

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.)

© Pearson Education Limited 2005.
This translation of PRACTICAL SOFTWARE ENGINEERING: A CASE-STUDY APPROACH, First Edition is published by arrangement with Pearson Education Limited.
© Перевод на русский язык, БИНОМ. Лаборатория знаний, 2009

Оглавление

Экскурс в структуру книги	20
Введение	22
Благодарности	29
Часть 1. Проектирование программного обеспечения	33
Глава 1. Жизненный цикл разработки программного обеспечения.	36
1.1. Сущность программной инженерии	37
1.1.1. Система ПО меньше, чем информационная система предприятия	38
1.1.2. Процесс создания и эксплуатации ПО является частью бизнес-процесса	39
1.1.3. Программная инженерия отличается от традиционной инженерии	41
1.1.4. Программная инженерия больше, чем программирование	43
1.1.5. Программная инженерия напоминает моделирование	44
1.1.6. Система ПО сложна	45
1.2. Стадии жизненного цикла	48
1.2.1. Анализ требований	48
1.2.2. Проектирование системы	50
1.2.3. Реализация	51
1.2.4. Интеграция и внедрение	52
1.2.5. Процесс функционирования и сопровождения.	54
1.3. Модели жизненного цикла	55
1.3.1. Жизненный цикл «водопад с обратной связью»	56
1.3.2. Итеративный пошаговый жизненный цикл	59
Спиральная модель	60
Rational Unified Process (RUP)	62
Model Driven Architecture (MDA)	63
Быстрая разработка ПО с короткими итерациями	65
<i>Резюме</i>	67
<i>Ключевые термины</i>	69
<i>Обзорные вопросы</i>	70
Глава 2. Язык моделирования программного обеспечения	72
2.1. Язык структурного моделирования	73
2.1.1. Моделирование потока данных	74
2.1.2. Моделирование сущностей и отношений.	77
2.2. Язык объектно-ориентированного моделирования	79
2.2.1. Диаграммы классов.	80
2.2.2. Диаграммы сценариев использования	83
2.2.3. Диаграммы взаимодействия	87
Диаграммы последовательности действий	88

		90
2.2.4.		91
2.2.5.		93
2.2.6.		94
		95
		97
		98
		99
		100
		101
Глава 3.	Инструментальные средства программной инженерии.	103
3.1.	Инструментальные средства управления проектом	104
3.1.1.	Планирование и управление проектом	105
3.1.2.	Управление проектированием и реализацией с учетом основных показателей	107
3.1.3.	Унификация управления проектом с организацией совместной работы и информационного обеспечения на основе Web-технологии.	107
3.1.4.	Унификация управления проектом на основе портфельной Web-технологии.	109
3.1.5.	Интеграция управления проектом с метриками	111
3.1.6.	Интеграция управления проектом с управлением рисками	113
3.2.	Инструментальные средства моделирования систем	114
3.2.1.	Управление требованиями	116
3.2.2.	Визуальное UML-моделирование	119
3.2.3.	Формирование отчетов	121
3.2.4.	Моделирование БД	124
3.3.	Интегрированные среды разработки	126
3.3.1.	Задачи стандартного программирования	127
	Написание программы	127
	Выполнение программы	131
	Отладка программы	131
3.3.2.	Интеграция с моделированием ПО	134
3.3.3.	Разработка приложения предприятия	135
3.3.4.	Интеграция с бизнес-компонентами	137
3.3.5.	Интеграция с управлением изменениями и конфигурацией	138
3.4.	Инструментальные средства управления изменениями и конфигурацией	140
3.4.1.	Поддержка изменений	141
3.4.2.	Поддержка версий.	144
3.4.3.	Поддержка формирования системы	144
3.4.4.	Поддержка реинжиниринга.	146
		149
		151
		151
		152
Глава 4.	Планирование и отслеживание проекта программного обеспечения	155
4.1.	Разработка плана проекта	155
4.2.	Планирование проекта	160
4.2.1.	Задачи, контрольные точки и подлежащие сдаче продукты	160
4.2.2.	Планирование задач в виде ленточной диаграммы.	162

