Содержание

Эту книгу мы посвящаем памяти Тома Джонсона (Tom Johnson), который побуждал нас вести семинары, давшие материал для этого издания. Безвременная кончина Тома в момент, когда подготовка рукописи уже близилась к завершению, опечалила всех, кто его знал. Нам будет его не хватать.

Предисловие

В первой своей книге, «Use Case Object Modeling with UML», мы отмечали, что разница между теорией и практикой состоит в том, что теоретически такой разницы быть не должно, а практически она существует. В данной работе мы попытались свести теорию объектно-ориентированного моделирования к некоему полезному подмножеству, которое можно легко изучить и применять для решения широкого круга задач. В основе книги – наш опыт преподавания (примерно с 1993 года) этого материала людям, работавшим над сотнями разнообразных проектов.

За два года книга выдержала уже пять изданий. Но, хотя наш труд получил лестную оценку, нам кажется, что работа еще не доведена до конца. Довольно часто на протяжении последних двух лет нам приходилось слышать, что «нужно больше примеров прецедентов и моделирования на UML». А поскольку мы пользовались первой книгой как основой для проведения семинаров, на которых применяли теорию к реальным проектам, стало ясно, что исключительно важная и плохо понимаемая тема – критический анализ или рецензирование (reviewing) моделей.

Поэтому, несмотря на то что в нашей первой книге приведен развернутый пример, мы убедили издательство Addison-Wesley выпустить продолжение, в котором очень подробно, шаг за шагом рассматриваем проектирование книжного Internet-магазина. Это позволило нам продемонстрировать многие распространенные ошибки и показать фрагменты моделей, в которых эти ошибки устранены. Мы выбрали именно такую разработку, поскольку в ней сконцентрированы особенности, присущие многим проектам в современном мире, пронизанном «Всемирной паутиной». Кроме того, данный пример мы использовали во многих семинарах, так что в нашем распоряжении оказалось множество учебных моделей на языке UML и, естественно, большая коллекция ошибок, которые часто совершают студенты.

Мы отобрали некоторые из наших «любимых» ошибок, которые повторялись снова и снова, и шаг за шагом проанализировали их. Затем добавили еще три главы, посвященные рецензированию требований, предварительного проекта и окончательного проекта.

От других изданий эта книга отличается, прежде всего, тем, что читатель учится исправлять допущенные при разработке недочеты.

Структура книги и соглашения

В главе 1 мы приводим обзор процесса ICONIX, а в следующих четырех более подробно рассматриваем четыре основных этапа этого процесса. Все эти главы строятся по одному принципу:

- в первом разделе описывается суть вопроса – моделирования предметной области (глава 2), моделирования прецедентов (глава 3), анализа пригодности (глава 5) или построения диаграмм последовательности (глава 7). Показано, как излагаемый материал соотносится со всем процессом. В каждой главе вам предлагается проработать отдельные фрагменты примера книжного Internet-магазина, а в завершение приводится диаграмма, в которой эти фрагменты объединены. В главе 3 мы рассмотрим «кусочки» десяти разных прецедентов; пять из них послужат предметом предварительного и детального проектирования в главах 5 и 7 соответственно. Фрагменты диаграмм классов, впервые появляющиеся в главе 2, также будут развернуты до текстового описания прецедента и полной диаграммы классов в главах 5 и 7;
- в следующем разделе описываются основные элементы рассматриваемого этапа. При этом в сжатом виде излагается материал соответствующей главы из книги «Use Case Object Modeling with UML», а также приводится некоторая дополнительная информация;
- далее мы рассматриваем десять самых распространенных ошибок, которые наши студенты допускали во время работы на семинарах. В каждую из трех глав, посвященных рецензированию, мы включили соответствующие перечни: десять ошибок при анализе пригодности, при построении диаграмм последовательности и т.д.;
- напоследок предлагается цикл из пяти упражнений для самостоятельной работы. В ходе их выполнения вы можете проверить степень усвоения материала.

У всех циклов упражнений есть общие черты:

- упражнения, посвященные моделированию предметной области и прецедентам, пронумерованы. В упражнениях на анализ пригодности

и диаграммы последовательности указывается название прецедента (чуть позже мы объясним смысл такого соглашения);

- в одном упражнении из каждой пары допущено три или четыре ошибки. Все они помечены на рисунке значком с надписью «Top 10». Цифра на этом значке показывает, какое правило нарушено;
- название упражнения, содержащего ошибки, сопровождается знаком 👎, а название правильного варианта, идущего следом, – знаком 👍;

В вашу задачу входит исправление ошибок, обнаруженных в первом упражнении из каждой пары.

