Содержание

Базовые методики 5

Совершенная методика дает совершенный результат.

Сенсей Рик Виллемсен

Основа гибкого моделирования — это набор его методик. В своих проектах вы будете применять именно методики AM, основанные на ценностях и принципах AM. Как и принципы, методики AM делятся на базовые и дополнительные. Как уже говорилось в главе 3, вам придется принять базовые методики AM, чтобы иметь основание утверждать, что вы занимаетесь гибким моделированием. Конечно, вы можете успешно работать, используя лишь некоторые из этих методик, но лучше принять их все, если они соответствуют культуре производства вашей организации. Применение методик AM в вашей организации подробно рассмотрено в главе 28.

Базовые методики AM делятся на четыре категории, подробно описанные в главе 7: как объединены методики AM. Вот эти категории и соответствующие им базовые методики:

1. Итеративное и инкрементное моделирование:
   - применяйте нужный артефакт (артефакты);
   - создавайте несколько моделей параллельно;
   - переходите от одного артефакта к другому;
   - моделируйте постепенно.
2. Работа в группе:
   - моделируйте в группе;
   - активное участие заинтересованных лиц;
   - совместное владение;
   - демонстрируйте модели открыто.
3. Простота:
   - создавайте простое содержание;

- изображайте модели просто;
- используйте простейшие инструментальные средства.

4. Подтверждение правильности:
   - учитывайте тестируемость;
   - подтверждение модели кодом.

СОВЕТ: НИ ОДНА ИЗ ЭТИХ МЕТОДИК НЕ ЯВЛЯЕТСЯ НОВШЕСТВОМ

Когда вы прочитаете данную главу и главу 6, многие (если не все) методики AM наверняка покажутся вам знакомыми. Дело в том, что отдельно взятые методики AM не новы — они давно и успешно используются разработчиками. Новшеством является то, что я впервые объединил их и дал их совместное описание. Более того, они являются квинтэссенцией сотен лучших методик, которым следуют разработчики, и как вы увидите в главе 7, они объединены в синергетическое целое, которое я называю гибким моделированием.

Методики итеративного и инкрементного моделирования

AM определяет четыре методики, которые основаны на итеративном и инкрементном подходе к моделированию:

1. Применяйте нужный артефакт (артефакты).
2. Создавайте несколько моделей параллельно.
3. Переходите от одного артефакта к другому.
4. Моделируйте постепенно.

Применяйте нужный артефакт (артефакты)

Данная методика полностью соответствует выражению «Всему свое время». В данном случае вы хотите создать верную модель (модели), чтобы выполнить работу. Каждый артефакт, будь то схема состояний UML, базовый элемент Use Case, исходный код или диаграмма потока данных (DFD), имеет свои сильные и слабые стороны и поэтому применяется в зависимости от ситуации. Например, диаграмма деятельности UML (Рамбо, Джекобсон и Буч, 1999) на рис. 5.1 подходит для описания обработки деловой информации, тогда как статическую структуру вашей базы данных лучше изобразить в виде модели физического представления данных (Рейнгрубер и Грегори, 1994), как показано на рис. 5.2. Точно так же и диаграмма лучше, чем исходный код. Если рисунок сбережет вам тысячу слов, то модель, использованная при определенных обстоятельствах, может сэкономить вам 1024 кодовые строки (Карл Вигерс, 1999). Порой для эффективного анализа вариантов проекта лучше нарисовать со своими коллегами пару диаграмм, чем просиживать часами над разработкой примеров кода.

Для разработчика гибких моделей важно понять, когда следует и когда не следует применять тот или иной тип артефакта. Данная информация представлена

в приложении А и применяется в различных методах моделирования. Изучить каждый тип артефактов весьма трудно; проблема осложняется тем, что в вашем распоряжении может быть значительное количество артефактов.