4.2.3. Ресурсы и календари ресурсов	165
4.2.4. Планирование, определяемое трудозатратами, в виде ленточной диаграммы	166
4.2.5. Неполное и избыточное распределение ресурсов	168
4.3. Оценка бюджета проекта	170
4.3.1. Оценка бюджета на основе графика выполнения.	172
4.3.2. Алгоритмическая оценка бюджета	176
Принципы алгоритмических моделей	177
СОСОМО 81	178
СОСОМО II	180
4.4. Отслеживание выполнения проекта	184
4.4.1. Отслеживание графика	185
4.4.2. Отслеживание бюджета.	188
Фактические затраты, полученные из графика выполнения	188
Фактические затраты, полученные из бухгалтерского учета.	189
Выполненная стоимость	190
<i>Резюме</i>	194
<i>Ключевые термины</i>	196
<i>Обзорные вопросы</i>	197
<i>Примеры задач</i>	197
Глава 5. Управление процессом создания и отслеживания программного обеспечения	200
5.1. Управление людьми	202
5.1.1. Привлечение и мотивация людей	202
Формирование коллектива	203
Теории мотивации.	204
5.1.2. Организация связи в проекте.	206
Формы связи	206
Линии связи	207
Показатели связи	208
Связь в разрешении конфликтов	209
5.1.3. Создание коллектива	210
5.2. Управление рисками	211
5.2.1. Идентификация рисков	212
5.2.2. Оценка рисков	213
5.2.3. Обработка рисков	216
5.3. Управление качеством	217
5.3.1. Показатели качества программного обеспечения	218
5.3.2. Контроль качества.	221
Тестирование ПО	221
Технологии тестирования	223
Планирование испытаний	227
5.3.3. Гарантия качества	229
Контрольные списки	229
Обзоры	230
Ревизии.	231
5.4. Управление изменениями и конфигурацией	232
5.4.1. Изменения требований	233
5.4.2. Версии продуктов разработки	235
5.4.3. Дефекты и усовершенствования	237

5.4.4. Метрики	240
<i>Резюме</i>	243
<i>Ключевые термины</i>	245
<i>Обзорные вопросы</i>	247
Часть 2. От требований через структурное проектирование к готовому программному обеспечению.	249
Глава 6. Модель бизнес-объектов.	252
6.1. Advertising Expenditure Measurement, ее бизнес	253
6.2. Диаграмма бизнес-контекста	254
6.3. Модель бизнес-сценария использования	255
6.3.1. Бизнес-сценарий использования и бизнес-акторы	255
6.3.2. Модель бизнес-сценариев использования для АЕМ	256
6.3.3. Альтернативная модель бизнес-сценариев использования для АЕМ	258
6.4. Бизнес-гlossарий	261
6.4.1. Бизнес-гlossарий для АЕМ	261
6.5. Модель бизнес-классов	262
6.5.1. Бизнес-сущности	262
6.5.2. Модель бизнес-классов для АЕМ	262
6.5.3. Альтернативная модель бизнес-классов для АЕМ	264
<i>Резюме</i>	265
<i>Ключевые термины</i>	266
<i>Обзорные вопросы</i>	266
<i>Вопросы для обсуждения.</i>	266
<i>Вопросы учебного примера.</i>	267
<i>Примеры задач.</i>	267
<i>Упражнения учебного примера</i>	267
<i>Небольшой проект — оценка расходов на рекламу</i>	267
<i>Упражнения</i>	269
Глава 7. Объектная модель предметной области	271
7.1. Управление деловыми партнерами — предметная область	272
7.2. Модель сценариев использования предметной области	273
7.2.1. Сценарии использования и акторы	273
7.2.2. Отношения сценариев использования	274
7.2.3. Модель сценариев использования для управления деловыми партнерами	275
7.2.4. Альтернативная модель сценариев использования для управления деловыми партнерами	277
7.3. Гlossарий предметной области	279
7.3.1. Гlossарий предметной области для управления деловыми партнерами	279
7.4. Модель классов предметной области	281
7.4.1. Классы и атрибуты	282
7.4.2. Отношения классов	284
7.4.3. Модель классов для управления деловыми партнерами	285
7.4.4. Альтернативная модель классов для управления деловыми партнерами	286
<i>Резюме</i>	288
<i>Ключевые термины</i>	289
<i>Обзорные вопросы</i>	289
<i>Вопросы для обсуждения.</i>	289

<i>Вопросы учебного примера</i>	290
<i>Примеры задач</i>	290
<i>Упражнения учебного примера</i>	290
<i>Небольшой проект — временной протокол</i>	291
Глава 8. Итерация 1. Требования и объектная модель	294
8.1. Модель сценариев использования	295
8.2. Документ сценария использования	296
8.2.1. Краткое описание, предусловия и постусловия	297
8.2.2. Основной поток	298
8.2.3. Подпотoki	299
8.2.4. Потoki исключений	302
8.3. Концептуальные классы	303
8.4. Дополнительная спецификация	304
<i>Резюме</i>	306
<i>Ключевые термины</i>	307
<i>Обзорные вопросы</i>	307
<i>Вопросы для обсуждения</i>	307
<i>Вопросы учебного примера</i>	308
<i>Примеры задач</i>	308
<i>Упражнения учебного примера</i>	308
<i>Небольшой проект — временной протокол</i>	309
Глава 9. Структурный проект	310
9.1. Структурные уровни и управление зависимостями	311
9.1.1. Структурные модули	311
Классы проекта	312
Пакеты	312
9.1.2. Зависимости пакетов	313
9.1.3. Зависимости между уровнями	314
9.1.4. Зависимости классов	317
9.1.5. Наследование зависимостей	318
Наследование без полиморфизма	321
Расширяющее и ограничивающее наследование	321
Вызовы методов подкласса	323
Вызовы методов суперкласса	323
9.1.6. Зависимости методов	323
Зависимости методов при наличии делегирования	325
Зависимости методов в присутствии наследования реализации	326
9.1.7. Интерфейсы	329
Зависимость реализации	330
Зависимость использования	330
Устранение циклических зависимостей с интерфейсами	331
9.1.8. Обработка событий	333
Обработка событий и зависимости уровней	335
Обработка событий и интерфейсы	336
9.1.9. Знакомство	338
Зависимости знакомства и интерфейсы	339
Пакет знакомств	340
9.2. Структурные шаблоны	343
9.2.1. Model-View-Controller (MVC)	343
9.2.2. Presentation-Control-Mediator-Entity-Foundation	345

