ПРОГРАММИСТУ

Л. А. Мацяшек, Б. Лионг

ПРАКТИЧЕСКАЯ программная инженерия

на основе учебного примера

БИНОМ

ПРАКТИЧЕСКАЯ
программная инженерия
на основе учебного примера

Leszek A. Maciaszek, Bruc Lee Liong
With contributions from Stephen Bills

PRACTICAL software engineering

A Case Study Approach

Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney • Singapore • Hong Kong
Tokyo • Seoul • Taipei • New Delhi • Cape Town • Madrid • Mexico City • Amsterdam • Munich • Paris • Milan

ПРОГРАММИСТУ

Л. А. Мацяшек, Б. Л. Лионг

ПРАКТИЧЕСКАЯ программная инженерия

на основе учебного примера

Перевод с английского
А. М. Епанешникова и В. А. Епанешникова

Москва
БИНОМ. Лаборатория знаний
2009

УДК 681.1.06
ББК 32.973-018.2
М12

Мацяшек Л.

М12 Практическая программная инженерия на основе учебного примера / Л. А. Мацяшек, Б. Л. Лионг : пер. с англ. — М. : БИНОМ. Лаборатория знаний, 2009. — 956 с. : ил. — (Программисту).

ISBN 978-5-94774-488-0 (русск.)
ISBN 0-321-20465-4 (англ.)

Рассмотрены вопросы современных методов создания сложного программного обеспечения, использующего информацию, хранимую в базе данных. Подчеркнуты особенности создания такого программного обеспечения коллективом разработчиков: итеративный характер разработки, использование стандартных средств создания программ (стандартные компоненты, паттерны, Bean-компоненты и т. д.). Большое внимание уделено разработке структуры программного обеспечения, позволяющей наиболее просто организовать все стадии его жизненного цикла. Весь материал проиллюстрирован на одном достаточно сложном примере.

Для разработчиков сложного программного обеспечения, а также для студентов вузов, специализирующихся в вопросах создания современного ПО.

УДК 681.1.06
ББК 32.973-018.2

По вопросам приобретения обращаться:

«БИНОМ. Лаборатория знаний» (499) 157-52-72, e-mail: binom@Lbz.ru
http://www.Lbz.ru

ISBN 978-5-94774-488-0 (русск.)
ISBN 0-321-20465-4 (англ.)

Оглавление

ДЕВИЗ

Делайте все настолько просто, насколько возможно, но не проще.

Альберт Эйнштейн

ПОСВЯЩЕНИЯ

Диане, Доминике и Томашу

Лешек А. Мацяшек

Моим родителям, Эдисону и Тине

Брюс Ли Лионг

[. . .]

Введение

История создания книги

История создания этой книги представляет собой итеративный и пошаговый процесс. Конечно, она соответствует всем четырем основным фазам популярной модели жизненного цикла: замысел, разработка, реализация и использование. Книга явилась плодотворным результатом работы двух авторов с учетом требований пользователей, вытекающих из их практики, с широким использованием непрерывной интеграции и рефакторинга, но, к сожалению, с достаточно длинным циклом разработки[1].

Замысел создания книги датируется публикацией в 1990 г. книги Мацяшека *Database Design and Implementation* (Prentice Hall) — *Проектирование и создание баз данных*. Многие читатели настаивали на продолжении этой книги с использованием законченных учебных примеров, маленьких и больших упражнений и с пошаговым (то есть итеративным и возрастающим) увеличением технической трудности и сложности содержания. Деловое предложение насчет книги было сделано, и проект вступил в стадию разработки — представление о книге было уточнено, риски устранены, требования и границы определены. Однако чтобы завершить целевую платформу, вместо работы над книгой пришлось заниматься… большим объемом учебной работы и консультаций в промышленности.

Десятью годами позже после работы над бесчисленными промышленными проектами количество собранного практического материала буквально кричало относительно необходимости публикации для широкой аудитории. Но аудитория изменилась — промышленность вступила в век Интернета. Требовался новый учебник, чтобы дать необходимые знания для разработки современного программного обеспечения (ПО). Таким учебником стал учебник Мацяшека *Requirements Analysis and System Design: Developing Information Systems with UML* (Addison-Wesley, 2001) — Анализ требований и проектирование систем: разработка информационных систем с использованием UML, — вышедший вторым изданием одновременно с этой книгой. Вскоре после этого книга, которую вы держите в руках, вступила в стадию создания.

---

[1] Под рефакторингом в программировании понимается улучшение структуры программного кода с целью более легкого его понимания, но без изменения внешнего поведения этого кода. В данном конкретном случае этот термин относится не к программному коду, а к самой книге, чем авторы подчеркивают некоторую схожесть процесса написания программного кода и данной книги. — *Прим. перев.*

Стадия *реализации* была ухабистой. Первоначально книга предполагалась как дополнение к учебнику Мацяшека 2001 г. или любой подобной книге. Позже акцент сместился в пользу самостоятельного учебника, который использует подход итеративного учебного примера к выработке навыков практической разработки ПО. Книга сконцентрирована на проектировании ПО, программировании и администрировании. Она посвящена вопросам современной практики разработки, методам, техническим приемам и инструментальным средствам.

Стадия *использования* этой книги находится в ваших, читатель, руках. Бета-тестирование (предварительное тестирование) этой книги проводилось и в классных комнатах, и в процессе проектирования ПО. Дальнейшее расширение использования — в воле читателя. Пожалуйста, присылайте пожелания по изменению, сведения о выявленных дефектах и предложения по совершенствованию книги в группу разработчиков.

Схема и организация книги

Отличительный характер этой книги проистекает из двух стремлений авторов. Во-первых, это книга о способе разработки ПО, проверенном *на практике*. Во-вторых, она о разработке ПО применительно к *промышленным приложениям*. Следующее описание того, что промышленные приложения включают и исключают, полностью применимо к этой книге: «Промышленные приложения включают платежную ведомость, регистрацию пациентов, отслеживание отгрузки, стоимостный анализ, оценку кредита, страхование, цепь поставок, бухгалтерский учет, обслуживание клиентов и международную биржевую торговлю. Промышленные приложения не включают вопросы, связанные с инжекцией автомобильного топлива, текстовыми процессорами, устройствами управления подъемниками, устройствами управления химическими заводами, телефонными коммутаторами, операционными системами, компиляторами и играми» [31].