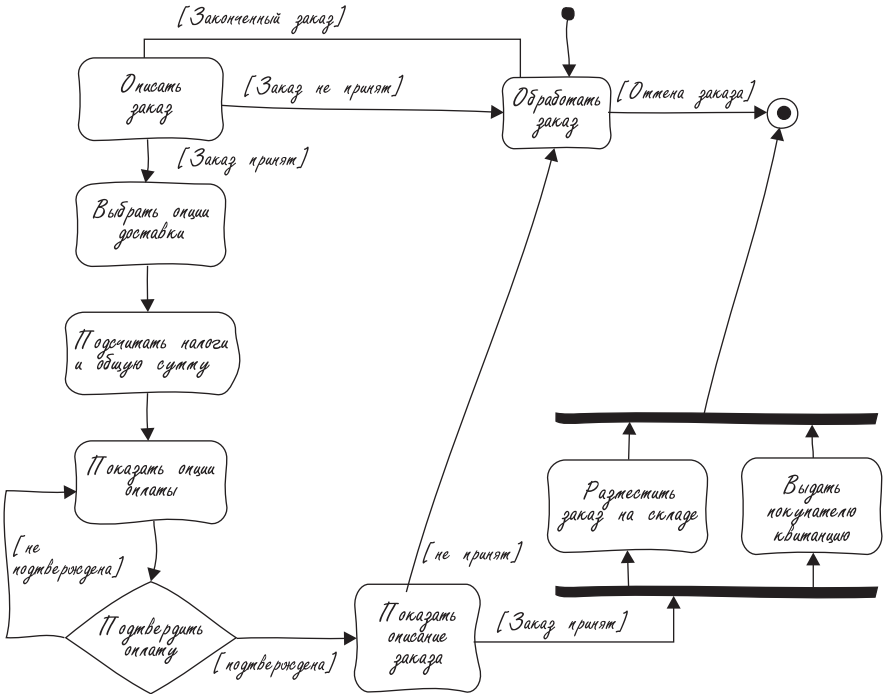

Рис. 5.1. Диаграмма деятельности UML для обработки заказа

СОВЕТ: НАЧНИТЕ С ИЗУЧЕНИЯ ПОДМНОЖЕСТВА АРТЕФАКТОВ

Поражены количеством артефактов, описанных в приложении А? Не беспокойтесь, это нормально. AM имеет обратную сторону. Оно требует от вас научиться применять множество артефактов, а на это уходит много времени и сил. Для начала следует остановиться на изучении основного подмножества, а затем изучать новые приемы по мере необходимости. Такие методологии, как ICONIX (Розенберг и Скотт, 1999) и Catalysis (Д'Суза и Вилс, 1999), рекомендуют несколько артефактов и описывают их применение на практике. В случае, если вы обнаруживаете недостающие аспекты методологии, вы можете пополнить их подходящими артефактами.

Создавайте несколько моделей параллельно

Любая модель имеет свои плюсы и минусы, поэтому ни одна модель не может удовлетворить ваши требования полностью. Например, вы рассматриваете требования и решаете разработать элемент Use Case (Джекобсон, 1992; Кокбёрн, 2001а),

как показано на рис. 5.3; сущностный прототип UI, как показано на рис. 5.4 (Константайн и Локвуд, 1999); CRC-модель (Каннингем и Бэк, 1989), как на рис. 5.5. Элемент Use Case описывает, как размещается заказ; прототип UI определяет требования к экрану или странице для поддержки ввода заказа; карточки CRC содержат концептуальную информацию о сфере деятельности.