Уровни PCMEF	346
Принципы PCMEF	348
Знакомство в PCMEF+	349
Развертывание PCMEF-уровней	350
9.3. Структурные паттерны	352
9.3.1. Фасад	352
9.3.2. Абстрактная фабрика	354
9.3.3. Цепочка обязанностей	355
9.3.4. Наблюдатель	355
9.3.5. Посредник	358
<i>Резюме</i>	359
<i>Ключевые термины</i>	361
<i>Обзорные вопросы</i>	362
<i>Примеры задач</i>	363
<i>Упражнения учебного примера</i>	363
<i>Небольшой проект — управление информацией о партнерах</i>	363
<i>Упражнения</i>	370
Глава 10. Проектирование и программирование базы данных	371
10.1. Быстрое обучение реляционным базам данных с точки зрения разработки программного обеспечения	372
10.1.1. Таблица	373
10.1.2. Ссылочная целостность	375
10.1.3. Концептуальная модель в сравнении с логической моделью БД	377
10.1.4. Реализация бизнес-правил	378
10.1.5. Программирование логики СУБД-приложения	381
10.1.6. Индексы	383
10.2. Отображение временных объектов в сохраняемые записи	387
10.2.1. Объектные БД, SQL:1999 и потеря соответствия	388
10.2.2. Объектно-реляционное отображение	389
Отображение ассоциации и агрегирования «один ко многим»	390
Отображение ассоциации «многие ко многим»	390
Отображение ассоциации «один к одному»	392
Отображение рекурсивной ассоциации «один ко многим»	393
Отображение рекурсивной ассоциации «многие ко многим»	394
Отображение обобщения	395
10.3. Проектирование и создание БД для управления электронной почтой	396
10.3.1. Модель БД	396
10.3.2. Создание схемы БД	398
10.3.3. Пример содержимого БД	399
<i>Резюме</i>	401
<i>Ключевые термины</i>	401
<i>Обзорные вопросы</i>	402
<i>Вопросы для обсуждения</i>	402
<i>Вопросы учебного примера</i>	403
<i>Примеры задач</i>	403
<i>Упражнения учебного примера</i>	403
<i>Небольшой проект — управление информацией о партнерах</i>	403
Глава 11. Проектирование классов и взаимодействия	405
11.1. Определение классов из требований сценария использования	406

11.1.1. Определение классов из требований сценария использования для управления электронной почтой	408
11.1.2. Проектирование исходных классов для управления электронной почтой	412
Константы в интерфейсе	414
11.2. Структурная разработка проекта классов	414
11.2.1. Структурная разработка проекта классов для управления электронной почтой	415
11.2.2. Проект классов для управления электронной почтой после структурной проработки	419
11.2.3. Инициализация классов	419
Кто инициализирует первый объект?	421
Диаграмма инициализации для управления электронной почтой.	421
11.3. Взаимодействия	422
11.3.1. Диаграммы последовательности действий	423
11.3.2. Диаграммы связей	425
11.3.3. Диаграммы просмотра взаимодействий.	427
11.4. Взаимодействия для управления электронной почтой	427
11.4.1. Взаимодействие «Регистрационное имя».	429
11.4.2. Взаимодействие «Выход»	431
11.4.3. Взаимодействие «Просмотр непосланных сообщений».	431
11.4.4. Взаимодействие «Отображение текста сообщения»	433
11.4.5. Взаимодействие «Сообщение, передаваемое по электронной почте»	434
11.4.6. Взаимодействие «Неправильное имя пользователя или неправильный пароль»	436
11.4.7. Взаимодействие «Неправильная опция»	436
11.4.8. Взаимодействие «Слишком много сообщений»	437
11.4.9. Взаимодействие «Сообщение не может быть послано по электронной почте»	438
<i>Резюме</i>	439
<i>Ключевые термины</i>	440
<i>Обзорные вопросы</i>	441
<i>Вопросы для обсуждения</i>	441
<i>Вопросы учебного примера</i>	441
<i>Примеры задач</i>	441
<i>Упражнения учебного примера</i>	441
<i>Небольшой проект — система использования временного протокола</i>	442
<i>Небольшой проект — управление информацией о деловых партнерах</i>	443
Глава 12. Программирование и тестирование.	445
12.1. Быстрое обучение языку Java с точки зрения разработки программного обеспечения	446
12.1.1. Класс.	446
12.1.2. Ассоциации и коллекции классов	450
От концептуальной модели к модели проектирования классов.	450
Коллекции Java	452
Ассоциации на объектах-сущностях	454
Параметризованные типы C++.	455
12.1.3. Доступ к БД в Java	458
Сравнение JDBC и SQLJ.	459
Установление связи с БД.	460