Помимо этого в книге приняты следующие соглашения:

- смысловые выделения в тексте обозначены *курсивом*;
- на рисунках серым цветом выделены исправления ошибок, допущенных студентами и отмеченных значками с надписью «Top 10» на предыдущих рисунках.

Подведем итоги. В главе 2 описаны классы, которые будут использоваться в десяти выбранных прецедентах. Их фрагменты представлены в главе 3. В главах 5 и 7 приведены диаграммы, соответствующие пяти различным прецедентам. Идея состоит в том, чтобы пройти все стадии от частичного понимания каждого прецедента через диаграммы последовательности до ассоциированных с ними элементов детального проекта.

В главе 4 описана процедура рецензирования требований, в результате проведения которой следует удостовериться, что прецеденты и модель предметной области правильно описывают функциональные требования заказчика.

В главе 6 рассказывается о том, как выполнять рецензирование предварительного проекта, которое позволяет вам убедиться в наличии диаграмм пригодности для всех прецедентов (и в их согласованности с прецедентами), а также в том, что модель предметной области обладает развитым набором атрибутов, достаточным для описания всех имеющихся прототипов (и всех объектов, которые необходимы прецедентам, представленным в модели). Наконец, необходимо удостовериться, что команда разработчиков готова приступить к детальному проектированию.

Глава 8 посвящена рецензированию окончательного проекта. На этой стадии необходимо гарантировать, что реализация (*как*), представленная в детальном проекте, соответствует спецификации (*что*), описанной с помощью прецедентов. Кроме того, нужно убедиться, что детальный проект проработан в достаточной степени, и таким образом исключить серьезные проблемы при переходе к кодированию.

В главах 4, 6, 8 представлены общий обзор проблемы, детальный анализ и перечень десяти самых распространенных ошибок, но упражнения

отсутствуют. Цель, общая для всех трех видов рецензирования, состоит в том, чтобы удостовериться в согласованности различных частей модели в соответствии с принципами, продемонстрированными на правильных диаграммах в упражнениях из других глав.

В приложении содержится сводный отчет о модели нашего книжного магазина. Полную модель можно загрузить со страницы http://www.iconixsw.com/WorkbookExample.html. В приложении отражены все диаграммы, встречающиеся в книге, а полная модель включает также все детали пяти прецедентов, оставшихся нерассмотренными. Проработку этих прецедентов можно рассматривать как дополнительное упражнение. Полученные результаты вы можете сравнить с нашим решением, и мы настоятельно рекомендуем так и поступить.

Ну что, нравится перспектива? Нам не попадались другие книги такого же плана. Надеемся, что сумеем помочь вам в освоении объектного моделирования на основе анализа прецедентов.

Благодарности

Дуг благодарит отважную команду, работающую над проектом ICONIX, в особенности Андреа Ли (Andrea Lee) за сценарий компакт-диска, посвященного процессу ICONIX, из которого мы многое позаимствовали для главы 1, а также Криса Старчака (Chris Starczak), Джеффа Кантора (Jeff Kantor) и Эрина Арнольда (Erin Arnold). Еще Дуг благодарит Кендалла за то, что он, в конце концов, снисходил до просьб соавтора, говоря: «да, это *действительно* улучшит книгу» и «да, у нас есть время это добавить», и к тому же согласился с тем, что, поскольку буква P предшествует букве C, мнение г-на Розенберга имеет приоритет над мнением г-на Скотта[1].

Дуг и Кендалл признательны Полу Беккеру (Paul Becker) и всем сотрудникам издательства Addison-Wesley, включая Росса Венабля (Ross Venables) – он там уже не работает, но начинал этот проект, – которым удалось максимально сократить график запуска в производство, компенсировав тем самым задержки на стадии написания книги (все Кендалл виноват). Мы также выражаем благодарность рецензентам рукописи, особенно Марку Вудбери (Mark Woodbury), чьи язвительные замечания по поводу «дефрагментации» привели к тому, что книга стала, по нашему мнению, блестящей, а не просто «классной». И еще мы благодарим Грега Вилсона (Greg Wilson), который писал рецензию на нашу первую книгу для журнала «Dr. Dobbs' Journal». Идея ему понравилась, и он предложил написать продолжение. Если быть точными, вот его слова: «Никогда не думал, что выскажу

[1] Надо бы мне официально поменять имя на Скотт Кендалл, будет тогда знать! – Прим. соавтора.

такую претензию к книге, но... она слишком короткая! Наконец-то я нашел действительно полезное, приятное для чтения и применимое на практике описание методологии разработки, в основе которой лежит проектирование. И мне хочется видеть десяток, а то и поболее примеров на каждую тему. Если авторы напишут продолжение, обещаю, что хотя бы один покупатель будет».