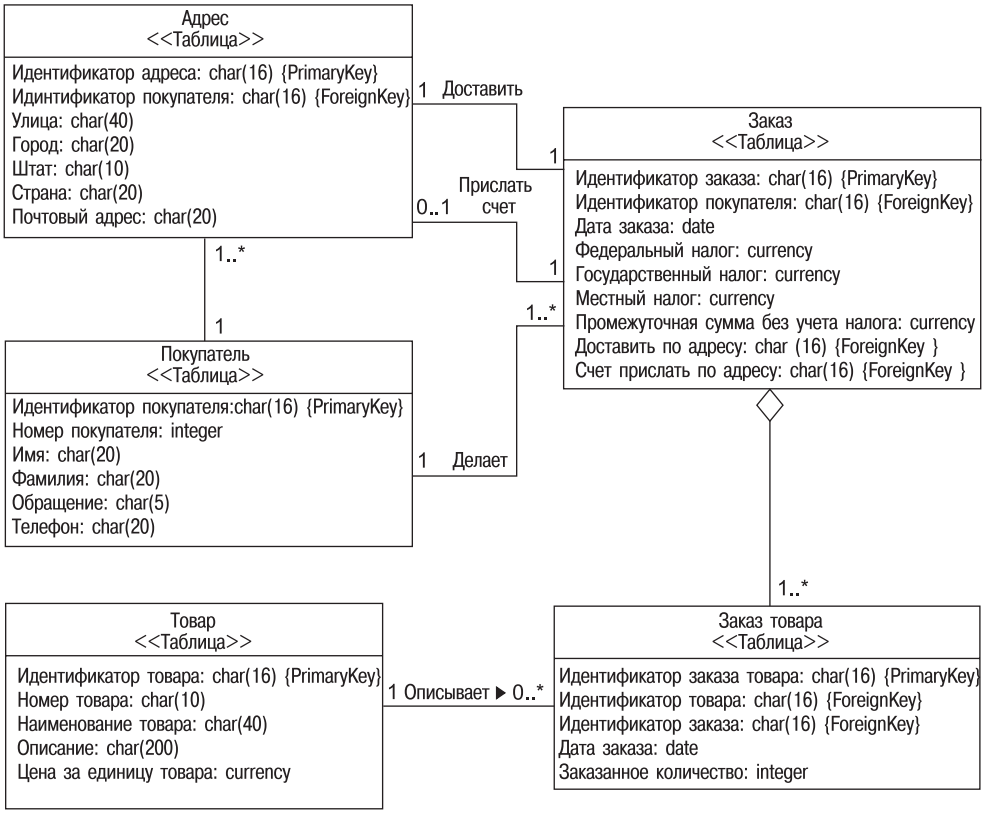

Рис. 5.2. Запуск модели физического представления данных для SWA Online

Выясняя требования заинтересованных лиц, вы вносите необходимые усовершенствования в каждую модель, заключая информацию в тот артефакт, где эта информация будет наиболее эффективна. То же происходит, например, когда вы разрабатываете ПО на Java и обнаруживаете, что вам нужно разработать диаграмму классов UML для определения структуры ПО, диаграмму схем состояний UML для изучения внутренней деятельности сложного класса, диаграмму последовательности UML для определения способа обеспечения логики потока в пределах элемента Use Case (см. рис. 5.1) или бизнес-правила, и модель физического представления данных для понимания структуры вашей реляционной базы данных (см. рис. 5.2). Используя данную методику в сочетании с методикой «Переходите

от одного артефакта к другому» (см. следующий раздел), разработчики гибких моделей быстро осознают, что они работают более продуктивно, разрабатывая несколько моделей одновременно, в отличие от случаев, когда они сосредоточиваются на одной модели.

1. Элемент Use Case начинается с того момента, когда покупатель решает разместить заказ.
2. Покупатель ищет товар посредством элемента Use Case «Поиск товара (товаров)».
3. Покупатель добавляет товар к своему заказу.
4. Покупатель указывает количество заказываемого товара.
5. Система считает промежуточную сумму , умножая цену товара на его количество.
6. Покупатель повторяет шаги с 2 по 5 по мере формирования заказа.
7. Покупатель заканчивает добавлять товары к заказу.
8. Покупатель дает информацию по доставке и оплате, включая имя, телефон и физический адрес.
9. Система считает промежуточную сумму всего заказа, путем сложения промежуточных сумм отдельных групп товара.
10. Система считает налоги, которыми облагается заказ, в соответствии с деловым регламентом *Расчет налога на заказ*.
11. Система считает действующие скидки, в соответствии с деловым регламентом *Расчет скидок на заказ*.
12. Система отображает действующие налоги и скидки.
13. Система считает общую сумму заказа путем добавления действующих налогов к промежуточной сумме заказа и вычитания скидкок.
14. Система отображает итог заказа.
15. Покупатель подтверждает правильность заказа.
16. Система планирует выполнение заказа (см. элемент Use Case «Выполнение заказа»).
17. Система выдает покупателю подтверждение получения заказа с его описанием.

Рис. 5.3. Основные действия при размещении заказа

Интересное свойство данной методики заключается в том, что она дискредитирует два распространенных антипаттерна в информационных технологиях. К сторонникам первого, который я называю «разработчик одного артефакта», относятся те, кто специализируется в разработке одного вида комплектующих узлов. Пример — те, кто желает только писать коды, кодировщики низкого уровня, которые считают, что важнее исходного кода нет ничего на свете, и разработчики моделей данных, которые думают, что по степени важности ничто не сравнится с данными.

На самом деле такой узкий подход не эффективен при современном уровне развития ПО. Я советую вам не только накопить как можно больше опыта в области создания различных типов артефактов (исходный код и модели данных — отличные варианты), но и обзавестись хотя бы поверхностными знаниями о различных методах. Другими словами, заполните свой багаж знаний до предела. Другой

антипаттерн — «сеансы моделирования одного артефакта». Наиболее распространенным примером работы над одним типом модели являются сеансы моделирования элементов Use Case. Для меня важен сеанс моделирования требований, а вот сеанс моделирования элементов Use Case просто не имеет смысла в условиях гибкого моделирования.

Рис. 5.4. Сущностный прототип UI, демонстрирующий требования к экрану/странице

Кому-то первое время будет сложно создавать несколько моделей одновременно, особенно тем людям, которые предпочитают выполнять одну задачу в единицу времени, и тем, кто является сторонником использования антипаттерна «разработчик одного артефакта». Если ваша организация не знакома с гибким моделированием, то будьте готовы оказать помощь таким сотрудникам. Я обнаружил, что первым шагом к преодолению этих проблем является работа с несколькими моделями одновременно, следуя методике «Моделируйте в группе», также описанной

в данной главе. Для этого необходимо сотрудничество между людьми, успешно освоившими данную методику, и теми, кто испытывает проблемы с ее применением. Я обычно предлагаю своему партнеру известный ему артефакт в сочетании с другим, также известным ему артефактом, который он считает незначительным. Я бы выбрал такие сочетания: элемент Use Case и диаграмма последовательности UML, диаграмма классов UML и модель физического представления данных. Цель такого упражнения — на практике убедить партнера, что этот подход действительно эффективно работает.

