Введение

Ведущие программисты считали моделирование предметных областей (*domain modeling*) и проектирование архитектуры программных объектов на этой основе (*domain design*) ключевой методологией разработки сложных программных систем на протяжении вот уже более двадцати лет. Тем не менее за это время было написано удивительно мало об основных задачах (*что делать*) и методах (*как делать*) этой области знания. Но несмотря на отсутствие четкой формализации, в профессиональной среде постепенно "вызрела" система взглядов и подходов, которую я называю *предметно-ориентированное проектирование* (*domain-driven design*, *DDD*).

За прошедшее десятилетие мне довелось участвовать в разработке нескольких сложных систем, относящихся к различным областям техники и бизнеса. В своей работе я старался применять самые лучшие приемы проектирования и разработки, созданные лидерами объектно-ориентированного программирования. Некоторые из моих проектов имели успех, другие же оказались неудачными. Все успешные проекты объединяло одно свойство: для них была построена подробная модель предметной области, которая итеративно улучшалась в ходе разработки, пока не становилась органичной частью проекта.

В этой книге предпринимается попытка построить систему понятий, в рамках которой было бы удобно принимать проектные решения, а также создать технический язык для коммуникации при разработке архитектурной модели предметной области (т.е. в том процессе, который называют *domain design*). Общепризнанные практические приемы соседствуют здесь с моими собственными выводами и результатами. С помощью такой системы коллективы разработчиков программного обеспечения, работающие со сложными предметными областями, смогут выработать строгий, формализованный подход к построению своих проектов.

Разные судьбы трех проектов

Я хорошо помню три проекта, явившиеся прекрасными примерами того, как реализация предметной области влияет на конечный результат. Хотя в итоге всех трех проектов на свет появились полезные программы, только в одном из них удалось достичь по-настоящему дерзких целей и создать такое программное обеспечение, которое продолжает успешно развиваться, удовлетворяя растущие запросы работающей с ним организации.

Один из проектов "сошел со стапелей" очень быстро, и в результате появилась простая и удобная интернет-система брокерской торговли. Ее разработка велась довольно бессистемно, но все обошлось, потому что относительно простые программы можно писать и без тщательного проектирования их архитектуры. После первоначального успеха надежды на будущее были большими. Именно тогда меня и хотели привлечь к работе над второй версией программы. Когда я поближе ознакомился с проектом, я обнаружил, что у разработчиков отсутствует не только модель предметной области, но и общий язык для всего проекта, а плохо продуманная архитектура стала настоящим бременем. Руководители проекта не согласились с моей оценкой состояния дел, и я отказался от этой работы. Но год спустя коллектив разработчиков "застрял" на месте и оказался неспособен реализовать вторую версию. Пусть у них не все было ладно и по технической части, но все-таки поражение они потерпели именно в борьбе с прикладной моделью. И вот уже самая первая версия программы преждевременно перешла в разряд дорогостоящих в обслуживании реликтов.

Реализация сложных систем требует серьезного подхода к моделированию архитектуры предметной области. В начале своей трудовой деятельности мне посчастливилось принять участие в проекте, в котором работе с предметной областью действительно придавали большое значение. Этот проект, относящийся к области по крайней мере столь же сложной, как и первый, тоже начинался с создания несложной программы для институциональных трейдеров. Но в данном случае первоначальная версия получила самое интенсивное продолжение. На каждом новом этапе разработки открывались захватывающие перспективы по объединению и усовершенствованию возможностей предыдущих версий. Группа разработчиков сумела среагировать на потребности трейдеров с должной гибкостью и эффективностью. Этим взлетом они были непосредственно обязаны модели предметной области — глубоко и последовательно проработанной, поэтапно улучшаемой, четко отраженной в коде. По мере того как разработчики вникали в предмет, совершенствовалась и модель. Улучшалось понимание не только между самими разработчиками, но и между группами экспертов и разработчиков. И вместо того чтобы накладывать на авторов тяжкое бремя поддержки и доработки программы, архитектура проекта становилась, напротив, все более удобной для модификации и расширения.