Содержимое книги вращается вокруг одного основного **учебного примера**, двух **небольших проектов** с относящимися к ним упражнениями, большого количества сопутствующих **примеров**, **учебных средств** для рассмотрения основных концепций моделирования и программирования и **примеров задач** в конце каждой главы, которые содержат главным образом **упражнения, связанные с учебным примером**. Организация, на которую даются ссылки в учебном примере и в небольших проектах, а также в упражнениях, является компанией, специализирующейся в *оценке расходов на рекламу*. В книге эта организация по своей основной деятельности называется AEM (advertising expenditure measurement — оценка расходов на рекламу). Фактически же это ACNielsen's Nielsen Media Research в Сиднее, Австралия. Учебный пример называется *Email Management* (EM) — управление электронной почтой. EM является подсистемой системы *Contact Management* (CM) — управление деловыми партнерами AEM.

Как показано на диаграмме Венна, AEM занимается бизнесом, CM является одной из предметных областей бизнеса, а EM является учебным примером. Упражнения и небольшие проекты взяты из срезов областей AEM-биз-

неса, включая СМ. Некоторые примеры не связаны с АЕМ. Учебный пример обогащен дополнительными примерами и упражнениями. Учебные средства используются для того, чтобы быстро изучить вводимые темы, связанные с UML-моделированием, Java-программированием, реляционными базами данных (БД), конструкцией GUI (graphical user interface — графический интерфейс пользователя) и работой с бизнес-компонентами.

В книге делается попытка дать широкие знания по разработке ПО и представить исходную информацию до рассмотрения решений учебного примера. Другой же акцент книги — показать, как использовать эти знания при проектировании ПО. Для обеспечения данных требований книга итеративно развивает модели проектирования и реализации. Учебный пример, небольшие проекты, задачи и упражнения для решения задач тщательно отобраны, чтобы подчеркнуть многочисленные аспекты разработки ПО, как требуется в соответствии с уникальным характером различных приложений и целевых решений ПО.

Книга состоит из четырех частей. В части 1 (**Проектирование программного обеспечения**) обсуждаются жизненный цикл разработки ПО, языки моделирования, средства разработки, планирование проекта и управление процессом. Следующие три части (2-я, 3-я и 4-я) рассматривают учебный пример, небольшие проекты и примеры. Обсуждение в этих трех частях концентрируется на методах, технологиях, процессах и средствах разработки ПО.

Части 2, 3 и 4 соответствуют трем итерациям проекта (учебного примера). Каждая итерация начинается со спецификаций учебного примера, соответствующих начальной модели объекта. Базовая теория и практические знания, подкрепляющие каждую итерацию, объясняются в первую очередь демонстрацией проектирования и программных решений учебного примера. Любая информация, важная для решений учебного примера, но не обладающая существенным базовым значением, представлена внутри этого примера, либо как подраздел обсуждения учебного примера. Каждая итерация завершается полным решением и заканчивается вместе с главой, которая содержит исходный код с необходимыми комментариями, аннотациями и ссылками на объяснения в предшествующих главах.

Часть 2 (**От требований через структурное проектирование к готовому программному обеспечению**) начинается заданием бизнес-параметров для учебного примера ЕМ. Первые две главы этой части представляют модель бизнес-объектов для АЕМ и модель предметной области для СМ. Далее определены требования ЕМ и последовательно разрабатывается его первая итера-

ция. Краеугольный камень первой итерации — надежный структурный проект, поддающийся последовательным пошаговым улучшениям. «Подлежащим сдаче» итогом первой итерации является выпуск в свет ПО для пользователей (то есть читателей этой книги).

Часть 3 (**Рефакторинг программного обеспечения и разработка пользовательского интерфейса**) концентрируется на определении внешнего интерфейса системы и на отдельных частях приложения. В ней обсуждается проектирование графического пользовательского интерфейса (GUI — graphical user interface), включая Web-доступный внешний интерфейс. Переход от итерации 1 к итерации 2 обеспечивается структурным рефакторингом и разработкой дружественного пользовательского интерфейса.

Часть 4 (**Разработка данных и бизнес-компоненты**) перемещает внимание с внешнего интерфейса системы к ее сути и ее промежуточному логическому уровню. В этой части обсуждается хранение данных и манипуляция ими, реализация бизнес-правил, обработка транзакций и управление безопасностью. Она объясняет также, каким образом логика приложения может быть перемещена на сервер приложения в промежуточный логический уровень.

Хотя итерация 3 учебного примера является развитием итерации 2, части 3 и 4 книги можно изучать достаточно независимо. Читатель может сконцентрироваться на одной из этих частей и лишь вскользь просмотреть другие части. Например, проектировщика БД (программиста) может особо заинтересовать часть 3, в то время как GUI-проектировщика или Java-программиста может в первую очередь заинтересовать часть 4. Возможно, некоторые читатели сконцентрируют свое внимание на частях 1 и 2 книги и полностью определятся с частями 3 и 4 после экспериментирования и лучшей оценки знаний, содержащихся в первых частях. В продвинутых курсах, возможно, будет лучше начать использовать книгу с части 2.

Из общего количества 23 глав книги, 6 посвящены учебному примеру ЕМ (это главы 8, 13, 14, 18, 19 и 23). Образовательная цель этих шести глав — в понимании и анализе учебного примера. Это действительно метод обучения на примере. В противоположность этому первые пять глав (часть 1) объясняют основы разработки ПО и не обращаются к учебному примеру. Оставшиеся 12 глав имеют и теоретические части, и части, которые связывают теорию с учебным примером, небольшими проектами или другими примерами.

Отличительные особенности

Основная отличительная особенность книги выражена в ее подзаголовке: **«Подход с использованием учебного примера»**. Если вы считаете, как делают многие педагоги, что лучшая форма обучения — *обучать на примере*, то эта книга для вас. Если вы любите сомневаться и хотите *учиться на ошибках*, то эта книга дает вам много возможностей экспериментировать с вашими решениями и сравнивать их с ответами и объяснениями авторов. Если вдобавок ко всему вы хотели бы *настроить изучение на ваши текущие потребности и уровень ваших знаний*, тогда следует учесть, что каждая итерация имеет свой акцент, свою сложность моделирования и может потребовать отличный от других набор методов разработки и моделей.