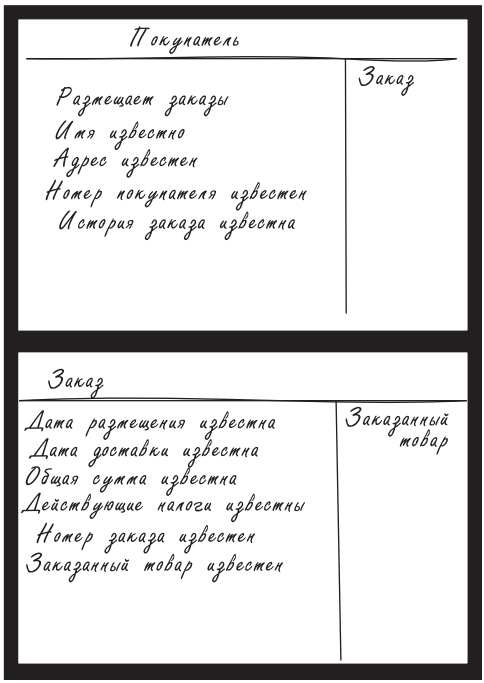

Рис. 5.5. Две карточки CRC для SWA Online

Переходите от одного артефакта к другому

Возможно, вы работаете над элементом Use Case, карточкой CRC, диаграммой последовательности или даже исходным кодом, и вдруг заходите в тупик. Это показатель того, что вам на некоторое время следует заняться другим артефактом. У каждого артефакта есть слабые и сильные стороны; каждый артефакт хорош для определенного типа работы. Если вы начинаете испытывать трудности при работе с артефактом (например, вы работаете с элементом Use Case и обнаруживаете, что не в состоянии описать бизнес-логику), это значит, что пора переключиться на другой артефакт. При этом вы немедленно «сниметесь с мели», потому что над

другим артефактом работа пойдет быстрее. Более того, за счет изменения точки зрения вы вдруг можете обнаружить, что успешно работаете над тем, что еще недавно вызывало у вас затруднения. То есть кроме того, что вы вновь обретете способность двигаться дальше, вы еще сделаете важную часть работы, а это и есть основная концепция гибкого моделирования.

Наибольшая сложность данной методики — выбрать подходящий объект, на который вы переключаетесь, когда зашли в тупик. Тут вам поможет опыт, но если его не хватает, то на этот случай в приложении А имеется список подходящих вариантов. Например, если вы работаете над основным элементом Use Case, то вы можете переключиться на сущностный прототип UI, модель CRC, бизнес-правило, системный элемент Use Case или вариант изменений.

Моделируйте постепенно

Принцип «Изменения должны быть небольшими» показывает, что инкрементный подход к разработке является основой AM. Как вы читали в главе 1, это действительно основной аспект гибкой разработки ПО вообще (Альянс гибких методологий, 2001b). Главная идея в том, что вы делите весь объем работы на небольшие части, которые выполняются постепенно, в идеале с интервалом в несколько недель или месяцев. Тем самым вы добиваетесь большей гибкости за счет ускорения передачи ПО в распоряжение пользователя, а также имеете конкретную и быструю обратную связь с пользователем во время всей работы над проектом. Если вы применяете инкрементный подход к разработке, то будете применять его и к моделированию.

При инкрементном подходе к разработке вы понемногу моделируете, понемногу пишете код, понемногу проводите испытания, затем предоставляете заказчику небольшую часть работы. Больше не нужно следовать схеме «Сначала все спроектируем» и тратить недели, а то и месяцы на создание моделей и документации. Теперь большинство сеансов моделирования (обычно импровизированные совещания для обсуждения одного-двух вопросов) длится 10–20 минут. Разработчики гибких моделей выполняют моделирование небольшими порциями, позволяющими оперативно вернуться к работающему ПО, как того требует принцип «Главная цель — программное обеспечение». Более продолжительные сеансы моделирования, иногда длящиеся несколько дней (обычно в начале проекта), могут иметь место, но они являются скорее исключением, чем правилом. Чем дальше вы продвигаетесь, не получая конкретной обратной связи, тем больше шансов обнаружить, что вы моделируете не то, что от вас требуется, то есть напрасно тратите силы. Далее, если вы подолгу сосредоточиваетесь на вашей модели (моделях), вы рискуете нарушить методики «Изображайте модели просто» и «Создавайте простое содержание», также описанные в данной главе.

Самое сложное в этой методике — прекратить моделирование, как только вы достигли определенной цели. Возможно, вам захочется моделировать то, с чем придется работать завтра, на следующей неделе или через месяц. Вы подсознательно хотите предусмотреть все наилучшим образом, но остановитесь! Наберитесь

смелости и решайте сегодняшние проблемы сегодня; скажите себе, что вы в состоянии разобраться с завтрашними проблемами завтра (Бек, 2000).

Методики эффективной работы в группе

Гибкое моделирование определяет четыре методики, способствующие эффективной работе в группе и налаживанию общения как внутри группы, так и с заинтересованными лицами:

1. Моделируйте в группе.
2. Активное участие заинтересованных лиц.
3. Совместное владение.
4. Демонстрируйте модели открыто.