К сожалению, для столь полного успеха проекта недостаточно только серьезного подхода к моделированию. Один из проектов, которым я занимался в прошлом, имел весьма амбициозную цель — создание глобальной корпоративной системы управления на основе предметной модели. Но после нескольких лет разочарований ожидания стали самыми заурядными. У группы разработчиков были в наличии хорошие технические возможности и хорошее понимание предметной области; они также уделили должное внимание модели. Но неправильно организованное распределение труда разработчиков привело к отделению моделирования от реализации, и в результате при проектировании архитектуры не учитывалась вся глубина проделанного анализа. По крайней мере, объекты прикладной модели были спроектированы недостаточно строго, чтобы обеспечить их взаимодействие в сложных приложениях. Несколько итераций доработки не внесли в код никаких улучшений из-за разброса в уровне квалификации разработчиков — они, в целом, не владели неформальным стилистическим и техническим аппаратом для создания модельных объектов, которые бы одновременно представляли собой практичные, работоспособные программы. Шли месяцы, процесс разработки все усложнялся и запутывался, группа потеряла целостное видение системы. После нескольких лет усилий проект все-таки выдал “на-гора” небольшую полезную систему, но в итоге группа разработчиков отказалась как от своих прежних амбиций, так и от самой модели.

О сложностях

В процессе создания проекта возможны различные препятствия, — например, бюрократизм, нечеткая постановка задач, недостаток ресурсов. Но именно стратегия архитектурного проектирования главным образом определяет потенциальную сложность будущего программного продукта. Когда сложность выходит из-под контроля, разработчики перестают понимать свое изделие достаточно хорошо, для того чтобы вносить в него исправления или расширять его возможности без особого труда и опасений. В противоположность этому, качественно спроектированная архитектура создает возможность грамотно использовать сложность системы.

Некоторые влияющие на проектирование программ факторы имеют технический характер. Много труда уходит на проектирование сетей, баз данных и другие технические аспекты программного обеспечения. О решении таких проблем написаны многие тома. Целые армии программистов оттачивают свои умения и осваивают каждое новшество в этой сфере.

И все же главная сложность во многих программных продуктах — вовсе не техническая. Ее истоки — в самой предметной области, в той отрасли знания, которой занимается пользователь программы. Если сложную структуру предметной области не отразить в архитектурном проекте приложения, то не будет иметь никакого значения качественная проработка технической инфраструктуры. Для успешного проектирования программ необходимо систематически подходить к этому основному аспекту программирования.

В этой книге развиваются две основополагающие идеи.

1. В большинстве программных проектов основное внимание должно быть сосредоточено на логической структуре предметной области и взаимосвязях в ней.
2. Архитектура сложного программного обеспечения должна основываться на модели предметной области.

Предметно-ориентированное проектирование — это и образ мышления, и система приоритетов, призванная ускорить разработку программных продуктов, которые применяются в сложных областях деятельности. Для достижения этой цели в настоящей книге представлен обширный набор приемов, подходов и принципов проектирования программного обеспечения.

Процессы проектирования и разработки

Книги по проектированию и книги по методике реализации программных систем... Почему-то эти книги нечасто ссылаются друг на друга. И та, и другая область знания достаточно сложна. Эта книга — о проектировании, но, по моему мнению, проектирование и реализация неотделимы друг от друга. Архитектурные концепции следует успешно воплощать в программах, иначе они выродятся в пустую теорию.

Когда кто-нибудь изучает приемы проектирования, от возникающих возможностей у него начинает кружиться голова. Но суровая реальность практического проекта быстро излечивает от головокружения. Новые архитектурные идеи не стыкуются с технологиями, которые приходится использовать. Непонятно, когда можно отказаться от того или иного аспекта грамотного проектирования в интересах скорости разработки, а когда нужно упорно искать архитектурно и стилистически “чистое” решение. Бывает, что разработчики говорят друг с другом на абстрактном уровне о применении принципов архитектурного проектирования, но все-таки для них более естественно говорить о конкретных вопросах реализации. И хотя эта книга — о проектировании, я собираюсь в случае необходимости прорываться через искусственно созданную границу прямо в методологию разработки. Это поможет нам рассмотреть принципы проектирования в практическом контексте.

