Вступление

Но Боб, ты говорил, что закончишь книгу в прошлом году.

Клаудиа Фрерс, UML World, 1999

Гибкая разработка – это умение быстро разрабатывать ПО перед лицом часто меняющихся требований. Для достижения такой гибкости мы должны задействовать методики, которые обеспечивают необходимую дисциплину и обратную связь. Мы должны использовать принципы проектирования, которые сохраняют наше ПО гибким и удобным для сопровождения, а также знать проверенные временем паттерны, позволяющие сбалансировать эти принципы для конкретных задач. Настоящая книга является попыткой увязать вместе все три концепции в функционирующее целое.

В книге описаны эти принципы, паттерны и методики, а также продемонстрировано их применение на десятках учебных примеров. Более важно то, что учебные примеры не представляются как законченные работы. Взамен они имеют вид *продолжающихся* проектов. Вы увидите, что проектировщики допускают ошибки, понаблюдаете, как они идентифицируют свои ошибки, и посмотрите, как они в итоге исправляют их. Вы увидите, как проектировщики ломают голову над дилеммами и беспокоятся о неоднозначностях и компромиссах. Вы увидите проектирование как *действие*.

Самое сложное кроется в деталях

Книга содержит *много* кода Java и C++. Мы надеемся, что вы внимательно изучите этот код, поскольку в значительной степени именно он является *отличительной чертой* книги. Код представляет собой воплощение в жизнь того, что эта книга призвана донести до читателя.

В книге используется повторяющийся шаблон. Она состоит из последовательности учебных примеров варьирующихся размеров. Одни учебные примеры совсем небольшие, другие же требуют для своего рассмотрения нескольких глав. Каждому учебному примеру предшествует материал, который предназначен для подготовки к нему. Скажем, перед учебным примером с системой расчета заработной платы находятся главы, в которых описаны принципы объектно-ориентированного проектирования и паттерны, применяемые в этом учебном примере.

Книга начинается с обсуждения методик и процессов разработки. Это обсуждение перемежается несколькими небольшими учебными примерами. Далее в книге раскрывается тема проектов и принципов проектирования, а затем некоторые паттерны, другие принципы проектирования, регулирующие пакетирование, и дополнительные паттерны. Все эти темы сопровождаются учебными примерами.

Итак, подготовьтесь к чтению кода и изучению диаграмм UML. Книга, к чтению которой вы приступаете, имеет *чрезвычайно* технический характер, и ее наставления кроются в деталях.

Немного истории

В середине 1990-х годов я написал книгу под названием *Designing Object-Oriented C++ Applications using the Booch Method*. Она стала для меня главным произведением, и я был очень доволен результатом и ее продажами.

Настоящая книга начиналась как второе издание *Designing*, но не тут-то было. От исходной книги в ней мало что осталось. До конца добрались немногим более трех глав, да и то они претерпели существенные изменения. Замысел, дух и многие уроки в книге сохранились теми же. Но все-таки за шесть лет, прошедших с момента выхода книги *Designing*, я изучил очень многое о проектировании и разработке ПО. Эта книга отражает результаты моего обучения.

Что это была за половина десятилетия! Книга *Designing* вышла прямо перед тем, как планету захватил Интернет. С того времени количество терминов и аббревиатур, с которыми нам приходилось иметь дело, удвоилось. У нас есть паттерны, Java, EJB, RMI, J2EE, XML, XSLT, HTML, ASP, JSP, сервлеты, серверы приложений, ZOPE, SOAP, C#, .NET и так далее и тому подобное. Поддерживать главы этой книги в актуальном состоянии было нелегко, скажу я вам!

Сотрудничество с Бучем

В 1997 году ко мне обратился Гради Буч с просьбой помочь ему в написании третьего издания поразительно успешной книги *Object-Oriented Analysis and Design with Applications*. Ранее я уже трудился с Гради над несколькими проекта-

ми и запоем читал и участвовал в его разнообразных работах, включая UML. Поэтому я принял приглашение с ликованием. Я попросил своего хорошего друга Джима Ньюкирка помочь с этим проектом.