Выполнение SQL-операторов	461
Вызов хранимых процедур и функций	464
12.2. Управляемая тестированием разработка	467
12.2.1. Шаблон JUnit	469
12.2.2. Управляемая тестированием разработка в управлении электронной почтой	472
12.3. Приемочные испытания и регрессионное тестирование	478
12.3.1. Сценарии тестирования в управлении электронной почтой.	480
12.3.2. Испытательные входные и выходные данные и регрессионное тестирование в управлении электронной почтой	482
12.3.3. Реализация сценария тестирования в управлении электронной почтой	485
12.4. Итерация 1. Образы экрана времени выполнения.	489
<i>Резюме.</i>	494
<i>Ключевые термины.</i>	495
<i>Обзорные вопросы</i>	495
<i>Примеры задач</i>	496
<i>Обучение и упражнения учебного примера</i>	496
<i>Небольшой проект — система использования временного протокола</i>	498
<i>Небольшой проект — управление информацией о деловых партнерах</i>	499
Глава 13. Итерация 1. Аннотированный код	500
13.1. Обзор кода	500
13.2. Пакет Acquaintance	502
13.2.1. Интерфейс IAConstants	503
13.2.2. Интерфейс IAEmployee.	505
13.2.3. Интерфейс IAContact	505
13.2.4. Интерфейс IAOutMessage	506
13.3. Пакет Presentation	508
13.3.1. Класс PMain	508
13.3.2. Класс PConsole	509
Конструирование объекта PConsole	510
Отображение регистрационного имени и меню	512
Просмотр исходящих сообщений	513
Требование к передаче по электронной почте исходящего сообщения. . . .	515
13.4. Пакет Control	517
13.4.1. Класс CActioner	517
Конструирование объекта CActioner	518
Инициализация регистрационного имени.	519
Поиск исходящих сообщений	520
Передача по электронной почте исходящего сообщения	521
Использование JavaMail™ API	522
13.5. Пакет Entity	522
13.5.1. Интерфейс IEDataSupplier	523
Идентификаторы объектов и паттерн Поле идентификации	525
13.5.2. Класс EEmployee	526
Конструирование объекта EEmployee.	527
Получение непосланных сообщений	527
Удаление посланных исходящих сообщений.	528
13.5.3. Класс EContact	528
Конструирование объекта EContact	529

Получение непосланных исходящих сообщений	529
Удаление посланных исходящих сообщений	530
13.5.4. Класс EOutMessage	530
Конструирование объекта EOutMessage	532
Получение и задание делового партнера для исходящего сообщения	533
Получение и задание служащего-создателя для исходящего сообщения	533
Получение и задание служащего-отправителя исходящего сообщения	534
13.6. Пакет Mediator	534
13.6.1. Класс MBroker	535
Конструирование объекта MBroker	536
Связь для запроса регистрационного имени	536
Создание кэша сотрудников	537
Извлечение непосланных сообщений	538
Создание кэша исходящих сообщений	539
Создание кэша деловых партнеров	540
Обновление исходящих сообщений после передачи по электронной почте и восстановление кэша	541
13.7. Пакет Foundation	542
13.7.1. Класс FConnection	542
Конструирование объекта FConnection	543
Получение соединения с БД	544
13.7.2. Класс FReader	545
13.7.3. Класс FWriter	545
<i>Резюме</i>	546
<i>Ключевые термины</i>	547
<i>Итерация 1. Вопросы и упражнения</i>	547

Часть 3. Рефакторинг программного обеспечения и разработка пользовательского интерфейса 549

Глава 14. Требования к итерации 2 и объектная модель	551
14.1. Модель сценариев использования	551
14.2. Документ сценариев использования	554
14.2.1. Краткое описание, предусловия и постусловия	554
14.2.2. Основной поток	555
14.2.3. Подпотoki	556
14.2.4. Потoki исключений	561
14.3. Концептуальные классы и реляционные таблицы	562
14.4. Дополнительная спецификация	564
<i>Резюме</i>	566
<i>Ключевые термины</i>	566
<i>Обзорные вопросы</i>	566
Глава 15. Структурный рефакторинг	567
15.1. Цели рефакторинга	568
15.2. Методы рефакторинга	569
15.2.1. Класс извлечения	569
15.2.2. Метод подключения	571
15.2.3. Интерфейс извлечения	571
15.3. Паттерны рефакторинга	573
15.3.1. Коллекция идентичности объектов	575

