемещаются по сети (network) и могут представлять:
    - поток управления;
    - объект;
    - некоторые данные.
  - Движение маркеров по ребрам от начального узла к целевому узлу зависит от:
    - постусловий начального узла;
    - сторожевых условий ребра;
    - предусловий целевого узла.
  - Деятельности могут иметь предусловия и постусловия.
- Узлы действия.
  - Выполняются при одновременном поступлении маркеров по всем входным ребрам И удовлетворении всех предусловий.
  - После выполнения узлы действия предлагают маркеры *одновременно* на всех выходных ребрах, постусловия которых удовлетворены:
    - неявное ветвление.
  - Узел вызова действия:
    - инициирует деятельность – используется символ «грабли»;
    - инициирует поведение;
    - инициирует операцию.
  - Узел действия, посылающий сигнал (см. раздел 15.6).
  - Узел действия, принимающий событие (см. раздел 15.6).
  - Узел действия, принимающий событие времени, выполняется, когда временное выражение становится истинным:
    - некоторое событие (например, конец финансового года);
    - конкретный момент времени (например, 11/03/1960);
    - временной интервал (например, ожидать 10 секунд).
- Узлы управления:
  - начальный узел показывает, где начинается поток при вызове деятельности;
  - конечный узел деятельности заканчивает деятельность;

    - конечный узел потока заканчивает конкретный поток деятельности;
    - узел решения – поток направляется по исходящему ребру, сторожевое условие которого истинно:
        - может иметь стереотип «decisionInput»;
    - узел слияния копирует входные маркеры в единственное исходящее ребро;
    - узел ветвления разделяет поток на несколько параллельных потоков;
    - узел объединения синхронизирует несколько параллельных потоков:
        - может иметь {описание объединения}.
- Разделы деятельности – высокоуровневая группировка взаимосвязанных действий.
    - Разделы формируют иерархию, корнем которой является измерение.
- Объектные узлы представляют экземпляры классификатора.
    - Входящие и исходящие ребра – потоки объектов – представляют движение объектов.
    - Исходящие ребра объектного узла конкурируют за каждый исходящий маркер.
    - Объектные узлы работают как буферы:
        - { upperBound = n }
        - { ordering = FIFO } XOR { ordering = LIFO };
        - по умолчанию применяется { ordering = FIFO };
        - могут иметь стереотип «selection».
    - Объектные узлы могут представлять объекты, находящиеся в определенном состоянии:
        - должны соответствовать конечным автоматам.
    - Параметры деятельности – это объектные узлы, входящие в или исходящие из деятельности:
        - на диаграмме перекрывают рамку деятельности;
        - входящие параметры имеют один или более исходящих ребер, поступающих *в* деятельность;
        - исходящие параметры имеют один или более входящих ребер, поступающих *из* деятельности.
- Контакт – это объектный узел, представляющий один вход в или выход из действия или деятельности.