Главная цель этой книги состоит в том, чтобы связать теорию с действительностью, уделяя особое внимание проектированию и реализации ПО (одновременно не пренебрегая анализом) и делая акценты на нетривиальных практических проблемах. В своей цели «пояснять на примерах» **эта** книга уникальна наличием следующих моментов:

1. *Цель — обучение*. Книга была написана для обучения. Учебный пример, отдельные примеры и упражнения не только взяты из реальной жизни, они сформированы, чтобы удовлетворить потребности обучения. Реальные решения являются частью сложного бизнеса в контексте создания ПО. Этот контекст, вероятно, может показаться читателю чрезвычайно обширным и неинтересным, так что он упрощен в максимально возможной степени. Представление GUI и проектов БД наряду с примерами программирования исключает несущественные зависимости, «информационный шум» и повторение задач.
2. *Аннотируемые решения*. В информационных системах не существуют «черное или белое», «истина или ложь», «ноль или один». Часто решение преследует конкретную специфическую цель и может показаться совершенно неправильным с точки зрения других применений. Поэтому ответы и решения тщательно аннотируются.
3. *Альтернативные решения*. Иногда отдельное решение, независимо от того, как оно аннотируется и объясняется, ненамного лучше, чем другие потенциальные решения. В этом случае часто даются и объясняются альтернативные решения.
4. *Списки ключевых терминов* в конце каждой главы, организованные в алфавитном порядке с указанием номеров страниц. Списки можно использовать для самостоятельного изучения основной терминологии, помещенной в каждую главу. Они могут также использоваться преподавателями для проверки знаний студентов по каждой главе.
5. *Обзорные вопросы* — для закрепления знаний читателей с помощью серьезных вопросов по каждой главе. Вопросы разделены, когда это удобно, на вопросы, касающиеся обсуждения проблемы, и вопросы по учебному примеру. Ответы на все обзорные вопросы преподаватели могут получить на Web-сайте книги.
6. *Упражнения, связанные с решением конкретных проблем*, имеющие целью стимулировать читателя исследовать проблемы перед попыткой их решения и попробовать получить расширенные или альтернативные решения учебного примера, небольших проектов или других примеров. Типовые решения всех обзорных вопросов доступны преподавателям на Web-сайте книги.
7. *Web-сайт* с полным набором сопутствующего материала, включая модели и код программ (главным образом код UML, Java и БД Oracle). Весь код программ, включая код, не представленный в тексте, доступен на Web-сайте книги.
8. *Акцент на основы*. Имеются определенные четкие основы (модели, структуры, стандарты, библиотеки и т. д.), позволяющие разрабатывать хорошие ПО и системы. В книге определяются и объясняются эти основы и даются источники информации.

9. *Сбалансированная смесь профессиональной глубины и образовательных возможностей*. В общем-то, писать ПО и писать учебники – несколько разные вещи. Хотелось бы надеяться, что данная книга противоречит этому тезису.
10. *Использование вместо профессионального образования и обучающих курсов*. Занятые профессионалы часто решают обычные задачи и могут быстро отстать в искусстве и даже в практике своей дисциплины. Найти время и средства для получения дорогого профессионального образования и посещения обучающих курсов с использованием учебных примеров, подобных тем, которые рассмотрены в этой книге, может оказаться затруднительным. Есть надежда, что эта книга может предоставить профессионалам возможность ознакомиться с самыми последними разработками как альтернативу этим вариантам или даже в процессе выполнения обычных рабочих обязанностей.

Для кого предназначена эта книга

Эта книга нацелена на широкую аудиторию студентов и IT-профессионалов (профессионалов в области информационных технологий). Идеальный читатель — это студент курса разработки ПО или конструктор ПО (а также руководитель проекта/менеджер). Книга написана так, чтобы она могла быть абсолютно понятна студентам и профессионалам, которые обладают элементарными знаниями по информационным системам и основными навыками программирования (желательно на объектно-ориентированном языке и с некоторыми навыками использования БД). Для большинства читателей это соответствует первому году университетского курса по компьютерным наукам или информатике (информационные системы, информационная технология).

Для *студентов* основное использование этого учебника — в курсах разработки ПО с компонентами его проектирования. Книга может также использоваться как учебник в курсах разработки информационных систем, проектирования ПО, или анализа и проектирования систем, которые преподаются на более старших курсах обучения, или как пособие для более подготовленных студентов. Кроме того, книга может быть рекомендована в курсах по объектной технологии, объектному программированию, системам на основе Web-технологий, проектированию и программированию БД и подобных курсах.

Практики, особенно заинтересованные в книге, — это в первую очередь системные проектировщики, программисты, проектировщики ПО, бизнес- и системные аналитики, руководители проектов и менеджеры, Web-разработчики и разработчики по наполнению Web-ресурсов, рецензенты, испытатели, менеджеры по обеспечению качества и промышленные испытатели. Книга может использоваться для профессионального образования, обучающих курсов и семинаров. Она может также быть взята в качестве источника информации для команд проектировщиков. Для практиков, уже использующих UML, Java и реляционные БД в проектах ПО, эта книга может служить как средство проверки правильности своей практической разработки ПО и как источник идей и направлений проектирования.

Сопутствующие материалы

Полный пакет сопутствующих материалов находится на Web-сайте компаньонов. Большая часть содержания Web-сайта свободно доступна всем читателям, но некоторый материал защищен паролем и предназначен для преподавателей, которые будут использовать книгу в обучении. Домашняя страница этой книги находится на сайтах:

http://www.comp.mq.edu.au/books/pse

http://www.booksites.net/maciaszek

Web-сайт этой книги содержит два вида ресурсов: для всех читателей и для преподавателей, использующих эту книгу для обучения. Ресурсы преподавателей защищены паролем.