15.3.2. Преобразователь данных	577
Загрузка — импорт	579
Выгрузка — экспорт	580
15.3.3. Альтернативные стратегии Преобразователя данных	580
Несколько Преобразователей данных	581
Преобразование метаданных	582
15.3.4. Загрузка по требованию	585
Инициализация по требованию	585
Виртуальный заместитель	586
Заместитель идентификатора объекта	589
Навигация по коллекции идентичности объектов	590
Навигация по классам пакета entity	592
15.3.5. Единица работы	594
15.4. Улучшенная модель классов	595
<i>Резюме</i>	596
<i>Ключевые термины</i>	599
<i>Обзорные вопросы</i>	600
<i>Вопросы для обсуждения</i>	600
<i>Вопросы учебного примера</i>	601
<i>Примеры задач</i>	601
Глава 16. Проектирование и программирование пользовательского интерфейса	602
16.1. Основные принципы проектирования пользовательского интерфейса	603
16.1.1. Пользователь в управлении	604
16.1.2. Непротиворечивость интерфейса	606
16.1.3. Снисходительность интерфейса	606
16.1.4. Адаптируемость интерфейса	607
16.2. Компоненты пользовательского интерфейса	608
16.2.1. Контейнеры	609
Управление расположением	612
Управление выбором уровней	614
16.2.2. Меню	615
16.2.3. Элементы управления	617
16.3. Управление событиями пользовательского интерфейса	619
16.4. Паттерны и пользовательский интерфейс	623
16.4.1. Наблюдатель	624
16.4.2. Декоратор	626
16.4.3. Цепочка обязанностей	626
16.4.4. Команда	628
16.5. Пользовательский интерфейс для управления электронной почтой	629
<i>Резюме</i>	633
<i>Ключевые термины</i>	634
<i>Обзорные вопросы</i>	635
<i>Примеры задач</i>	636
Глава 17. Проектирование и программирование пользовательского интерфейса на основе Web-технологии	638
17.1. Допустимые технологии для уровня Web-клиента	640
17.1.1. Основы HTML	640
17.1.2. Язык скриптов	643

17.1.3. Апплет: тонкий и толстый	645
17.2. Допустимые технологии для уровня Web-сервера	650
17.2.1. Сервлет	650
17.2.2. JSP	653
17.3. Транзакции Интернет-систем, не имеющих состояний	658
17.4. Паттерны и Web-технология	660
17.4.1. Наблюдатель	662
17.4.2. Компоновщик	662
17.4.3. Фабричный метод	663
17.4.4. Стратегия	664
17.4.5. Декоратор	665
17.4.6. Model-View-Controller (MVC)	665
17.4.7. Контроллер запросов	666
17.4.8. Повторное использование тегов в JSP	667
17.4.9. Несвязное управление: Struts	672
17.5. Реализация сервлета, обеспечивающего управление электронной почтой	673
<i>Резюме</i>	680
<i>Ключевые термины</i>	681
<i>Обзорные вопросы</i>	682
<i>Примеры задач</i>	683
Глава 18. Итерация 2. Аннотированный код	684
18.1. Обзор кода	684
18.2. Пакет Acquaintance	686
18.2.1. Интерфейс IAEmployee	687
18.3. Пакет Presentation	687
18.3.1. Класс PWindow	688
Конструирование и запуск PWindow	689
Извлечение данных в PWindow	691
Активизация фильтра	694
18.3.2. Класс PMessageDetailWindow	696
18.3.3. Класс PMessageTableModel	699
18.3.4. Класс PDisplayList	703
18.3.5. Класс PDisplayList.Filter	706
18.4. Пакет Control	708
18.4.1. Класс CAdmin	708
18.4.2. Класс CMsgSeeker	708
18.5. Пакет Entity	710
18.5.1. Класс Коллекция идентичности объектов	712
18.6. Пакет Mediator	714
18.6.1. Класс MModerator	715
18.6.2. Класс MDataMapper	716
Извлечение и загрузка исходящих сообщений	718
Сохранение и выгрузка исходящего сообщения	721
18.7. Уровень Presentation: версия апплета	724
18.8. Уровень Presentation: версия сервлета	726
18.8.1. Класс PEMS	727
Регистрационное имя в сервлете	728
Изображение исходящих сообщений в сервлете	730
18.8.2. Класс PEMSEdit	735

<i>Резюме</i>	737
<i>Ключевые термины</i>	738
<i>Итерация 2. Вопросы и упражнения</i>	738
Часть 4. Разработка данных и бизнес-компоненты	741
Глава 19. Требования к итерации 3 и объектная модель	744
19.1. Модель сценариев использования	744
19.2. Документ сценария использования	746
19.2.1. Краткое описание, предусловия и постусловия	746
19.2.2. Основной поток	747
19.2.3. Подпотoki	749
19.2.4. Потoki исключений	757
19.3. Концептуальные классы и реляционные таблицы	758
19.4. Дополнительная спецификация	760
19.5. Спецификация БД	763
<i>Резюме</i>	765
<i>Ключевые термины</i>	765
<i>Обзорные вопросы</i>	766
Глава 20. Безопасность и целостность	767
20.1. Проектирование безопасности	768
20.1.1. Контролируемая авторизация	769
Системные и объектные полномочия	780
Программная контролируемая авторизация	772
20.1.2. Принудительная авторизация	779
20.1.3. Авторизация предприятия	781
20.2. Проектирование целостности	785
20.2.1. Null-ограничение и ограничение по умолчанию	785
20.2.2. Ограничения «домен» и «проверка»	786
20.2.3. Уникальный и первичный ключи	787
20.2.4. Внешние ключи	788
20.2.5. Триггеры	790
20.3. Безопасность и целостность в управлении электронной почтой	795
20.3.1. Безопасность в управлении электронной почтой	795
Явно заданная таблица авторизации	798
Использование индивидуальных схем, глобальной схемы и хранимых процедур	799
Использование индивидуальных схем, глобальной схемы, представлений и хранимых процедур	800
Администрирование авторизации	803
20.3.2. Целостность управления электронной почтой	805
<i>Резюме</i>	808
<i>Ключевые термины</i>	809
<i>Обзорные вопросы</i>	810
<i>Примеры задач</i>	811
Глава 21. Транзакции и параллелизм	812
21.1. Параллелизм в системных транзакциях	813
21.1.1. ACID-свойства	814
21.1.2. Уровни изоляции	816
21.1.3. Способы блокировки и уровни блокировки	817