Эта книга не привязана к какой-то конкретной методологии, но в целом ориентирована на новое семейство гибких методик разработки (*agile development processes*). В частности, всегда подразумевается использование в проектах двух подходов, являющихся необходимыми предпосылками для применимости всего того, о чем говорится в книге.

1. **Итеративный характер разработки**. Итерационная разработка программ пропагандируется и практикуется уже не первое десятилетие. Этот подход лежит в основе гибкой методологии. По ней, а также по экстремальному программированию (*extreme programming, XP*) есть много хорошей литературы. Можно назвать такие книги, как *Surviving Object-Oriented Projects* (Cockburn, 1998) и *Extreme Programming Explained* (Beck, 1999).
2. **Тесное взаимодействие между разработчиками и специалистами в предметной области**. Особенность предметно-ориентированного проектирования такова, что в его

процессе в модель закладывается огромное количество знаний, отражающих глубокое понимание предметной области и концентрацию на ее ключевых концепциях. Это требует сотрудничества между теми, кто владеет предметом, и теми, кто умеет писать программы. Так как разработка носит итеративный характер, сотрудничество должно продолжаться на протяжении всего срока существования проекта.

Экстремальное программирование, авторами которого являются Кент Бек (Kent Beck), Уорд Каннингем (Ward Cunningham) и др. (см. *Extreme Programming Explained* [5]) — наиболее известная из гибких методологий разработки программ, с которой лично я работал больше всего. На протяжении этой книги для конкретизации изложения я буду использовать именно ее в качестве основы для рассмотрения взаимодействия между проектированием и разработкой. Приведенные принципы можно легко адаптировать к другим гибким методологиям разработки.

В последние годы среди разработчиков назрел "бунт" против громоздких методик разработки, перегружающих проект ненужной статичной документацией и чрезмерно детализированным планированием. В противоположность этому такие методики, как экстремальное программирование, делают акцент на способности гибко справляться с изменениями и неопределенностью.

В экстремальном программировании признается важность принятия проектных решений, но категорически отрицается жесткое заблаговременное планирование с полной детализацией проекта (что называют *upfront design*). Вместо этого прилагаются большие усилия для обеспечения коммуникации и способности быстро менять направленность проекта. Имея такую способность, разработчики могут на любом этапе проекта выбирать простейшее из работоспособных решений, а затем постоянно выполнять рефакторинг, внося много мелких проектных улучшений и постепенно приходя к архитектуре, которая будет удовлетворять действительные потребности клиента.

Такой минимализм уже послужил "лекарством" против некоторых излишеств энтузиастов архитектурного проектирования. Многие проекты в свое время утонули в море документации, толку от которой практически не было. Их подвел "мандраж" — члены группы разработчиков так боялись несовершенств архитектуры, что вообще никуда не двигались. Надо было что-то менять.

К сожалению, некоторым из этих методических идей грозит неправильная интерпретация. У каждого программиста есть свое понятие о том, что такое "простейшее" решение. Непрерывный рефакторинг — это последовательность мелких изменений проектной архитектуры. Разработчики, не имеющие четких ориентиров, могут произвести на свет базу кода, плохо понятную и не поддающуюся доработке, — т.е. прямую противоположность слову "гибкость" (*agility*), которое дало название методике. Да, страх перед непредвиденными требованиями часто ведет к излишней детализации архитектуры, но попытки избежать жесткого планирования могут перейти в другую фобию — страх вообще перед любой глубокой проработкой модели.

Фактически, методика экстремального программирования лучше всего подходит разработчикам "с обостренным чутьем" в области архитектурного проектирования. Эта методика предполагает, что архитектуру можно улучшить рефакторингом и что происходить это должно часто и быстро. Но принятые в прошлом проектные решения могут как облегчить, так и затруднить сам процесс рефакторинга. Методика экстремального программирования призвана активизировать коммуникацию в группе разработчиков, но на сам процесс коммуникации также влияют решения, принятые при моделировании и проектировании.