Ресурсы преподавателей на Web-сайте включают (но только этим не ограничиваются):

1. *Руководство для преподавателя*, содержащее:
   а) *вопросы и ответы* ко всем обзорным вопросам в конце каждой главы и упражнения по решению проблем;
   б) дополнительные *проекты и решения*, не содержащиеся в учебнике и доступные на Web-сайте, чтобы помочь преподавателям выбрать для студентов задания и проекты.
2. *Слайды для лекций* в PowerPoint и в модифицируемом pdf-формате (Acrobat).
3. *UML-модели и исходный код Java/БД* для:
   а) упражнений и небольших проектов;
   б) проектов, находящихся на Web-сайте книги;
   в) альтернативных подходов к проектированию/программированию учебного примера книги.

Ресурсы для всех читателей включают (но не только это):

1. *Слайды лекций* — в pdf-формате (Acrobat) только для чтения.
2. *Опечатки и дополнения*.
3. *UML-модели и исходный код Java/БД* для учебного примера книги и законченных примеров с инструкциями о том, как компилировать и запускать код.

Ваши комментарии, исправления, предложения по усовершенствованию, дополнения и т. д. очень ценны. Пожалуйста, направляйте любую корреспонденцию по адресу:

Leszek A. Maciaszek
Department of Computing
Macquarie University
Sydney
NSW 2109, Australia
email: leszek@ics.mq.edu.au
web: http://www.comp.mq.edu.au/~leszek
телефон: +61 2 9850-9519
факс: +61 2 9850-9551
курьерская почта: North Ryde, Herring Road, Bid. E6A, Room 319

Благодарности

Благодарности авторов

Эта книга потребовала для написания значительного времени, но это время пренебрежимо мало по сравнению со временем, потребовавшимся, чтобы получить знания и навыки, необходимые для ее написания. Наша особая благодарность — нашим друзьям и коллегам в **ACNielsen, Сидней, Австралия**, которые обеспечили начальную «испытательную среду» для многих идей, предложенных читателям в этой книге. Наша благодарность прежде всего Стивену Биллсу (который является участником создания этой книги) и его команде разработчиков, включая Пола Антоуна, Бруно Бейру, Сью Дэйз, Стивена Гротта, Джефа Хонга, Йиджуна Ли, Кевина Мати, Денизу Маккрей, Шанталь О'Коннел, Джеймса Риса, Джована Споа, Эрика Цурхера.

Написание книги — немалый проект. Как сказано в главе 4 и еще в нескольких местах книги, успешное создание проекта требует большой работы и материальных ресурсов для решения его задач. Рабочие ресурсы состоят из людей и оборудования, включая аппаратное и программное обеспечение. Материальные ресурсы — расходные материалы и товарно-материальные ценности. Авторы этой книги распределили себя по задачам, но проект потерпел бы серьезную неудачу без всей сопутствующей работы и материальных ресурсов. Ресурсы обеспечивались **Университетом Маккуэри, Сидней, Австралия** и **Университетом Экономики, Вроцлав, Польша**. Мы обязаны нашим друзьям, коллегам и студентам в этих двух университетах за их советы, поддержку и помощь, за все средства, которые они предоставили этому проекту.

Говоря о ресурсах, следует отметить, что эта книга не могла бы быть написана без интенсивного и всестороннего использования инструментального ПО и сред программирования. ПО, необходимое для книги, было получено различными способами — от приобретения демонстрационных копий до использования свободно распространяемых источников ПО. Люди, связанные с этим ПО, полученным таким образом, неизвестны нам, так что мы не можем передать благодарность конкретно им. Наши благодарности, однако, вместо них адресуются продавцам ПО, которые отвечали на наши запросы, как получить ПО бесплатно для использования в учебных целях нами и нашими студентами. Мы особенно обязаны фирмам Oracle Corporation (Oracle и JDeveloper), Rational Software Corporation, а в настоящее время IBM (Rational Suite), Sybase (PowerDesigner) и yWorks (yDoc).

Особо мы благодарны **Кейту Мансфилду** из Pearson Education, редактору этой книги, а также и редактору книги Мацяшека *Requirements Analysis and Systems Design.* Спасибо, Кейт, за профессиональное видение обеих книг и за Вашу упорную работу от начала до производства и передачи в продажу. Книжное производство на самом деле является усилием целой команды. Рискуя упустить ряд фамилий, мы хотели бы выразить особую благодарность **Аните Аткинсон** (главный редактор), **Элен Макфадьен** (корректор), **Рут Фристон Кинг** (выпускающий редактор) и **Оуэн Найт** (помощник редактора). Спасибо вам за все исправления, улучшения, понимание, советы и тесное сотрудничество.

Благодарности издателя

Мы благодарны за разрешение воспроизвести авторский материал:

Таблица 1.1 основана на информации из *Fundamentals of Software Engineering,* Prentice Hall — Основы разработки ПО (Pearson Education, Inc.), [34]; рисунок 1.10 преобразован из рисунка *Rational Unified Process* (рациональный унифицированный процесс), перепечатанного по разрешению © Copyright 2003 by International Business Machines Corporation. All Rights Reserved с сайта http:// www.rational.com/products/rup/; рисунки 3.1, 4.3, 4.4, 4.5, 4.7, 4.8, 4.9, 4.10, 4.11, 4.13, 4.14, 4.16, 4.17, 4.19, 4.20, 4.21, 4.22, 4.23, 4.25, 4.26, 4.28 и 4.29 — с изображений экрана Microsoft® Office Project (2003), перепечатанные по разрешению Microsoft Corporation, Copyright © 1998–2003 Microsoft Corporation; рисунок 3.2 — с изображения экрана Manage-Pro™ 6.1 из сайта www.managepro.net, March 2004, Performance Solutions Technology, LLC, перепечатанный с любезного согласия Performance Solutions Technology, LLC; рисунок 3.3 — с изображения экрана eRoom из сайта www.eroom.net/eRoomNet/, July 2003, Documentum, Inc., перепечатанный с любезного разрешения Documentum, Inc.; рисунок 3.4 — с изображения экрана eProject Enterprise, July 2003, перепечатанный с любезного разрешения eProject, Inc.; рисунки 3.5 и 3.6 — с изображений экрана *Small Worlds*, перепечатанные с www.thesmallworlds.com/ по разрешению © Copyright 2003 by International Business Machines Corporation. All Rights Reserved; рисунок 3.8 — с изображения экрана @Risk из сайта www.palisade-europe.com, перепечатанный с любезного разрешения Palisade Corporation; рисунки 3.9, 3.10, 3.12, 3.14, 3.16, 3.30, 3.31, 5.14, 5.15 — с изображений экрана *IBM Rational Suite*, перепечатанные по разрешению *Rational Suite Tutorial, Version 2002.05.00,* © Copyright 2002 by International Business Machines Corporation. All Rights Reserved; рисунок 3.9 — с изображения экрана Microsoft® Word, перепечатанный по разрешению Microsoft Corporation, Copyright © 1998–2003 Microsoft Corporation; рисунок 3.11 — с изображения экрана DOORS® из сайта www.telelogic.com, перепечатанный по любезному разрешению Telelogic UK Ltd., Copyright © 2004 Telelogic AB; рисунок 3.13 — с изображения экрана Enterprise Architect из сайта www.sparxsystems.com.au, August 2003, Sparx Systems Pty Ltd., перепечатанный по любезному разрешению Sparx Systems Pty Ltd.; рисунок 3.15 — с изображения экрана Gentleware's Poseidon Sales Application model, August 2003, перепечатанный