За следующие два года мы с Джимом написали несколько глав для книги Буча. Конечно, это означало, что я не мог уделять своей книге столько времени, сколько хотел бы, но я чувствовал, что книга Буча стоит потраченных усилий. Кроме того, на то время моя книга была всего лишь вторым изданием *Designing*, и мое сердце ей не принадлежало. Если уж я собирался что-то сказать, то хотел, чтобы это было чем-то новым и особенным.

К сожалению, той версии книги Буча не суждено было выйти. Трудно найти время на написание книги, не отрываясь от повседневной работы. В течение тех стремительных дней разрастания пузыря ".com" это было почти невозможно. Гради был постоянно занят в компании Rational и новых предприятиях вроде Catapulse. Таким образом, проект остановился. В конце концов, я спросил у Гради и в издательстве Addison-Wesley, можно ли включить главы, написанные вместе с Джимом, в *эту* книгу. Они любезно согласились. Так что несколько глав с учебными примерами и диаграммами UML попали сюда из этого источника.

Влияние экстремального программирования

В конце 1998 года появилось экстремальное программирование, которое сразу же бросило вызов нашим заботливо взращиваемым представлениям о разработке ПО. Должны ли мы создавать массу диаграмм UML до написания любого кода или же избегать любого вида диаграмм и просто писать много кода? Должны ли мы готовить множество повествовательных документов, описывающих проект, или же пытаться сделать *код* повествовательным и выразительным до такой степени, чтобы в дополнительных документах отпала необходимость? Должны ли мы программировать парами? Должны ли мы писать тесты до написания производственного кода? Что мы вообще должны делать?

Эта революция пришла в подходящее для меня время. С середины и до конца 1990-х годов компания Object Mentor оказывала помощь довольно многим компаниям в решении вопросов объектно-ориентированного проектирования и управления проектами. Мы помогали компаниям доводить их проекты *до конца*. Как часть этой помощи, мы прививали командам свои позиции и методики. К сожалению, эти позиции и методики не были представлены в письменном виде. Взамен они были устным преданием, передаваемым от нас нашим заказчикам.

К 1998 году я осознал, что нам необходимо описать наши процессы и методики, чтобы мы могли более ясно излагать их заказчикам. Таким образом, я написал много статей о процессе в журнале *C++ Report*. Но эти статьи пролетели мимо цели. Они были информативными, а иногда и занимательными, но вместо систематизации методик и позиций, которые мы на самом деле использовали в своих проектах, статьи непредумышленно дискредитировали те ценности, которые навязывались мне десятилетиями. Именно на это указал мне Кент Бек.

Сотрудничество с Беком

В конце 1998 года, когда меня заботила систематизация в Object Mentor, я наткнулся на работу Кента по экстремальному программированию. Эта работа была разбросана по вики-сайту[1] Уорда Каннингема вперемешку с текстами многих других авторов. Тем не менее, приложив определенный труд и старание, мне удалось уловить суть того, о чем хотел сказать Кент. Я был заинтригован, но настроен скептически. Некоторые аспекты экстремального программирования в точности укладывались в мою концепцию процесса разработки. Однако другие аспекты, такие как отсутствие четких шагов проектирования, оставили меня в недоумении.

Вряд ли могли существовать настолько несопоставимые программные среды, чем те, из которых вышли Кент и я. Он был известным консультантом по Smalltalk, а я – известным консультантом по C++. Этим двум мирам сложно стыковаться друг с другом. Между обеими парадигмами существовала едва ли не кунианская[2] пропасть.

При других обстоятельствах я ни в коем случае не попросил бы Кента написать статью для журнала *C++ Report*. Но согласованность наших размышлений о процессе помогла преодолеть пропасть между языками. В феврале 1999 года я встретился с Кентом на конференции по объектно-ориентированному программированию в Мюнхене. Он читал лекцию об экстремальном программировании в помещении, находящемся напротив того, где я читал лекцию о принципах объектно-ориентированного проектирования. Не имея возможности послушать его лекцию, я отыскал Кента во время обеденного перерыва. Мы говорили об экстремальном программировании, и я предложил ему написать статью для журнала *C++ Report*. Это была великолепная статья о реальном случае, когда Кент вместе с коллегой сумели за считанные часы внести радикальное проектное изменение в работающую систему.