В этой книге проектирование архитектуры и разработка программного продукта переплетаются; на примерах показано, как предметно-ориентированное проектирование и

гибкая методология разработки помогают друг другу. Если в контексте гибкой методологии еще и принимается серьезный подход к моделированию предметной области, то это позволяет значительно ускорить разработку. Взаимосвязь между методикой разработки и моделированием предметной области делает такой подход более практичным, чем любые оторванные от жизни "чистые" архитектурные проекты.

Структура книги

Эта книга разделена на четыре большие части.

Часть I, "Модель предметной области в работе", содержит основные постановки задач предметно-ориентированного проектирования. От этих постановок будут непосредственно зависеть практические приемы в дальнейших разделах. Существует множество подходов к разработке программного обеспечения, поэтому в части I даются определения терминов, а также очерчивается тот неявный контекст, который порождается применением модели предметной области для целей коммуникации и проектирования.

Часть II, "Структурные элементы предметно-ориентированного проектирования", обобщает опыт практического объектно-ориентированного моделирования предметных областей, сводя его к набору основных структурных элементов-"кирпичиков". В этой части основное внимание уделяется преодолению пропасти между абстрактными моделями и конкретными работающими программами. Использование стандартных шаблонов привносит в проектирование упорядоченность, а члены группы разработчиков лучше понимают работу друг друга. Стандартные шаблоны также позволяют обогатить общий язык терминологией, с помощью которой все разработчики могут обсуждать модели и проектные решения.

Но главная задача этой части книги — выделить такие решения, которые позволяют модели и ее программной реализации не отставать друг от друга, а также повышать эффективность как одной, так и другой. Это требует большого внимания к проработке отдельных элементов. Тщательная проработка в мелком масштабе дает разработчику прочный фундамент, на котором можно будет реализовывать методики моделирования из частей III и IV.

Часть III, "Углубляющий рефакторинг", посвящена переходу от отдельных "кирпичиков" к задаче сборки их в практические модели, дающие в конце концов конечный результат. Вместо того чтобы сразу изложить секретные принципы проектирования, в этой части книги делается акцент на исследовательском процессе. Действительно ценные модели не возникают в один день; они требуют глубокого понимания предметной области. А такое понимание приходит не иначе, как в ходе работы: вначале архитектура реализуется в первом приближении, основанном пока на примитивной и наивной модели, а затем она снова и снова подвергается преобразованиям. Всякий раз, когда группа разработчиков делает скачок в понимании предмета, модель преобразуется для учета новых знаний, после чего код подвергается рефакторингу для отражения усовершенствований модели и раскрытия ее возможностей в приложении. Поэтапное погружение в предметную область (наподобие послойной очистки луковицы) дает результат в виде прорыва на новый уровень моделирования, после чего следует серия кардинальных изменений в архитектуре.

Творчество исследователя неформально по своей природе, но не обязательно хаотично. В части III рассматриваются принципы моделирования, которые помогают принимать решения на этом пути, а также методы, способствующие выбору направления в поиске.

Часть IV, "Стратегическое проектирование", посвящена ситуациям, возникающим в сложных системах, больших организациях, а также при взаимодействии с внешними и устаревшими системами. В этой части книги рассматривается три принципа, примени-

мых к системе в целом: соблюдение контекста, дистилляция (в смысле выделения сути, концентрации на логике приложения) и крупномасштабный подход. Стратегические проектные решения принимаются как на уровне групп разработчиков, так и на уровне взаимодействия между группами. Стратегическое проектирование делает возможной реализацию задач из части I в более крупном масштабе — масштабе системы или приложения на уровне обширной и разветвленной корпоративной сети.

На протяжении всей книги изложение иллюстрируется не упрощенными "игрушечными" задачами, а примерами из реальных проектов по разработке программного обеспечения.

Значительная часть книги написана в виде набора "архитектурных шаблонов". Читатели смогут понять материал и не задумываясь об этом, но тем из них, кто интересуется стилем и форматом таких шаблонов, рекомендуется прочитать приложение.