Моделируйте в группе

Разработка ПО во многом подобна плаванию — и то и другое опасно делать в одиночку. Моделируя с определенной целью, вы часто ловите себя на том, что делаете это для того, чтобы в чем-то разобраться или донести свои идеи до окружающих или чтобы создать общее представление о проекте. Этим лучше заниматься в группе, поскольку вам понадобится вклад нескольких эффективно сотрудничающих людей. Вы не раз убедитесь, что над созданием базового набора моделей, имеющих критическое значение для проекта, должна работать вся группа разработчиков. Например, при разработке метафоры или архитектуры системы требуется, чтобы моделирование выполняла вся группа. Так находится согласованное простейшее решение, удовлетворяющее всю группу. В большинстве случаев для этого следует обсудить проблему с несколькими людьми. Никто не запрещает вам нарисовать простую схему, которую вы можете обдумать самостоятельно, но после этого вам следует с кем-нибудь поделиться своими идеями, чтобы понять, в верном ли направлении вы двигаетесь. Одна голова — хорошо, а две лучше. Применение данной методики наладит общение между людьми, занятыми в проекте, поможет выработать общую терминологию, увеличит шансы успешного выполнения работы и предоставит членам группы возможность обмениваться опытом между собой.

Применение данной методики может оказаться сложным на первых порах в организациях, которые придерживаются политики «разделяй и властвуй», когда объем работ делится на отдельные части, выполняемые разными людьми, или если в организации процветает жесткая конкуренция, что не способствует сотрудничеству. Чтобы устранить эту проблему, вам придется донести до окружающих выгоды совместной работы; возможен следующий подход: «Новая методика подразумевает новые методы работы, так что давайте попробуем и посмотрим, что из этого выйдет». Вам также понадобится рабочая зона (см. главу 11), которая поможет совместной работе в группе. Иногда достаточно белой доски, расположенной

на стене чьей-нибудь загородки или в специально отведенном месте для работы группы.

Активное участие заинтересованных лиц

Заинтересованным лицом может быть любое лицо, являющееся прямым пользователем, косвенным пользователем, руководителем пользователей, старшим руководителем, оператором, сотрудником группы поддержки пользователей, испытателем, разработчиком, работающим над другой системой, которая является частью разрабатываемой системы или взаимодействует с ней, или работником технического обслуживания, которого так или иначе коснулась разработка и/или развертывание проекта. Поскольку речь идет о гибком моделировании, я намеренно исключил разработчиков, работающих над проектом, из списка заинтересованных лиц. Хотя разработчики тоже заинтересованы в проекте, термин «заинтересованное лицо» будет обозначать всех таковых лиц, кроме разработчиков, работающих над проектом. То есть говоря о разработчиках и заинтересованных лицах, я имею в виду разные группы людей. В качестве альтернативы могу предложить такой вариант: «разработчики, заинтересованные в проекте» и «не-разработчики, заинтересованные в проекте». Нравится? Вот и я так считаю.

Методика гибкого моделирования «Активное участие заинтересованных лиц» близко соприкасается с методикой «Моделируйте в группе». Более того, это расширенная методика экстремального программирования (XP), которая называется «Локальный заказчик» (Бек, 2000). Данная методика заключается в присутствии при разработке лиц (обычно это пользователи или их представители), которые имеют полномочия и способности для предоставления информации по разрабатываемой системе, а также могут принимать уместные и своевременные решения относительно требований и их приоритета. Такой минимальный уровень сотрудничества необходим для успешной разработки ПО, хотя этого явно недостаточно, если речь идет об организации, где сотрудники занимаются не созданием работающей системы, а интригами. Для успеха проекта часто требуется более активное вовлечение заинтересованных лиц. Старшее руководство должно поддерживать проект как открыто, так и в частном порядке; операторы и обслуживающий персонал должны активно сотрудничать с вашей группой, чтобы подготовить производственную среду к принятию вашей системы; другие группы разработчиков должны сотрудничать с вами для интеграции усилий; сотрудники сопровождения должны разобраться в технологиях и методах, используемых в вашей системе. Данная методика основывается на принципах «Быстрая обратная связь» и «Открытые и доброжелательные контакты».

Само собой, чтобы добиться успеха, все заинтересованные в проекте лица должны активно сотрудничать с вашей группой. Вот несколько выводов, сделанных с помощью данной методики:

- Пользователи должны быть готовы поделиться с группой своими знаниями бизнеса и принимать уместные и своевременные решения относительно цели проекта и приоритетов требований.