с любезного разрешения Gentleware AG; рисунки 3.16, 3.34 и 3.35 — с изображений экрана No Magic's MagicDraw™, перепечатанный по разрешению из сайта www. magicdraw.com, July 2003, © Copyright No Magic, Inc. 1998–2004, All Rights Reserved; рисунок 3.17 — с изображения экрана Sybase® PowerDesigner® version 9.5 из сайта www.sybase.com, July 2003, © Copyright 2002, Sybase, Inc., All Rights Reserved; рисунок 3.25 — с изображения экрана *Borland® Together® ControlCenter® example* (2003), www.borland.com,, перепечатанный по разрешению Borland Software Corporation; рисунок 3.27 — с изображения экрана Oracle JDeveloper из сайта www.otn.oracle.com, перепечатанный по разрешению Oracle Corporation; рисунок 3.29 — с изображения экрана Perforce из сайта www.perforce.com, перепечатанный с любезного согласия Perforce Software, Inc.; рисунок 3.32 — с изображения экрана Microsoft® Visual SourceSafe из сайта http://msdn.microsoft.com/ssafe/default.asp, (2003), перепечатанный по разрешению Microsoft Corporation, Copyright © 1998–2003 Microsoft Corporation; таблицы 4.2 и 4.3 основаны на информации из *Software Cost Estimation with COCOMO II* — оценка стоимости программного обеспечения с помощью COCOMO II, Prentice Hall (Pearson Education, Inc.) [11]; рисунки 5.11 и 20.4 — из *Requirements Analysis and Systems Design with UML* — анализ требований и проектирование систем с помощью UML, Addison Wesley, (Pearson Education), (Maciaszek, L.A., 2001); рисунок 12.11 — с изображения экрана Microsoft® DOS, перепечатанный по разрешению Microsoft Corporation, Copyright © 1990–2003 Microsoft Corporation; рисунок 17.18 преобразован из *Mastering Jakarta Struts* by James Goodwill. Copyright © 2002 by Ryan Publishing Group. Перепечатан здесь по разрешению Wiley Publishing, Inc. All rights reserved; рисунки 22.3 и 22.4 — с изображений экрана Oracle Business Component Browser из сайта www.otn.oracle.com, перепечатанный по разрешению Oracle Corporation.

В некоторых случаях было невозможно проследить владельцев авторского права, и мы будем признательны за любую информацию, которая позволила бы нам сделать это.

Часть 1

Проектирование программного обеспечения

Начинать с определения основной терминологии — хороший стиль в области передачи знаний (типа написания книги или производства ПО). Термины нужно только определять и разъяснять, но по возможности не изобретать. Действительно, большинство терминов, с которыми мы работаем, было изобретено до нас.

Эта книга о ПО и системах. Она посвящена программной инженерии и разработке систем ПО. В ней рассматриваются большие проекты ПО и информационные системы предприятий. В книге используются следующие определения, данные видными экспертами в рассматриваемой области.

«*Программная инженерия* — область информатики, имеющая дело с созданием систем ПО, которые являются настолько большими или настолько сложными, что создаются коллективом или коллективами инженеров» [34]. В этом определении имеется ряд важных моментов.

«Создание систем ПО» — другими словами можно сказать «*разработка систем*». На самом деле программная инженерия — это несколько большее, чем собственно разработка ПО. Она включает также и *сопровождение ПО*. Подобно любому техническому изделию, типа моста или дома, ПО должно поддерживаться. В отличие от случая типичного технического изделия, сопровождение ПО включает его развитие (добавление новых модулей) и глубокие изменения в самих основах продукта. В этом смысле разработка ПО дополняется его обслуживанием.

«*Система* — целенаправленное собрание находящихся во взаимосвязи компонентов, которые работают вместе для достижения некоторой цели» [96]. Разработка системы (и обслуживание) связана с процессами, методами и инструментальными средствами производства ПО. Поскольку ПО является моделью действительности, его разработка связана с *моделированием* продуктов ПО.

Система — это больше, чем просто ПО. «*Инженерия систем* связана со всеми аспектами разработки и развития сложных систем, в которых ПО играет главную роль» [96]. «Инженерия систем концентрируется на разнообразии элементов, анализе, проектировании и организации этих элементов в систему, которая может быть изделием, сервисом или технологией преобразования информации или управления» [81]. А так как ПО играет главную роль в большинстве современных систем, эта книга также и об инженерии систем.