И наконец, Кендалл благодарит Дуга за то, что он поднял искусство быть всем недовольным на такую высоту, что на этом фоне Кендалл стал выглядеть эталоном благодушия.

Дуг Розенберг
Санта Моника, штат Калифорния
Май 2001
dougr@iconixsw.com
http://www.iconixsw.com

Кендалл Скотт
Харрисон, штат Теннесси
Май 2001
kendall@usecasedriven.com
http://www.usecasedriven.com

Глава 1. Введение в ICONIX

Процесс ICONIX представляет собой нечто среднее между очень громоздким рациональным унифицированным процессом (Rational Unified Process – RUP) и весьма компактной методологией программирования eXtreme (XP). Процесс ICONIX, как и RUP, основан на прецедентах, но не характеризуется множеством его недостатков. В этом процессе тоже применяется язык моделирования UML (Unified Modeling Language), однако основное внимание уделяется анализу требований. Отметим еще раз, что ICONIX, по представлению Айвара Джекобсона (Ivar Jacobson), «основан на прецедентах», вот почему с помощью этого процесса создаются вполне конкретные и простые для понимания прецеденты, которые легко использовать для разработки системы.

ICONIX вобрал в себя лучшие стороны трех методологий, разработанных в начале 90-х годов Айваром Джекобсоном, Джимом Рамбо (Jim Rumbaugh) и Грейди Бучем (Grady Booch), называющими себя «три амиго». Мы пользуемся подмножеством языка UML, отобранным в результате проведенного Дугом анализа этих методологий.

В главе 32 руководства по языку UML, «The Unified Modeling Language User Guide», говорится: «80% всех задач можно промоделировать с помощью 20% UML». Однако в книге нет ни слова о том, что это за двадцать процентов. В подмножество UML мы включили ту базовую нотацию, которой должно хватить для большинства моделей. В соответствующих разделах мы поясняем, как можно использовать другие элементы UML и при каких обстоятельствах они уместны.

Один из наших любимых афоризмов гласит: «разница между теорией и практикой состоит в том, что теоретически этой разницы не должно быть, а практически она существует». В практической работе никогда не хватает времени для моделирования, анализа и проектирования. Руководство всегда требует быстрого перехода к кодированию, поскольку степень завершенности программных проектов традиционно оценивается по количеству написанных строк. Мы придерживаемся минималистского подхода, обращая особое внимание на область между прецедентами и кодом. Мы хотим показать, что должно происходить в той точке жизненного цикла проекта, в которой вы начинаете работу: у вас уже есть несколько прецедентов и необходимо заняться анализом и проектированием.

Наша цель – выявить минимальное подмножество UML (и методологии моделирования в широком смысле), которого было бы достаточно для работы над программным проектом. Мы уточняем наше определение такого «минимально достаточного» подмножества на протяжении восьми или даже девяти лет. Подход, описываемый в этой книге, выдержал проверку на сотнях проектов и может применяться в разнообразных предметных областях и при разных способах организации работы.

Краткий обзор процесса ICONIX

На рис. 1.1 показан главный вопрос, на который должен ответить процесс ICONIX.

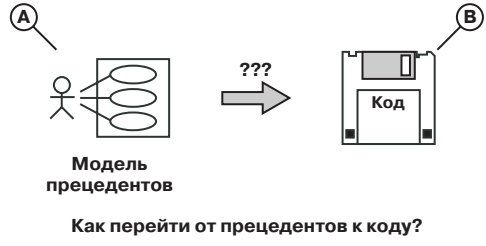

Рис. 1.1. Прецеденты, подлежащие кодированию

Наша задача – показать, как попасть из точки A в точку B за кратчайшее время. (На самом деле мы не собираемся проходить весь путь до получения готового кода, но того, что мы сделаем, будет достаточно для понимания смысла происходящего.) Можно считать, что точка A соответствует моменту, когда у вас сложилось представление о том, что должна делать система, и пора сформулировать кое-какие прецеденты. А в точке B имеется готовый, протестированный и отлаженный код, который делает то, что описано в прецедентах. Иными словами, код реализует требуемое поведение, определенное с помощью прецедентов. В этой книге основное внимание уделяется тому, как перейти от расплывчатых и неоднозначных фраз типа «думается, что это должно работать примерно так» к недвусмысленным, полным и четким формулировкам, на основе которых можно выстроить архитектуру, спроектировать надежную программу и написать красивый код, реализующий пожелания заказчика.