Дополнительные материалы можно найти на сайте `http://domaindriven-design.org`, в том числе примеры кода и дискуссии в профессиональном сообществе.

Для кого предназначена эта книга

Эта книга написана главным образом для разработчиков объектно-ориентированного программного обеспечения. Для большинства членов рабочих групп, занимающихся такими проектами, окажутся полезными те или иные части книги. Наиболее ценной она, по-видимому, будет для тех людей, которые работают над проектами прямо сейчас, пытаясь по ходу дела освоить те или иные приемы и методы, а также для уже имеющих большой опыт в такого рода проектах.

Для чтения книги требуется некоторое знакомство с объектно-ориентированным моделированием. В примерах используются диаграммы UML и код Java, так что полезно владеть хотя бы основами этих языков (хотя доскональное их знание и не требуется). Рассуждения о процессе разработки будут иметь более глубокий смысл для знающих экстремальное программирование, но даже и без этого знания материал должен быть вполне понятен.

Разработчикам программ среднего уровня квалификации, которым кое-что известно об объектно-ориентированном проектировании (в том числе из пары-тройки прочитанных книг), эта книга поможет заполнить пробелы в знаниях и составить представление о реальном месте объектного моделирования в проектах по разработке программ. Такие разработчики смогут научиться применению сложных приемов моделирования и архитектурного проектирования при решении реальных проблем.

Высококлассным профессиональным разработчикам программного обеспечения будет интересна всеобъемлющая система работы с предметной областью, предложенная в этой книге. Систематический подход к проектированию поможет также техническим руководителям вести свои рабочие группы к успеху. А единая терминология, употребляемая в книге, позволит квалифицированным разработчикам эффективно общаться с коллегами.

Материал книги изложен последовательно — ее можно читать с начала до конца или же с любой главы. Читатели с различной профессиональной подготовкой могут выбрать свой порядок чтения, но я бы рекомендовал всем начать с введения к части I, а также с главы 1. Помимо этого, основными можно считать главы 2, 3, 9 и 14. Те, кто уже до некоторой степени владеет материалом, смогут определить главные моменты, читая заголовки и текст жирным шрифтом. Высококвалифицированный читатель может пропустить части I и II и перейти к, вероятно, наиболее интересным для него частям III и IV.

В дополнение к основному кругу читателей, кое-что полезное из этой книги могут почерпнуть также аналитики и руководители проектов, частично занятые в технической

их части. Аналитиков может привлечь взаимосвязь между моделированием и проектированием, что позволит им сделать более эффективный вклад в работу в контексте проектов с гибкой методологией разработки. Стратегические принципы проектирования также могут помочь им в постановке задач и организации работ.

Руководителям программных проектов стоит задуматься, как сделать работу команды более эффективной, сосредоточившись на проектировании программ, близких и понятных специалистам и пользователям в соответствующих профессиональных областях. Поскольку стратегические проектные решения тесно связаны с организацией группы и стилем работы, они неизбежно должны приниматься с участием руководства и иметь решающее влияние на ход выполнения проекта.

Предметно-ориентированная команда

Даже самостоятельному разработчику может принести немало пользы знание многих ценных приемов и общих принципов, почерпнутых из предметно-ориентированного проектирования. Но все же наибольший эффект имеет место тогда, когда предметно-ориентированный подход принимает на вооружение целая группа разработчиков, ставящая модель предметной области в центр внимания своего проекта. Таким образом члены группы приобретают общий язык, делающий более эффективным их взаимодействие и помогающий не оторваться от непосредственного создания программного продукта. В итоге это позволяет четко реализовать замысел, точно следуя модели и имея в руках надежный инструмент практической работы. Такой подход дает разработчикам схему взаимодействия рабочих групп, занятых в проектировании, и позволяет систематически сосредоточиться на функциях и возможностях, наиболее характерных и важных для организации-потребителя.

Предметно-ориентированное проектирование — это технически сложный процесс, но он окупает себя многократно, открывая широкие возможности там, где большинству программных проектов не остается ничего больше, как уйти в прошлое.