21.1.4. Модели транзакций	819
21.1.5. Схемы управления параллелизмом	821
21.2. Параллелизм в бизнес-транзакциях	825
21.2.1. Контексты выполнения бизнес-транзакций	825
21.2.2. Бизнес-транзакции и технология компонентов	826
21.2.3. Распределение по уровням сервисов транзакции	826
Web-уровень	828
Уровень приложения	828
Уровень БД	830
21.2.4. Паттерны автономного параллелизма	832
Единица работы	832
Оптимистическая автономная блокировка	835
Пессимистическая автономная блокировка	836
21.3. Транзакции и параллелизм в управлении электронной почтой	837
21.3.1. Модель плоской транзакции	838
21.3.2. Единица работы и поддержка транзакций	838
<i>Резюме</i>	839
<i>Ключевые термины</i>	842
<i>Обзорные вопросы</i>	843
<i>Примеры задач</i>	844
Глава 22. Бизнес-компоненты	846
22.1. Enterprise JavaBeans	847
22.1.1. Основные принципы EJB	849
22.1.2. Bean-компоненты сущностей	853
22.1.3. Bean-компоненты сеанса	858
22.2. Бизнес-компоненты для Java	860
22.2.1. Создание компонентов сущностей	860
XML для компонентов сущности	861
Java для компонентов сущности	863
22.2.2. Создание компонентов-представлений	864
XML для компонентов-представлений	865
Java для компонентов-представлений	866
22.2.3. Создание модуля приложения	867
<i>Резюме</i>	867
<i>Ключевые термины</i>	869
<i>Обзорные вопросы</i>	869
Глава 23. Итерация 3. Аннотированный код	871
23.1. Обзор кода	871
23.2. Пакет Acquaintance	873
23.2.1. Интерфейс IAReportEntry	874
23.3. Пакет Presentation	874
23.3.1. Класс PWindow	874
Заполнение списка деловых партнеров в отчете	875
Окно отчета	876
Отчет о деятельности	878
Печать отчета	879
Заполнение таблицы отчета	879
Отображение окна авторизации	881
Преобразование из матрицы правил в таблицу авторизации	883
Сохранение измененных прав доступа	884

Преобразование из таблицы авторизации в матрицу правил	884
Удаление исходящего сообщения	886
Изменение исходящего сообщения	888
Создание исходящего сообщения	889
23.3.2. Класс PTableWindow	889
Динамическая регистрация кнопок	890
Добавление приемников к динамически сформированным кнопкам	891
Возвращаемое состояние кнопки	892
Печать в PTableWindow	893
23.4. Пакет Control	894
23.5. Пакет Entity	894
23.5.1. Класс EIdentityMap	894
Регистрация и удаление отчета	896
Извлечение отчета	896
23.6. Пакет mediator	899
23.6.1. Класс MModerator	900
Права доступа	900
Извлечение отчета	902
Создание исходящего сообщения	904
Корректировка исходящего сообщения	904
23.6.2. Класс MDataMapper	905
Изменения в существовавших методах	907
Извлечение отчета в MDataMapper	908
Загрузка прав доступа в MDataMapper	910
Сохранение прав доступа в MDataMapper	910
23.6.3. Класс MUnitOfWork	913
Получение MUnitOfWork	914
Регистрация новой сущности в MUnitOfWork	915
Регистрация измененной сущности в MUnitOfWork	916
Удаление сущности в MUnitOfWork	916
Фиксация MUnitOfWork	917
Выполнение транзакции	918
Начало транзакции	919
23.7. Пакет Foundation	920
23.7.1. Транзакции в FConnection	920
23.7.2. Операторы Execute в FWriter	921
23.7.3. Запрос к БД в FReader	923
23.8. Код БД	924
23.8.1. Ref Cursor для ResultSet	925
23.8.2. Извлечение исходящих сообщений	926
23.8.3. Извлечение исходящих сообщений отдела	926
23.8.4. Удаление исходящего сообщения	927
23.8.5. Создание исходящего сообщения	928
23.8.6. Создание отчета	930
23.8.7. Триггер для таблицы OutMessage	932
<i>Резюме</i>	934
<i>Ключевые термины</i>	935
<i>Итерация 3. Вопросы и упражнения</i>	935
Литература	937
Предметный указатель	943

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 разработки¹.