Мы начнем с кода и отправимся вспять, комментируя шаги, ведущие к достижению поставленной цели. Постараемся пояснить, почему предлагаемый набор этапов минимален и в большинстве случаев достаточен для осуществления перехода между прецедентами и кодом. На рис. 1.2 перечислены три исходных предположения:

- уже проведены работы по созданию прототипа;
- уже есть некоторые представления об интерфейсе пользователя;
- возможно, уже выявлены некоторые сценарии или прецеденты в системе.

Рис. 1.2. В начале пути

После этого можно приступать к анализу и проектированию. Проблема в том, как добраться из данной точки до готового кода. В начале пути у нас есть лишь один главный вопрос: «Что должна делать система?».

В объектно-ориентированных системах структура кода определяется классами. Поэтому, прежде чем приступать к кодированию, нужно выяснить, какие классы понадобятся. С этой целью следует нарисовать одну или несколько диаграмм классов. Для каждого класса необходимо описать все атрибуты, то есть данные-члены, и операции, которые представляют собой функции программы. Другими словами, мы должны идентифицировать все функции и удостовериться, что у нас есть данные, с которыми эти функции должны работать.

Нужно будет показать, как функции и данные инкапсулируются в классах. Для иллюстрации способов организации и взаимодействия классов будут использоваться диаграммы, а точнее, имеющиеся в UML диаграммы классов. В конечном счете мы намерены получить очень детальные диаграммы классов уровня проектирования. Говоря об *уровне проектирования*, мы имеем в виду такой уровень детализации, при котором диаграмма класса может служить шаблоном для создания кода; она должна точно отражать организацию будущей программы.

На рис. 1.3 показано, что создание диаграмм классов предшествует кодированию. Из рисунка видно, что классы на диаграмме и классы в коде соответствуют друг другу. Теперь возникает другая проблема – нужно

Рис. 1.3. Отображение диаграмм классов на структуру кода

как-то перейти от прецедентов к диаграммам классов уровня проектирования.

Один из самых сложных вопросов при разработке объектно-ориентированных программ заключается в *распределении поведения*, что подразумевает определение функций, из которых будет состоять программа. Нужно также решить, какому классу должна принадлежать каждая функция. Требуется распределить поведение всей системы, то есть отнести каждую функцию к определенному проектируемому классу.

В UML для этой цели лучше всего подходят *диаграммы последовательности* – идеальное средство для принятия решений о распределении поведения. Диаграммы последовательности разрабатываются отдельно для каждого сценария и показывают, какой объект отвечает за ту или иную функцию. На диаграмме последовательности видно, что экземпляры объектов взаимодействуют путем обмена сообщениями. Каждое сообщение приводит к вызову той или иной функции объекта-получателя. Именно поэтому она прекрасно выполняет задачу визуализации распределения поведения.

На рис. 1.4 расстояние между прецедентами и кодом стало заметно меньше. Осталось перейти от прецедентов к диаграммам последовательности.

Решения о приписывании поведения классам принимаются по мере создания диаграмм последовательности. В результате мы должны определить операции классов. При работе с такими программами визуального моделирования, как Rational Rose или GDPro, рисование направленных линий, соответствующих сообщениям, означает, по сути дела, создание операций

Рис. 1.4. Диаграммы последовательности помогают распределить операции (поведение) между классами

классов, изображаемых на диаграммах классов. Инструментарий автоматически распределяет поведение на основе диаграмм последовательности. По мере их рисования в диаграммах классов появляются операции.

Итак, вся сложность в том, как перейти от прецедентов к диаграммам последовательности. В большинстве случаев это непростая задача, так как прецеденты описывают систему на уровне требований, а диаграммы последовательности дают детальное представление с точки зрения проектирования. Этим-то и отличается наш подход от разработок других авторов. В других методиках много говорится и о прецедентах, и о диаграммах последовательности, но не уточняется, как преодолеть расстояние между ними. Именно преодоление пропасти между «что» и «как» – центральная тема в процессе ICONIX.

Следующий этап – переход от нечетких формулировок прецедента к очень точным и детальным диаграммам последовательности. Для этого служат *диаграммы пригодности*. Они упрощают процесс перехода.

Если вы изучали литературу по UML, то наверняка знаете, что первоначально диаграммы пригодности были включены в этот язык лишь частично. Они появились в работе Айвара Джекобсона и были утверждены в стандарте UML как дополнения. Это связано с историей о том, как Буч и Джекобсон решили объединить свои методики. Важность этого вида диаграмм для моделирования никоим образом не оспаривается.

В верхней части диаграммы последовательности размещается множество объектов, участвующих в данном сценарии. Перед тем как рисовать диаграмму последовательности, нам надо, в частности, обдумать, какие объекты

понадобятся. Неплохо было бы также иметь представление о выполняемых ими функциях. Работая над диаграммой последовательности, мы будем размышлять о том, как отобразить множество функций, реализующих требуемое поведение, на множество участвующих в сценарии объектов.