- Чтобы старшее руководство всячески содействовало осуществлению вашего проекта, оно сначала должно понять преимущества и дополнительную ценность технологии и методов, используемых вашей группой, понять причину, по которой ваша группа их использует, и понять результаты их использования. При наличии этих знаний действия старшего руководства на внутренней политической арене вашей организации наверняка станут более эффективными и своевременными. Старшее руководство не получит этих так необходимых знаний, читая еженедельный отчет по проекту или посещая ежемесячное совещание по управлению проектом. Для этого руководитель должен потратить время на изучение того, чем он руководит, и принимать активное участие в разработке вашей системы.
- Операторы и обслуживающий персонал должны приложить все возможные усилия для того, чтобы понять вашу систему и используемые в ней технологии. Обслуживающему персоналу придется изучить все тонкости системы; он должен работать с вашей системой в процессе разработки, а ваша группа должна будет обучать персонал. Далее, операторы должны в совершенстве овладеть установкой и управлением вашей системой. Можно включить одного или более инженеров по эксплуатации в состав группы разработчиков или использовать проектные ресурсы на необходимую подготовку операторов. В любом случае группа эксплуатации и группа обслуживания должны активно сотрудничать с группой разработчиков.
- Другие группы, занятые в проекте, должны сотрудничать с вами в том случае, если ваша система должна интегрироваться с другими системами. Например, ваша система должна обращаться к действующей базе данных, взаимодействовать с оперативной системой, работать с файлами данных, созданными внешней системой, или обеспечивать извлечение данных XML для других систем. Без активного участия других разработчиков интеграция часто оказывается сложной, если возможной вообще. Представьте, насколько сложным будет обращение к действующей базе данных, если владельцы БД отказываются предоставить соответствующую информацию. Помните, что общение — это улица с двусторонним движением; вы также должны делиться информацией о вашей системе с другими группами.
- Сотрудники сопровождения должны работать с вашей группой, чтобы ознакомиться с вашей системой. Если вы намереваетесь частично или полностью поручить сопровождение вашей системы другим сотрудникам, то стоит пригласить программистов, имеющих опыт по сопровождению и модернизации существующих систем, чтобы разгрузить вашу группу. В этом случае ваша группа должна сотрудничать с этими программистами, чтобы они могли принять у вас систему. Даже если кто-то из числа ваших постоянных сотрудников продолжает работать над этой задачей, вы должны постараться передать знания новым членам группы. Хорошо, если при освоении системы ваши опытные сотрудники будут руководить новичками или просто будут работать с ними в паре.

Совместное владение

Каждый может работать над любой моделью, а в идеальной ситуации и над любым артефактом в проекте, если возникнет такая необходимость. Например, если я нарисую на белой доске диаграмму потока данных (DFD), то такой подход имеет несколько преимуществ. Во-первых, чем больше людей участвует в разработке артефакта, тем больше шансов определить его потенциальное несоответствие требованиям, согласно принципу «Быстрая обратная связь». Во-вторых, это предоставляет людям большие возможности накопления опыта при разработке различных типов моделей, пополняя их багаж знаний и тем самым повышая их эффективность как разработчиков гибких моделей. Это согласуется с принципом «Учитесь друг у друга», потому что члены группы могут наблюдать за работой других и за счет этого повышать свой уровень. В-третьих, это подавляет искушение сказать что-нибудь типа «Твоя модель никуда не годится», потому что если кто-нибудь замечает проблему, то должен решить ее, а не жаловаться. В-четвертых, это снижает риск присвоения определенных артефактов. Никто не может сказать: «Эта модель классов — мое детище, а значит, с ней все в порядке», потому что никакой отдельный артефакт не принадлежит кому бы то ни было в отдельности. В-пятых, это способствует тому, что члены группы лучше понимают систему и налаживают общение в группе. Также это снижает необходимость написания пространной документации и необходимость полагаться на отдельного сотрудника или подгруппу. В руководстве проектом есть следующее понятие — «число кирпичей». С его помощью оценивается минимальное количество сотрудников, при отсутствии которых (например, им упадет кирпич на голову) проект придет в упадок. Число кирпичей, равное единице, является серьезной проблемой; если число кирпичей превышает или равняется количеству людей в вашей группе, то это идеальная ситуация. Совместное владение увеличивает число кирпичей для вашей группы.

Данная методика может с трудом приниматься в организациях, где личность ставят выше группы и/или назначают членам группы узкие задачи. Сотрудники должны признать, что артефакты, над которыми они работают, являются групповой собственностью. Они также должны научиться работать над различными типами задач, то есть не должно получаться так, что один работает только над пользовательским интерфейсом, а другой создает отдельную подсистему или системный интегрирующий код.

Демонстрируйте модели открыто

Вам следует открыто демонстрировать свои модели, например с помощью того, что часто называют «стеной моделирования» или «стеной чудес» (Готтсдинер, 2001). Тем самым в вашей группе поддерживается принцип «Открытые и доброжелательные контакты», потому что все текущие модели доступны членам группы, а также заинтересованным лицам, от которых вы ничего не скрываете. Ваша стена моделирования находится там, где вы выставляете свои модели на всеобщее обозрение; стена моделирования должна быть доступна вашей группе разработчиков и остальным заинтересованным лицам. Ваше моделирование может быть физическим (например, специально для этого предназначенная белая доска, на

которой вы рисуете диаграмму архитектуры, или просто место, где вы приклеиваете скотчем распечатку модели физического представления данных) или виртуальным (например, страничка во внутренней сети, которая обновляется с помощью сканированного изображения).

Следующим преимуществом данной методики является то, что с ее помощью вы показываете заинтересованным лицам выполненную вами работу — вот она, у них перед глазами. Это особенно удобно, когда вы разрабатываете простые модели впервые и боитесь, что вы не так продуктивны (в отношении количества), как ранее; демонстрация нескольких простых моделей в доступном для всех месте имеет огромное значение для оценки вашей работы.