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 инъекцией 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 АЕМ (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 АЕМ.

Как показано на диаграмме Венна, АЕМ занимается бизнесом, CM является одной из предметных областей бизнеса, а EM является учебным примером. Упражнения и небольшие проекты взяты из срезов областей АЕМ-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 манипуляции 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 EM (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 вы добавок 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. Рискую 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.thsmallworlds.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

- Глава 1. Жизненный цикл разработки программного обеспечения**
- Глава 2. Язык моделирования программного обеспечения**
- Глава 3. Инструментальные средства программной инженерии**
- Глава 4. Планирование и отслеживание проекта программного обеспечения**
- Глава 5. Управление процессом создания и отслеживания 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 системе, 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.

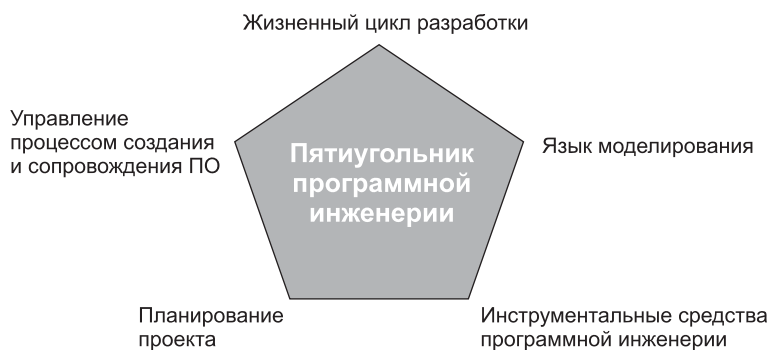
Большинство проектов ПО выполняется в связи с организационными потребностями заняться задачами идентификации в бизнес-процессе или улучшением этого процесса из-за конкуренции. *Проектирование ПО* — запланированная операция, предназначенная для создания программного продукта или сервиса и выполняющаяся 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.

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 образом, 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 «перевосплощениями». 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

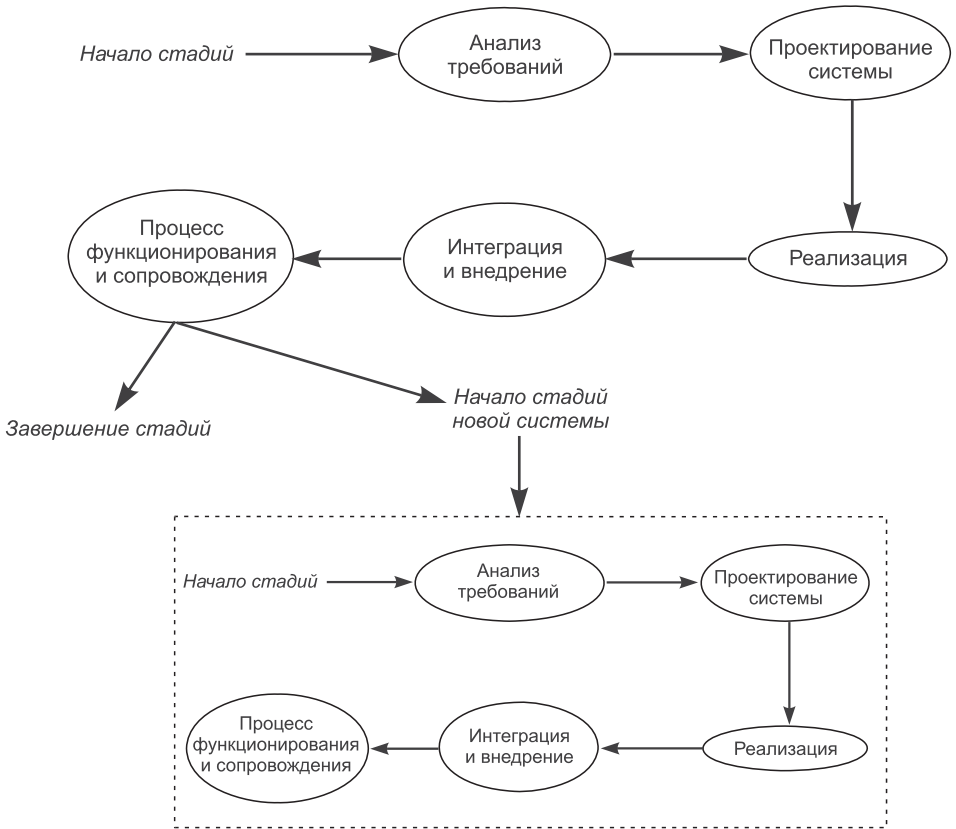


Рис. 1.1. Стадии жизненного цикла ПО

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:

[. . .]