Целесообразно заранее знать о том, какие объекты нужны и какие функции они будут выполнять. Когда вы решаете эту задачу во второй раз, то результат получается намного лучше, чем в первый раз. Принятый нами процесс по существу совпадает с вариантом, предложенным в работе Айвара Джекобсона, и состоит в том, чтобы учесть наш предварительный проект, который изображается на диаграмме пригодности. Затем эти предварительные идеи эволюционируют до детального проекта, изображаемого на диаграмме последовательности. Диаграммы последовательности строятся для каждого известного сценария.

На рис. 1.5 мы добавили новый вид диаграмм в подмножество UML. Диаграммы пригодности были описаны в оригинальных спецификациях UML, но это определение включено в дополнительный документ под названием «Objectory Process-Specific Extensions». За прошедшие десять лет мы обнаружили, что перейти от прецедентов к диаграммам последовательности без этого механизма очень сложно. Применение диаграмм пригодности помогает решить проблему, характерную для многих коллективов, в которых тщательным образом обсуждают прецеденты, но ни на шаг

Рис. 1.5. Диаграммы пригодности служат связующим звеном между требованиями и детальным проектом

не продвигаются в проектировании программы. Воспользовавшись этим этапом, вы существенно упростите работу над проектом. Мы не изобретали анализ пригодности, но советуем своим читателям не забывать о нем. Анализ пригодности – незаменимое средство для осуществления перехода между требованиями и проектом.

Анализ пригодности занимает место между тем, что система должна делать, и тем, как этого достичь. Переход от одного к другому подразумевает одновременное выполнение нескольких видов деятельности. Сначала надо выявить объекты, о которых мы забыли на первом этапе проектирования системы. Возможно, прослеживая потоки данных на диаграммах пригодности, мы придем к выводу, что в классы надо добавить некоторые атрибуты. Немаловажно еще и то, что по результатам работы над диаграммой пригодности мы уточняем словесное описание прецедента.

Но один вопрос пока остается открытым. И относится он как раз к проблеме выявления объектов, позабытых при первой попытке спроектировать систему.

Чтобы научить людей правильной работе с прецедентами, мы часто повторяем магическую фразу: «описывайте использование системы в контексте объектной модели». Прежде всего это означает, что мы не будем тратить время на написание неоднозначных, туманных, абстрактных прецедентов, которые не содержат деталей, необходимых для вывода проекта программы. Задача авторов этой книги – научить читателей составлять прецеденты так, чтобы они были точными и недвусмысленными. При обсуждении прецедентов мы ставим перед собой вполне конкретную цель: из них должен получиться проект программы. Во многих книгах на данную тему прецеденты рассматриваются скорее как абстрактное средство изучения требований. Мы же придерживаемся другой точки зрения, поскольку наша цель – помочь читателям перейти от прецедентов к коду.

Начнем с так называемой модели предметной области. Это своего рода словарь основных абстракций, то есть самых важных существительных в пространстве задачи. Если, к примеру, предметной областью является электронная коммерция, как в этой книге, то, вероятно, в словарь войдут понятия «каталог» и «заказ на покупку». Существительные, которые описывают понятия из предметной области, будут называться *доменными объектами*. В самом начале анализа и проектирования мы создадим модель предметной области, в которой все доменные объекты будут изображены на одной большой диаграмме классов.

На диаграммах пригодности мы будем использовать *граничные объекты*. К ним относятся, например, мониторы, с помощью которых ведется работа с системой. В тексте наших прецедентов мы будем явно ссылаться

и на доменные, и на граничные объекты, описывать, как пользователь взаимодействует с монитором, а монитор – с доменными объектами, которые часто отображаются на базу данных, скрытую за объектно-ориентированной частью системы. Текст прецедента будет намного яснее и конкретнее, если придерживаться следующей рекомендации: описание способов работы с системой надо вести в контексте эволюционирующей объектной модели.

При моделировании предметной области мы хотим выявить наиболее важные абстракции, описывающие пространство задачи или, что то же самое, предметную область создаваемой системы. Для этого воспользуемся методологией OMT (Object Modeling Technique), разработанной Джимом Рамбо, в которой очень подробно рассматриваются разные полезные приемы, применяемые при создании модели предметной области.