В течение следующих нескольких месяцев я прошел через медленный процесс избавления от собственных страхов перед экстремальным программированием. Самым пугающим для меня было принятие процесса, в котором отсутствует явный предварительный шаг проектирования. Я обнаружил, что не могу переступить через это. Разве у меня нет обязанности перед клиентами, да и перед индустрией в целом, учить их тому, что проектирование является важным в достаточной степени, чтобы тратить на него время?

---

[1] `http://c2.com/cgi/wiki`. Этот веб-сайт содержит большое количество статей разнообразной тематики. Количество авторов исчисляется сотнями или даже тысячами. Говорили, что только Уорд Каннингем мог бы спровоцировать социальную революцию, используя несколько строк кода Perl.

[2] Любая заслуживающая доверия интеллектуальная работа, написанная между 1995 и 2001 годами, просто-таки обязана задействовать термин “кунианский”. Он отсылает к книге Томаса Сэмюэла Куна *The Structure of Scientific Revolutions*, The University of Chicago Press, 1962 год.

Со временем я понял, что на самом деле и сам не практикую такой шаг. Даже во всех написанных мною статьях и книгах, посвященных проектированию, диаграммам Буча и UML, я всегда использовал код как способ проверки содержательности диаграмм. При консультировании всех своих клиентов я тратил час или два, помогая им рисовать диаграммы, и затем предлагал им анализировать диаграммы посредством кода. Я пришел к пониманию того, что хотя трактовка проектирования в экстремальном программировании была для меня чужеродной (в кунианском[3] смысле), методики, лежащие за этой трактовкой, оказались знакомыми.

С другими страхами перед экстремальным программированием мне удалось справиться легче. Я всегда был скрытым парным программистом. Экстремальное программирование дало мне способ отбросить скрытость и наслаждаться страстно желаемой возможностью программировать с партнером. Рефакторинг, непрерывная интеграция и присутствие заказчика на месте – все это я принял очень легко. Оно было весьма близко к тому стилю работы, который я уже советовал своим клиентам.

Одна методика экстремального программирования стала для меня открытием. Идея разработки через тестирование поначалу звучит безобидно. Она предполагает написание тестовых сценариев до написания производственного кода. Весь производственный код пишется для того, чтобы обеспечить проход тестов, которые ранее не проходили. Я не был готов к глубоким последствиям, которые повлек за собой такой стиль написания кода. Эта методика полностью трансформировала способ, которым я писал программы, и трансформировала его в лучшую сторону. Вы увидите такую трансформацию в книге. Часть кода, приведенного в книге, была написана до 1999 года. Вы не найдете для него тестовых сценариев. С другой стороны, весь код, написанный после 1999 года, представлен с тестовыми сценариями, и тестовые сценарии обычно идут первыми. Уверен, что вы заметите это отличие.

Таким образом, к осени 1999 года я был убежден, что компания Object Mentor должна принять экстремальное программирование как предпочтительный процесс разработки, и что я должен отказаться от желания написать собственный процесс. Кент выполнил прекрасную работу по четкой формулировке методик и процесса экстремального программирования, и мои слабые попытки на этом фоне выглядели просто жалкими.

Структура книги

Книга организована в виде шести крупных частей, за которыми следуют приложения.

- **Часть I. Гибкая разработка**

  В этой части описана концепция гибкой разработки. Она начинается с Манифеста Agile Alliance. Затем предлагается обзор экстремального про-

[3] Двукратное упоминание термина “кунианский” повышает степень доверия к работе.

граммирования и, наконец, рассматриваются небольшие учебные примеры, проливающие свет на отдельные методики экстремального программирования, особенно те из них, которые оказывают влияние на способ проектирования и написания кода.