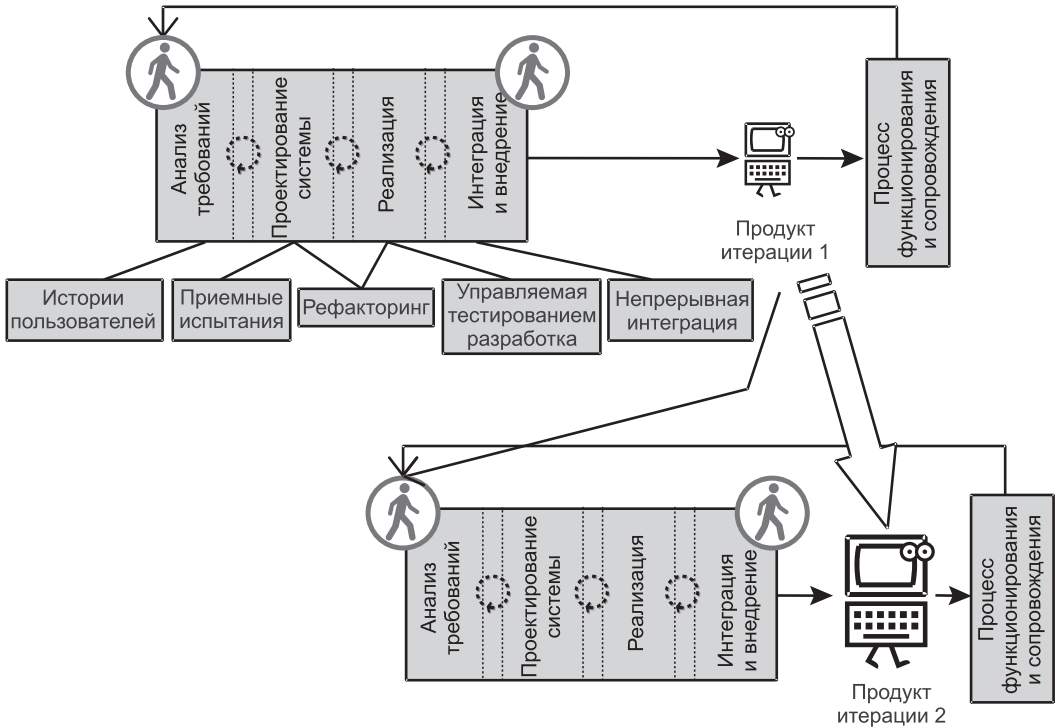


Рис. 1.12. Быстрая разработка ПО

«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 слеplено 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

CASE	См. computer assisted software engineering	бизнес-процесс	39
computer assisted software engineering	49	быстрая разработка ПО	65
eXtreme Programming	67	версия	43
formal systems development	63	внедрение	52
IDE	См. integrated development environment	возможность сопровождения	42
integrated development environment	51	выполнимые спецификации	63
MDA	См. Model Driven Architecture	гарантия качества ПО	49
Model Driven Architecture	63	жизненный цикл	36
OLAP	См. online analytical processing	жизненный цикл ПО	48
OLTP	См. online transaction processing	инженерия систем.	41
online analytical processing	40	интеграция	52
online transaction processing	40	интегрированные средства разработки	51
Rational Unified Process®	62	информационная система	38
RUP®	См. Rational Unified Process®	информационная система предприятия	38
software quality assurance	49	испытательная заглушка	53
SQA	См. software quality assurance	итеративный жизненный цикл	59
UML	См. Unified Modeling Language	итерация	59
Unified Modeling Language	43	компонент ПО	43
XP	См. eXtreme Programming	конструкция.	59
абстракция	44, 47	конфигурация	43
автоматизированная программная инженерия	49	модель.	44
анализ требований	48	модель водопада	56
аналитическая обработка в реальном времени	40	модель детального проекта	44
архитектура, управляемая моделями	63	модель жизненного цикла	55
		модель программного продукта	44
		модель программы	45
		модель процесса создания ПО	44

модель с опытными образцами	57	спиральная модель	60
модель спецификаций	44	средства тестирования	54
модель требований	44	стадии жизненного цикла	36
объектно-ориентированный подход	45	структурная модель	44
оперативная обработка транзакций	40	тестирование	52
определение требований	48	тестирование интеграции	53
опытный образец ПО	57	тестирование на основе кода	52
отладка	52	тестирование на основе технических требова- ний	52
парное программирование	66	тестовый драйвер	54
приемочные испытания	54	технические требования	48
программирование	43	техническое задание	49
программная инженерия	37	технология компонентов	64
проектирование детальное	50	традиционная инженерия	41
проектирование ПО	50	требования пользователя	48
проектирование систем	50	унаследованная система	36, 55
проектирование структурное	50	унифицированный язык моделирования	43
проектирование циклическое	43, 51	управление проектом	43
процесс создания и эксплуатации ПО	39	управление трассировкой	51
процесс функционирования	54	управляемая тестированием разработка	66
рациональный унифицированный процесс реализация	62	уровень управления	40
результативность	39	формальная разработка системы	63
рефакторинг	66	функциональный подход	45
система	38	шаг	59
сопровождение	54	эффективность	39
спецификация требований	49		

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.