Разница между нашим подходом и некоторыми другими, также основанными на прецедентах, заключается в том, что мы считаем принципиально важным начинать весь процесс с моделирования предметной области. Употребление в текстах прецедентов существительных, характерных для предметной области, то есть использование словаря модели, позволяет устранить неоднозначность терминологии. Особенно полезен такой подход в ситуации, когда над задачей работает коллектив, состоящий из нескольких групп специалистов, занятых описанием сценариев в разных частях системы. Если выработать единое соглашение о наиболее важных терминах, то можно избавиться от массы двусмысленностей в моделях прецедентов. Так, не возникнет расхождений в понимании того, что такое заказ на покупку, строка заказа и покупательская корзина. Все это определено на первом этапе, поскольку словарь терминов был создан еще до начала работы над прецедентами.

В терминологии UML модель предметной области – это, по сути дела, *диаграмма классов*. Обычно в этой модели мы опускаем большую часть деталей, в частности атрибуты и операции классов. Вот почему можно считать, что модель предметной области является сводной диаграммой классов. На самом деле это первый шаг к получению настоящей диаграммы классов, при котором мы стараемся представить систему в целом. Затем в процессе работы над прецедентами мы будем постепенно уточнять эту диаграмму и в конце концов получим детальную статическую модель системы.

На рис. 1.6 показано, как перейти от прецедентов и прототипов, изображенных в левой части, к диаграммам классов уровня проектирования и исходному коду в правой части.

Таким образом, наш подход не слишком замысловат. Мы пользуемся только четырьмя видами UML-диаграмм из девяти возможных. Как

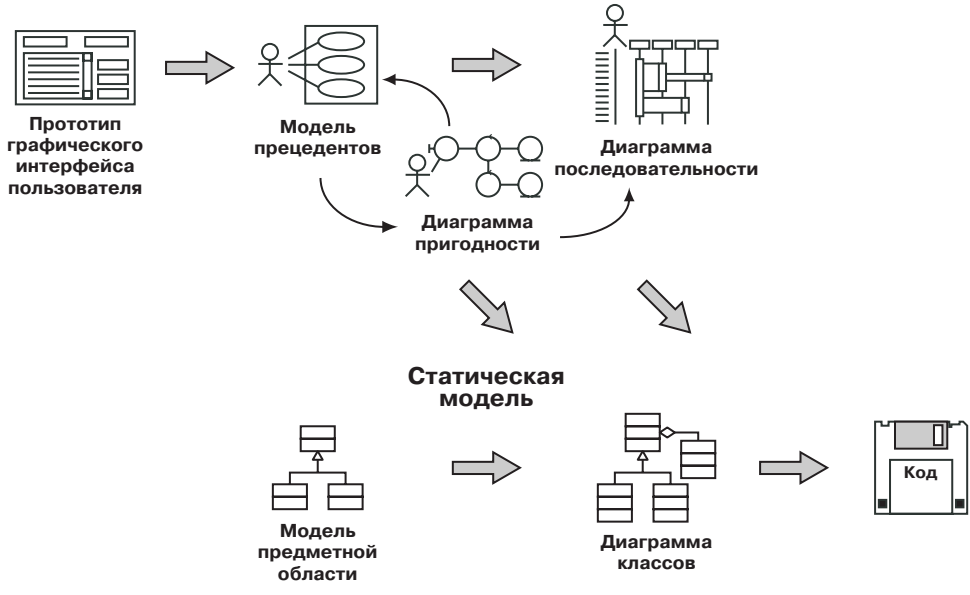

Рис. 1.6. Ссылка на доменные объекты по имени помогает избежать неоднозначности прецедентов

правило, для большинства проектов вполне хватает менее половины возможностей UML. Если вы сосредоточитесь на базовом подмножестве диаграмм, то сможете быстрее научиться созданию моделей средствами UML.

Мы будем отталкиваться от модели предметной области, которая представляет собой диаграмму классов аналитического уровня, – первое приближение к статической структуре системы. Затем начнем постепенно ее обогащать, стремясь к получению детального проекта. Диаграмма классов, изображенная в нижней половине рис. 1.6, – это статическое описание организации кода, тогда как прецеденты описывают динамическое поведение системы во время выполнения.

Свои первые представления о системе мы зафиксируем в виде статической модели, после чего займемся исследованием различных прецедентов. При проработке каждого из них мы будем что-то добавлять в диаграмму классов. После рассмотрения всех сценариев, которые должна поддерживать система, добавления необходимых деталей и повторного рецензирования результата должен получиться проект, удовлетворяющий всем требованиям. После этого можно приступать к написанию кода.