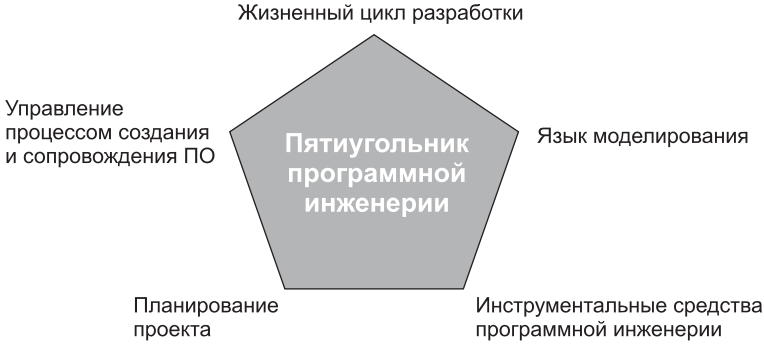
Большинство проектов ПО выполняется в связи с организационными потребностями заняться задачами идентификации в бизнес-процессе или улучшением этого процесса из-за конкуренции. *Проектирование ПО* — запланированная операция, предназначенная для создания программного продукта или сервиса и выполняющаяся в течение определенного времени. В этой книге особое внимание уделяется проектам ПО, предназначенным для *информационных систем предприятий*. Это большие и сложные системы. Как отмечено Фаулером [31], «Промышленные приложения часто содержат сложные данные, и большинство их работает на основе бизнес-правил, которым не хватает логических проверок».

Standish Group [98] исследует причины *отказов ПО*. Оригинальный *Отчет о хаосе* (Chaos Report), выполненный Standish Group в 1994 г., сообщил, что только 16.2 процентов проектов ПО были закончены вовремя и

в пределах выделенных средств. Насчет более крупных и сложных систем сведения оказались даже еще хуже: только 9 процентов таких проектов были закончены вовремя и без перерасхода средств. *Отчет о хаосе* за 2003 г. показал улучшение в этом деле — 34 процента проектов были закончены вовремя и в пределах выделенного бюджета. Хотя улучшение существенно, все это выглядит мрачно по сравнению с традиционными техническими дисциплинами типа архитектуры или электротехники.

Успех реализации проекта ПО обусловлен пятью взаимосвязанными аспектами — см. *пятиугольник программной инженерии*, изображенный ниже. Эти аспекты следующие:

- жизненный цикл разработки ПО — глава 1;
- язык моделирования ПО — глава 2;
- инструментальные средства программной инженерии — глава 3;
- планирование работ над проектом ПО — глава 4;
- управление процессом разработки и сопровождения ПО — глава 5.



Основные учебные цели части 1 состоят в том, чтобы получить знания в следующих вопросах:

- сущность программной инженерии;
- стадии жизненного цикла и модели; в частности, итеративная и пошаговая разработка;
- языки моделирования ПО, в частности, UML — объектно-ориентированный язык моделирования;
- инструментальные средства программной инженерии для управления проектом, моделирования систем, интегрированного коллективного программирования, управления изменением и конфигурацией проекта;
- планирование работы над проектом и оценка бюджета;
- технологии отслеживания хода выполнения проекта;
- управление людскими ресурсами, привлеченными к проекту;
- управление рисками проекта;
- управление качеством ПО;
- управление изменением и конфигурацией.

Глава 1

Жизненный цикл разработки программного обеспечения

В обычном использовании термин «жизненный цикл» означает «изменения, которые происходят в жизни животного или растения» [18]. В программной инженерии термин «**жизненный цикл**» (на английском языке lifecycle — обычно пишется единым словом) применяется к искусственным системам ПО и означает изменения, которые происходят в «жизни» программного продукта. Различные стадии между «рождением» изделия и его возможной «смертью» известны как **стадии жизненного цикла**.

Изменения, а следовательно, и стадии, являются последовательными. Изделие *создается* поэтапно, за ряд стадий. Таким образомь, разработка является повторяющейся и пошаговой. В конечном счете, изделие поэтапно *выводится из работы* — его использование постепенно прекращается. Следовательно, и прекращение его работы является пошаговым. Разумно думать, что ПО в любое время, кроме стадии непосредственного внедрения, находится или в фазе создания, или в фазе снятия с производства. Сопровождение ПО, несмотря на его эволюционный характер, одновременно начинает и процесс снятия с эксплуатации.

Рис. 1.1 показывает типичные стадии жизненного цикла ПО (объясненные более подробно в разделе 1.2). Эти стадии следующие:

1. анализ требований;
2. проектирование системы;
3. реализация;
4. интеграция и внедрение;
5. процесс функционирования и сопровождения.

Рис. 1.1 демонстрирует, что как только программный продукт внедрен в организацию, он остается там навсегда, хотя и под различными «перевоплощениями». Организация уже не может вернуться к ручному способу ведения бизнеса. Однажды включенный в действие, программный продукт поддерживается «до смерти».

Сопровождение, даже если оно развивает систему, ведет, в конечном счете, к ухудшению ее первоначальной структуры. Система становится **унаследованной системой** — она не может быть больше «усовершенствована», и даже вспомогательное и корректирующее сопровождение становится большой проблемой. Вся система или основные ее компоненты должны постепенно удаляться. Осознание, что система является унаследованной, приводит к