Данную методику сложно применять в организациях, где невозможно найти свободное место на стене; или если группа работает в сравнительно людной зоне, где ходят клиенты или даже конкуренты, а вы не хотите раскрывать рабочий процесс; или там, где на отделку стен были потрачены значительные средства (например, юридическая фирма со стенами, обшитыми дубовыми панелями, которые вы не хотите испортить). Если препятствием является одна из этих причин, возможно, вы захотите перевести свою группу в другое помещение. Далее, препятствием может явиться нежелание делиться информацией с посторонними людьми. В этом случае я советую вам набраться смелости и все-таки принять эту методику.

Упрощающие методики

AM выделяет три методики, которые позволяют упростить процесс моделирования:

1. Создавайте простое содержание.
2. Изображайте модели просто.
3. Используйте простейшие инструментальные средства.

Создавайте простое содержание

Содержание ваших моделей (моделей требований, анализа, архитектуры или проектных моделей) должно быть максимально простым, но при этом все-таки соответствовать требованиям заказчика. Смысл в том, что не стоит вносить дополнительные аспекты в вашу модель без особой надобности. Например, изображенная на рис. 5.6 диаграмма классов UML (Рамбо, Джекобсон и Буч, 1999) не отражает свойств и операций классов, которые предположительно должны определить те, кто будет заниматься программированием этих классов (надеюсь, ими окажутся люди, нарисовавшие эту диаграмму), когда до этого дойдет дело. Это соответствует методике экстремального программирования «Простой дизайн» (Бек, 2000). Эта методика также применима к таким не-моделируемым артефактам, как исходный код, план проекта или пользовательская документация.

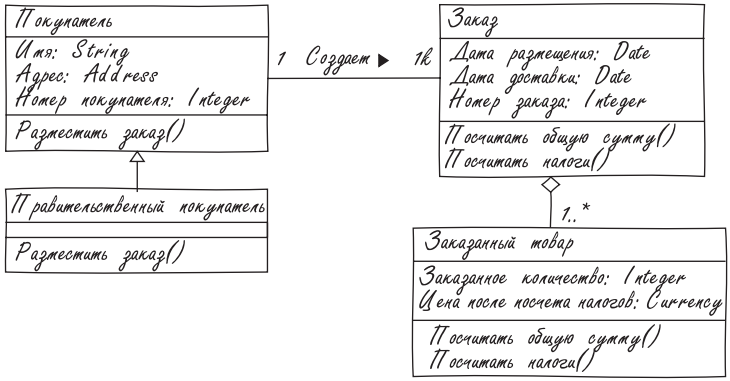

Рис. 5.6. Диаграмма классов UML

Как узнать, проста ли ваша модель по содержанию? Для определения простоты модели я применяю следующие показатели, основанные на советах Кента Бека (2000) по простейшему проектированию:

- Модель показывает все, что вы хотите показать. Другими словами, она отвечает своей цели.
- Модель не должна содержать дублирующей информации.
- Модель должна состоять из минимального количества элементов.

При освоении данной методики камнем преткновения может стать ваша склонность чрезмерно усложнять модели, но ее возможно преодолеть, если вы признáете за собой этот недостаток и будете сдерживать себя и других сотрудников вашей организации, старающихся добиться успеха с помощью сложных моделей и документации. Часто вы будете испытывать на себе давление организаций (обычно тех, которые используют предопределенные процессы и не знакомы с гибким подходом), требующих от вас соответствия существующим стандартам оформления документации и моделирования. В этом случае вам придется рассказать коллегам о своем подходе и объяснить причины, которые вами движут. Можно дать им почитать эту книгу, а еще лучше, если вы предложите им приобрести собственный экземпляр!

Изображайте модели просто

Когда вы рассматриваете возможные типы диаграмм, которые вы можете применить (диаграммы UML, диаграммы пользовательского интерфейса, модели данных и т. д.), вы быстро поймете, что чаще всего требуется всего лишь подмножество из доступных типов диаграмм. Например, простая модель (модель классов, отображающая основные обязанности классов и отношения между ними), показывающая основные характеристики, которые вы пытаетесь понять, оказывается зачастую достаточной. Вы, конечно, можете написать вспомогательный код, все

операции «получатели» и «установщики», в соответствии с кодовыми стандартами, но будет ли от этого польза? Для гибкого разработчика — очень маленькая.

Хотя эта методика дополняет методику «Создавайте простое содержание», идеи обеих методик являются ортогональными. Методика «Создавайте простое содержание» фокусируется на содержании модели, в то время как методика «Изображайте модель просто» рассматривает способы представления моделей. Ниже перечислены советы по созданию простых диаграмм (Амблер, 2002):

- Избегайте пересекающихся линий.
- Избегайте кривых линий.
- Избегайте диагональных линий.
- Избегайте рамок разного размера.
- Избегайте большого количества рамок (не больше 7±2).
- Избегайте ненужных деталей.