На рис. 1.7 представлена укрупненная схема процесса ICONIX. Этот рисунок повторяется в начале каждой главы нашей книги. Он состоит из

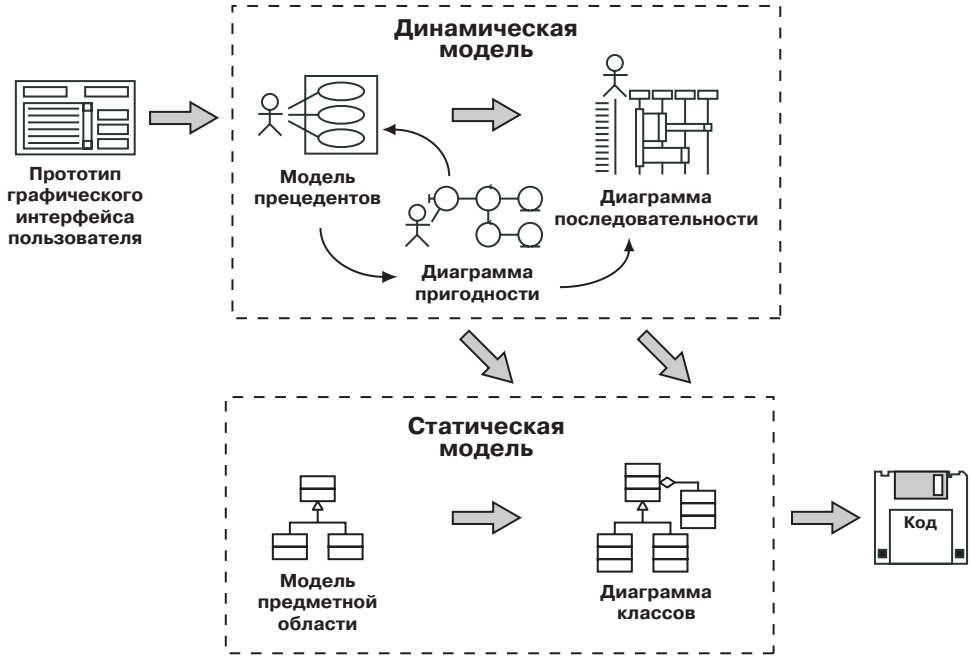

Рис. 1.7. Процесс ICONIX – простой подход к моделированию с помощью UML

двух частей: сверху изображена динамическая модель, описывающая поведение, а снизу – статическая, описывающая структуру.

Начнем с создания прототипов, которые достаточно просто нарисовать на экране. Затем, заручившись поддержкой потенциальных пользователей в правильности выбранного пути, нанесем прецеденты на диаграмму, стремясь показать все сценарии, которые должна выполнять система. Далее приступим к словесному описанию прецедентов. Тексты будут уточняться по ходу анализа пригодности. Важно добиться стабильного и правильного описания еще на этапе предварительного проектирования до того, как мы приступим к созданию детального проекта, изображаемого на диаграммах последовательности.

Многие разработчики жалуются на постоянное изменение требований. Некоторые расценивают это как предлог для преждевременного перехода к кодированию. Готовы поспорить, что эти люди, как правило, не выполняют анализа пригодности, который мог бы весьма ощутимо помочь в фиксации требований.

Разбивая работу над динамической моделью на три вышеописанных этапа, мы получаем шанс дважды отрецензировать описание поведения. Хочется надеяться, что ко времени второго рецензирования наше понимание

требований будет достаточно детальным и стабильным, чтобы лечь в основу проектирования.

Как видно из статической части картины, первое представление об объектах мы получаем исключительно из описания пространства задачи. Один длинный цикл последовательных уточнений выполняется при анализе динамического поведения системы. Мы стараемся во всех деталях понять, как должен выполняться отдельно взятый сценарий, а затем обновляем диаграммы классов. Наконец, возвращаемся назад и снова размышляем над тем, как должна вести себя система.

На следующем шаге мы уточняем структуру нашей программы. Этот подход, на 80% заимствованный из работы Айвара Джекобсона, позволяет разбить систему на части, определяемые границами прецедентов, после чего результаты анализа прецедентов используются с целью перехода на достаточно детальный для кодирования уровень модели.

Особенности процесса ICONIX

На рис. 1.7 изображен простой подход к разработке программного обеспечения, использующий минимальный набор UML-диаграмм и ряд приемов, которые позволяют быстро и эффективно перейти от прецедентов к коду. Этот подход обладает гибкостью и открытостью.

Хотелось бы подчеркнуть его особенности.

Во-первых, *простое использование UML*. Описанные в последующих главах шаги – это минимум, который, по нашему мнению, необходим и достаточен для успешной разработки объектно-ориентированного проекта. Взяв на вооружение лишь часть довольно громоздкого языка UML, коллектив сумеет справиться с «аналитическим параличом» на первых этапах работы над проектом.