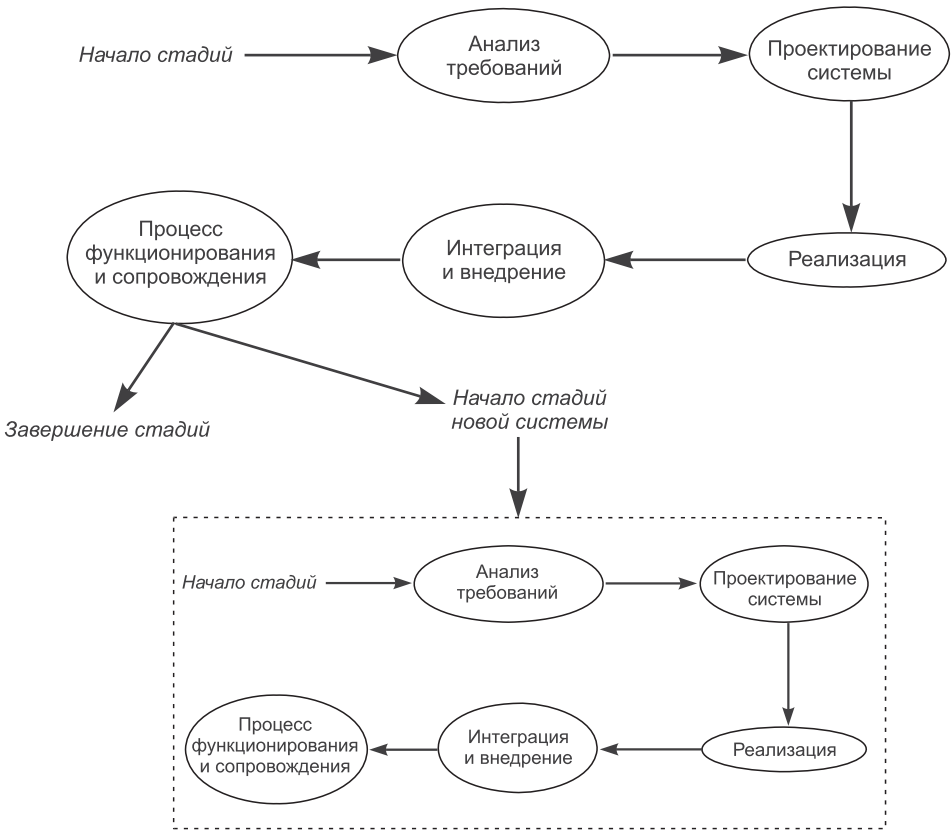


Рис. 1.1. Стадии жизненного цикла ПО

решению разработать новую систему. Это стимулирует новый жизненный цикл, показанный в нижней части рис. 1.1. Постепенное сокращение старой системы и синхронизация с новой системой проводится в параллель, пока новая система не будет полностью развернута для пользователей. Но даже и после развертывания старая система может оставаться в эксплуатации в течение некоторого времени, пока новая система не продемонстрирует свою полноценность.

Особенностью рис. 1.1 является отсутствие *тестирования* как стадии жизненного цикла. Тестирование, как и действия по управлению проектом, включая сбор *системы показателей* проекта, является всеобъемлющей деятельностью, которая выполняется на всех стадиях жизненного цикла.

1.1. Сущность программной инженерии

Понимание жизненного цикла ПО является необходимым условием понимания сущности программной инженерии — ее фундаментальной природы, контекста формирования ПО. Суть **программной инженерии** отражается в следующих ключевых выводах:

[. . .]

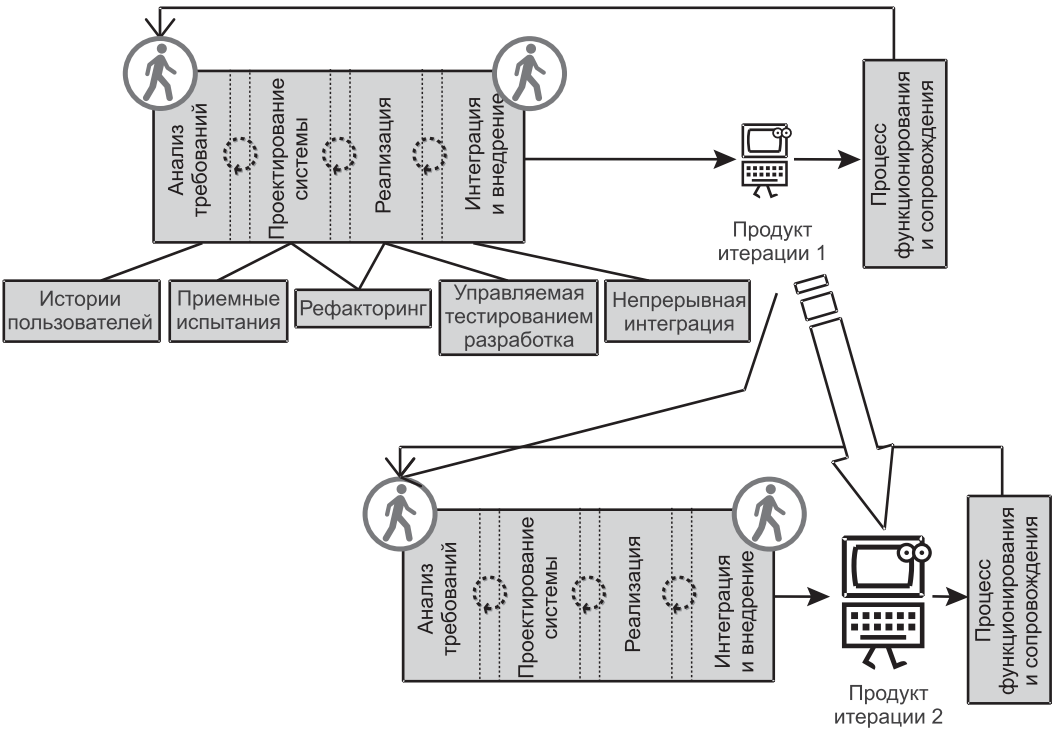


Рис. 1.12. Быстрая разработка ПО

«Обычные» виды проектирования систем и их реализация заменены в быстрой разработке комбинацией приемочных испытаний, рефакторинга и управляемой тестированием разработки. *Приемочные испытания* — это специальные программы, через которые разрабатываемая прикладная программа должна пройти, чтобы быть принятой клиентами. Этот процесс называется «**управляемой тестированием разработкой**» (Test-Driven Development — TDD) или *программированием намерений* (Intentional Programming — IP) — разработчик программирует свои намерения по приемочным испытаниям до того, как использует их. Этому подходу сопутствует частое использование **рефакторинга** (refactorings) — усовершенствования в структуре системы без изменения ее поведения.

Быстрая разработка поддерживает и другие концепции типа **парного программирования** и *коллективной собственности*. Все программирование выполняется парами программистов — двумя программистами, работающими вместе на одной рабочей станции. Один программист пишет код, а другой наблюдает за процессом и задает вопросы. Роли меняются всякий раз, когда один из программистов хочет «поставить точку с помощью клавиатуры». Пары программистов также меняются местами, по крайней мере, один раз в день. Результатом является *коллективная собственность*. Никто индивидуально не владеет кодом. Считается, что высокий дух коллективизма и желание выдать продукт перевесят любые требования индивидуальной ответственности.

«Обычные» интеграция и внедрение заменены в быстрой разработке *непрерывной интеграцией* и *короткими циклами*. Пары программистов могут экспортировать свой код и по своему желанию объединять его с остальной частью. Более того, одну и ту же часть кода может импортировать и работать над ней более одной пары программистов. Это может привести к конфликту, когда коллектив, который хочет экспортировать и сдать свой код, обнаруживает, что другой коллектив сделал ранее несовместимый код. Такие конфликты между участвующими в разработке коллективами должны быть урегулированы.