Используйте простейшие инструментальные средства

Большинство моделей может быть нарисовано на белой доске, бумаге или даже салфетке. Чтобы сохранить эти диаграммы, можно сфотографировать их цифровой камерой или просто перенести на бумагу. Вы ознакомились с несколькими диаграммами, созданными при помощи простейших средств. Рисунок 5.4 был создан при помощи планшета, стикеров (бумаги для записей на липкой основе) и маркеров; рис. 5.1 был нарисован на белой доске, а карточки CRC (Класс-Обязанность-Сотрудничество) на рис. 5.5 — при помощи обычных карточек и ручки. Использование простейших средств допустимо, потому что большинство диаграмм нужны лишь для временного использования. Истинное значение диаграммы в том, что вы можете обдумать проблему в процессе ее создания, но как только проблема решена, диаграмма утрачивает ценность. Поэтому белая доска и маркер — лучшие средства моделирования. Используйте графические утилиты для создания диаграмм, предназначенных для показа важным представителям заинтересованных лиц, а средства (утилиты) моделирования используйте тогда и только тогда, когда они стоят затраченных на них усилий (например, при генерации кода). Создавая простые модели, вы моделируете лишь для того, чтобы разобраться с той или иной проблемой; поэтому модели часто имеют временное значение и, скорее всего, не будут нужны после ее решения, следовательно, нет необходимости применять сложные средства.

В главе 10 эта методика рассматривается подробно. Хотя я ясно указал на это в главе 1, я сделаю это еще раз: AM не имеет ничего против CASE-средств. Если наиболее эффективным использованием ваших ресурсов является приобретение CASE-средства, то не теряйте времени и извлеките из этого средства все возможности. На рынке представлено множество различных CASE-средств, но внимания заслуживают лишь немногие. Если простых средств достаточно — используйте их.

Эту методику будет сложно применять в организациях, где привыкли создавать модели при помощи сложных средств. Многие разработчики не считают моделированием то, что создается без использования дорогостоящих CASE-средств, и им трудно согласиться с тем, что стопка учетных карточек может быть не менее эффективной. Лучший выход из ситуации — вовлечь их в использование простейших средств, обучая их таким методам, как моделирование CRC и основных прототипов пользовательских интерфейсов (UI).

Методики подтверждения правильности работы

AM выделяет две методики, относящиеся к тестированию и подтверждению правильности вашей работы:

1. Учитывайте тестируемость.
2. Подтверждение модели кодом.

Учитывайте тестируемость

Когда вы моделируете, постоянно задавайте себе вопрос: «Как мы будем это тестировать?» Если вы не можете тестировать программу, которую создаете, не стоит ее и создавать. Современные процессы разработки ПО включают тестирование и подтверждение качества на всех этапах выполнения проекта, а кое-где даже практикуют создание тестов до написания программы, например в методике XP «Сначала тест, потом программа» (Бек, 2000). Гибкие разработчики проводят тестирование на раннем этапе и делают это часто, чтобы убедиться в качестве выполняемой работы. Самое сложное в применении данной методики — научиться постоянно думать о том, как вы собираетесь тестировать свою работу.

Подтверждайте модель кодом

Модель — это абстрактное понятие, которое должно точно отражать аспекты того, что вы создаете. Вам следует написать соответствующий код, чтобы определить, будет ли она работать. Вы разработали схему странички HTML для приема информации об адресации счета? Закодируйте ее и покажите пользовательский интерфейс вашим пользователям для получения обратной связи. Вы разработали диаграмму последовательности UML, на которой представлена логика применения сложного бизнес-правила? Напишите код тестирования, программный код, а затем проведите тесты, чтобы убедиться в том, что все сделано правильно. Не забывайте о том, что при использовании итеративного и инкрементного подходов, ставших нормой в большинстве проектов по разработке ПО, моделирование — одна из многих задач, которые вы будете выполнять. Занимайтесь моделированием, выполняйте кодирование и проводите тестирование (наряду с остальными операциями). Хотя целью AM является моделирование, не забывайте о других аспектах разработки.

Есть ряд помех на пути к применению этого метода.

- Лучше всего он работает, когда люди, занимающиеся моделированием, также пишут программный код. Это предполагает, что гибкие разработчики должны обладать рядом умений и навыков. В главе 12 я пишу о том, что гибкие разработчики должны быть специалистами широкого профиля. Помните, что разработка ПО больше, чем моделирование.
- Многие разработчики придерживаются принципа «Сначала все смоделируем», поэтому на моделирование уходит больше времени, чем нужно, а кодирование отходит на второй план.
- Многие разработчики привыкли работать следующим образом: создание модели, рецензирование и доработка, а затем — кодирование. В гибком моделировании вы моделируете и кодируете небольшими порциями. В таком случае необходимость рецензирования отходит на второй план, потому что вы подтверждаете свои модели кодированием, являясь их потребителем, на которого непосредственно повлияют любые проблемы моделей, и это стимулирует выполнение качественной работы с самого начала.