Во-вторых, *возможность легкого отслеживания хода работы*. На каждом шаге мы так или иначе возвращаемся к требованиям. Ни на каком этапе процесс не может увести слишком далеко от потребностей пользователей. Говоря о легкости отслеживания, мы имеем в виду и то, что за эволюцией объектов при переходе от анализа к проектированию удается проследить без труда.

В-третьих, подход является *итеративным* и *инкрементным*, хотя эти термины мы рассматриваем не в традиционном смысле. При разработке модели предметной области, выявлении и анализе прецедентов выполняется несколько итераций. На протяжении жизненного цикла системы могут возникать и другие итерации. Статическая модель постепенно уточняется по мере выполнения последовательных итераций в работе над динамической моделью (составленной из прецедентов, результатов анализа

пригодности и диаграмм последовательности). Заметьте, впрочем, что мы не требуем выполнения формальностей и создания внутренних документов. Работа по уточнению развивается естественным образом по мере того, как коллектив расширяет свое понимание задачи.

В предисловии уже говорилось о том, что цель книги – продемонстрировать практическое применение процесса ICONIX на примере книжного магазина, работающего в режиме реального времени. При этом особое внимание уделено восприятию системы пользователем.

На протяжении последних десяти лет мы вели курсы по процессу ICONIX и с уверенностью можем говорить о его полезности и актуальности. Речь идет о поиске ответов на фундаментальные вопросы о системе:

- каковы пользователи системы (актеры) и что они пытаются сделать;
- что такое объекты «реального мира» (предметной области) и какие между ними ассоциации;
- какие объекты участвуют в каждом прецеденте;
- как объекты взаимодействуют друг с другом в каждом прецеденте;
- как учесть ограничения, налагаемые режимом реального времени;
- как будет выглядеть система на самом низком уровне.

Нам еще не встречались системы, которые не требовали ответов на эти вопросы (особенно на первые четыре), или системы, где ответы нельзя было бы получить с помощью описываемых в этой книге методов с использованием итеративного, инкрементного и оппортунистического («видишь ответ – подбери») подхода. Хотя строгое следование нашей методике предполагает выполнение шагов в определенном порядке, но это не столь уж важно. Многие проекты со временем прекращали свое существование из-за чрезмерно формализованного процесса, поэтому мы не являемся сторонниками такого подхода. Мы лишь говорим, что *отсутствие ответа на любой из сформулированных выше вопросов может подвергнуть разработку необоснованному риску*.

Базовые принципы

Мы утверждаем, что лучший способ сделать процесс привлекательным – научить ему как можно больше людей, рассказав о преимуществах поиска ответов на поднятые нами и другие подобные вопросы, а также об опасностях, связанных с игнорированием этих вопросов. Построение хорошей модели намного упрощается, если *вы твердо намерены искать ответы на фундаментальные вопросы о создаваемой системе и не увлечены второстепенными техническими вопросами моделирования*.

Заказчиками разработки являются как непосредственные участники процесса, так и руководящее звено. Мы представляем себе процесс как путь, по которому следует коллектив разработчиков, как дорогу, помеченную километровыми столбами или *вехами*, в конце которой находится высококачественный продукт.

Коллектив может выбирать разные пути в зависимости от способностей и предпочтений отдельных сотрудников. Но, каков бы ни был путь, необходимо пройти каждый этап. Это те точки, на которых работа становится видима руководству, – в ходе рецензирования промежуточных результатов. Соблюдение всех этапов не гарантирует получения качественного продукта, но существенно повышает шансы на это.

Мы полагаем, что в объектно-ориентированном процессе должны быть, по крайней мере, следующие вехи:

- выявление и описание всех сценариев использования системы;
- поиск повторно используемых абстракций (классов), участвующих в нескольких сценариях;
- анализ предметной области и идентификация принадлежащих ей классов. Другими словами, должны быть рассмотрены возможности повторного использования за пределами данной системы;
- проверка выполнения всех функциональных требований;
- тщательное обдумывание того, как требуемое поведение системы будет распределено между выявленными абстракциями с учетом принципов правильного проектирования, как то: минимизация степени связанности, максимизация плотности (cohesion), общность, достаточность и т.д.

Более того, к процессу предъявляются следующие требования:

- он должен быть достаточно гибким для адаптации к различным видам задач;
- он должен поддерживать реально применяемые способы организации труда (включая создание прототипов и инкрементную, итеративную разработку);
- он должен повышать производительность менее опытных членов коллектива, не отвлекая по мелочам более сведущих сотрудников;
- он должен демонстрировать руководству результаты работы, предшествующей кодированию, в достаточно понятном виде.

Краткое описание основных этапов процесса

На рис. 1.8–1.11 изображены основные этапы процесса ICONIX и связанные с ними вехи. Отметим, что первые три рисунка будут встречаться