Быстрая разработка не означает плохое планирование. Фактически даты внедрения тщательно планируются. Каждая итерация обычно планируется так, чтобы закончить работу за короткий цикл продолжительностью в две недели. Продукт в конце двухнедельного цикла — пробный вариант для оценки клиентом. Основная поставка продукта в производство является результатом приблизительно шести двухнедельных циклов.

Быстрая разработка отличается больше в методах, чем в подходе к итеративной разработке. Ее главный представитель — **eXtreme Programming** (XP) [6]. Как и с MDA-подходом, будущее покажет, сможет ли быстрая разработка применяться к большим и сложным системам. Главная опасность, стоящая перед сторонниками быстрого жизненного цикла — риск окончания работы с неудачно *созданной и принятой моделью* [92], в которой ПО слеплено без учета заданных требований к проекту.

Резюме

1. *Программная инженерия* связана с разработкой больших систем ПО. Программная инженерия — обычно центральная деятельность более широкого понятия — *инженерии систем*.
2. *Пятиугольник программной инженерии* состоит из жизненного цикла разработки ПО, языка моделирования ПО, инструментальных средств программной инженерии, планирования проектирования ПО и управления процессом создания и эксплуатации ПО.
3. Стадии процесса разработки ПО упомянуты как стадии *жизненного цикла* ПО. Стадии жизненного цикла, принятые в книге, — анализ требований, проектирование системы, реализация, интеграция и внедрение, а также процесс функционирования и сопровождения.
4. Система ПО — просто часть намного большей *информационной системы предприятия*.
5. Процесс создания и эксплуатации ПО — часть *бизнес-процесса*. Результат процесса создания и эксплуатации ПО — ПО. Результат бизнес-процесса — бизнес.
6. Система ПО может обслуживать любой из трех уровней управления: оперативный, тактический или стратегический.
7. Нематериальный и изменчивый характер ПО — всего лишь два фактора, которые отличают программную инженерию от *традиционной инженерии*.
8. Программная инженерия — больше, чем *программирование*. Программная инженерия применяется к сложным проблемам, которые не могут быть решены одним программированием.

[. . .]

19. *Спиральная модель* — на самом деле базовая метамодель, которая охватывает все итеративные модели. Модель состоит из четырех секторов жизненного цикла: планирование, анализ риска, инженерия и оценка проекта клиентом. Анализ рисков — наиболее характерная особенность спиральной модели.
20. *RUP* — больше, чем модель жизненного цикла. Это также и среда поддержки (называемая RUP-платформой), чтобы помочь разработчикам в использовании и приспособлении к жизненному циклу RUP. Подобно спиральной модели RUP использует управление рисками.
21. *MDA*-модель основана на идее выполнимых спецификаций. Это — современный представитель модели преобразования, которая в свою очередь происходит от формальной разработки систем. Технология компонентов — сердце MDA-модели.
22. *Быстрая разработка* подчеркивает, что производство ПО — творческая деятельность, которая зависит от сотрудничества людей и коллективов гораздо больше, чем от различных процессов, использования инструментальных средств, документации, планирования и других формальных операций.

Ключевые термины

Обзорные вопросы

1. Объясните, как программная инженерия и инженерия систем соотносятся друг с другом. Является ли это отношением включения или пересечения? Может быть, эти две концепции вообще не имеют друг к другу никакого отношения?
2. Каковы пять главных аспектов программной инженерии? Можете ли вы сказать, что любой случай программной инженерии обязательно охватывает все эти аспекты?
3. Какие факторы определяют, что система является унаследованной? Может ли унаследованная система быть преобразованной в современную систему? Как это может быть сделано, если вообще возможно?
4. Что мы подразумеваем, когда говорим, что информационные системы являются социальными системами? Может ли быть система социальной? Рассмотрите следующее объяснение термина «социальный» — «касающийся действий, когда вы встречаетесь и проводите время с другими людьми, и которые происходят в то время, когда вы не работаете» [18].
5. В науке управления существует строгое различие между эффективностью и результативностью систем и людей. В обычном использовании, термин «эффективность» означает «когда кто-то или что-то хорошо использует

время и энергию без каких-либо пустых затрат» [18]. Термин «результативность» означает «насколько успешно это ... в достижении результатов, которые вы хотите получить» [18]. Как эти два термина применимы к программной инженерии? Является ли программная инженерия эффективной или результативной или и тем, и другим? Объясните. Приведите примеры.

6. Каковы основные различия между БД и хранилищем данных? Как эти две концепции связаны с уровнями управления в организации?
7. Что такое приемлемая система? Может ли быть унаследованная система приемлемой? Объясните.
8. Что мы подразумеваем под прямым и обратным проектированием? Как это касается программирования?
9. Программная инженерия связана с моделированием систем. Что моделируется в процессе программной инженерии? Как это соотносится с понятием абстракции? Имеет ли смысл говорить о моделировании программ?
10. Как вы понимаете различие между функциональным и объектно-ориентированным подходами к разработке системы?
11. Какой фактор наиболее важен в определении сложности современной объектно-ориентированной системы? Какова основная технология управления сложностью и ее уменьшением? Объясните.
12. Объясните отношение между анализом требований и техническими требованиями.
13. Каково различие между определением требований и техническим заданием? Объясните.
14. Проведите линию раздела между анализом требований и проектированием системы. Когда анализ становится проектированием?
15. Как детальное проектирование и структурное проектирование относятся друг к другу?
16. Каковы главные подходы в технологии тестирования?
17. Объясните использование заглушек и драйверов в тестировании интеграции.
18. В современной программной инженерии модель водопада для жизненного цикла заменена итеративными моделями. Имеются ли какие-либо аспекты модели водопада, отсутствующие или не выполнимые в итеративных моделях, которые принесли бы пользу итеративному подходу? Обсудите.
19. Что такое управление рисками? Какая из моделей жизненного цикла наиболее явно использует управление рисками? Объясните.
20. Что такое выполнимые спецификации? Какая модель жизненного цикла использует выполнимые спецификации как свой основной момент? Объясните.
21. Каковы наиболее необычные аспекты быстрой разработки (необычные по сравнению с другими итеративными подходами)? Объясните.