- **Часть II. Гибкое проектирование**

  Главы в этой части посвящены объектно-ориентированному проектированию ПО. Первая глава части ставит вопрос: *что такое проект?* В ней обсуждается задача и приемы управления сложностью. Часть завершается рассмотрением *принципов объектно-ориентированного проектирования классов*.

- **Часть III. Учебный пример: система расчета заработной платы**

  Это самый крупный и наиболее полный учебный пример в книге. В этой части рассматриваются объектно-ориентированное проектирование и реализация на языке C++ простой пакетной системы расчета заработной платы. В первых нескольких главах описаны паттерны, которые встречаются в учебном примере. В финальных двух главах содержится полный код учебного примера.

- **Часть IV. Пакетирование системы расчета заработной платы**

  Часть начинается с рассмотрения *принципов объектно-ориентированного проектирования пакетов*. Затем демонстрируется применение этих принципов на пошаговом пакетировании классов из предыдущей части.

- **Часть V. Учебный пример: метеостанция**

  В этой части содержится один из учебных примеров, который первоначально планировался для книги Буча. В учебном примере с метеостанцией описана компания, которая приняла важное бизнес-решение, и показано, как на него отреагировала команда разработчиков на Java. Как обычно, часть начинается с рассмотрения используемых паттернов и завершается представлением процесса проектирования и реализации.

- **Часть VI. Учебный пример: ETS**

  Эта часть предлагает описание реального проекта, в котором участвовал автор. Проект был в производстве с 1999 года и представлял собой автоматизированную систему тестирования, применяемую для доставки и оценки экзаменов на получение лицензии в Национальном совете коллегий по регистрации архитекторов (NCARB).

- **Приложения по системе обозначений UML**

  Первые два приложения содержат несколько небольших учебных примеров, используемых для описания системы обозначений UML.

- **Другие приложения**

Как читать эту книгу

Если вы – разработчик...

Читайте книгу от корки до корки. Эта книга написана главным образом для разработчиков, и она содержит информацию, которая необходима для разработки ПО в гибкой манере. Читая книгу от корки до корки, вы ознакомитесь с методиками, затем с принципами, далее с паттернами и, наконец, с учебными примерами, которые увязывают все их вместе. Объединение всех этих знаний поможет вам доводить свои проекты *до конца*.

Если вы – менеджер или бизнес-аналитик...

Читайте часть I. В главах этой части подробно обсуждаются гибкие принципы и методики. Они проведут вас от требований и планирования до тестирования, рефакторинга и программирования. Вы получите руководство о том, как формировать команды и управлять проектами. Это поможет вам доводить свои проекты *до конца*.

Если вы хотите изучить UML...

Сначала читайте приложение А, затем приложение Б. После этого читайте все главы из части III. Такой ход чтения даст вам хорошее представление о синтаксисе и применении UML, а также поможет переводить диаграммы UML в код на языке программирования, подобном Java или C++.

Если вы хотите изучить паттерны...

Чтобы найти определенный паттерн, используйте перечень “Список паттернов”, находящийся сразу после оглавления книги. Для изучения паттернов в целом читайте часть II, чтобы сначала ознакомиться с принципами проектирования, а затем части III, IV, V и VI, в которых определены все паттерны и продемонстрировано их применение в типичных ситуациях.

Если вы хотите изучить принципы объектно-ориентированного проектирования...

Читайте части II, III и IV. В главах этих частей описаны принципы объектно-ориентированного проектирования и показано, как их использовать.

Если вы хотите изучить методы гибкой разработки...

Читайте часть I. В главах этой части описана гибкая разработка, начиная с требований и планирования и заканчивая тестированием, рефакторингом и программированием.

Если вы хотите улыбнуться...

Читайте приложение В.

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: info@dialektika.com

WWW: http://www.dialektika.com

Наши почтовые адреса:

в России: 195027, Санкт-Петербург, Магнитогорская ул., д. 30, ящик 116

в Украине: 03150, Киев, а/я 152