

Содержание

| | |
|--|-----------|
| Предисловие | 19 |
| Введение | 21 |
| Образовательные и Web-ресурсы | 22 |
| Для кого предназначена эта книга | 22 |
| Что необходимо знать | 22 |
| Примеры на языке Java | 22 |
| Структура книги | 23 |
| Об авторе | 23 |
| Контакты | 24 |
| Дополнения к предыдущему изданию | 24 |
| Благодарности | 25 |
| Соглашения, принятые в книге | 25 |
| От издательства | 26 |
| Часть I. Введение | 27 |
| Глава 1. Объектно-ориентированный анализ и проектирование | 29 |
| 1.1. Что вы узнаете из этой книги? | 29 |
| 1.2. Основная цель обучения | 32 |
| 1.3. Что такое анализ и проектирование | 33 |
| 1.4. Объектно-ориентированный анализ и проектирование | 33 |
| 1.5. Краткий пример | 34 |
| 1.6. Что такое UML | 37 |
| 1.7. Визуальное моделирование — хороший стиль | 41 |
| 1.8. История | 41 |
| 1.9. Дополнительные ресурсы | 43 |
| Глава 2. Итеративный, эволюционный и гибкий процесс | 45 |
| 2.1. Унифицированный процесс и другие методы | 46 |
| 2.2. Что такое итеративная и эволюционная разработка | 48 |
| 2.3. Последовательный жизненный цикл | 52 |
| 2.4. Как выполнять итеративный и эволюционный анализ и проектирование | 54 |
| 2.5. Итеративное планирование на основе рисков и с учетом потребностей пользователей | 56 |
| 2.6. Что такое гибкие методы | 58 |
| 2.7. Что такое гибкое моделирование | 59 |
| 2.8. Что такое гибкий унифицированный процесс | 62 |
| 2.9. Другие важные принципы унифицированного процесса | 63 |
| 2.10. Фазы унифицированного процесса | 63 |
| 2.11. Дисциплины унифицированного процесса | 64 |
| 2.12. Настройка процесса и выбор инструментов UP | 67 |

| | |
|---|----|
| 2.13. Вы не поняли, что такое итеративная разработка или унифицированный процесс, если... | 69 |
| 2.14. История | 69 |
| 2.15. Дополнительные ресурсы | 70 |

Глава 3. Конкретные примеры 73

| | |
|---|----|
| 3.1. Какие вопросы рассматриваются и <i>не</i> рассматриваются при изучении конкретных примеров | 74 |
| 3.2. Стратегия описания конкретных примеров: итеративная разработка + итеративное изучение | 75 |
| 3.3. Пример 1: POS-система NextGen | 76 |
| 3.4. Пример 2: игра Monopoly® | 77 |

Часть II. Начальная фаза 79

Глава 4. Начальная фаза — это не стадия анализа требований 81

| | |
|--|----|
| 4.1. Что такое начальная фаза | 82 |
| 4.2. Какова длительность начальной фазы | 83 |
| 4.3. Какие артефакты относятся к начальной фазе | 83 |
| 4.4. Вы не поняли, что такое начальная фаза, если... | 85 |
| 4.5. Как использовать UML на начальной стадии | 85 |

Глава 5. Эволюционное осмысление требований 87

| | |
|---|----|
| 5.1. Определение требований | 88 |
| 5.2. Определение требований для эволюционного и каскадного процессов разработки | 88 |
| 5.3. Грамотное определение требований | 90 |
| 5.4. Типы и категории требований | 90 |
| 5.5. Как требования связаны с артефактами UP | 91 |
| 5.6. Имеются ли примеры этих артефактов в данной книге | 92 |
| 5.7. Дополнительные ресурсы | 92 |

Глава 6. Описание прецедентов 95

| | |
|--|-----|
| 6.1. Пример | 96 |
| 6.2. Некоторые определения | 96 |
| 6.3. Прецеденты и модель прецедентов | 98 |
| 6.4. Зачем нужны описания прецедентов | 99 |
| 6.5. Прецеденты и функциональные требования | 100 |
| 6.6. Три типа исполнителей | 100 |
| 6.7. Основные форматы прецедентов | 101 |
| 6.8. Пример развернутого описания прецедента Оформление продажи | 101 |
| 6.9. Пояснения | 107 |
| 6.10. Другие форматы описания прецедентов | 113 |
| 6.11. Описание прецедентов, относящихся к интерфейсу пользователя, в свободном стиле | 114 |
| 6.12. Описания прецедентов должны быть лаконичными | 116 |

| | |
|--|-----|
| 6.13. Описание прецедентов в формате “черный ящик” | 116 |
| 6.14. Примите точку зрения исполнителя | 116 |
| 6.15. Как выделять прецеденты | 117 |
| 6.16. Как выделить полезные прецеденты | 121 |
| 6.17. Диаграммы прецедентов | 123 |
| 6.18. Диаграммы видов деятельности | 125 |
| 6.19. Дополнительные преимущества описания прецедентов | 125 |
| 6.20. Пример: игра “Монополия” | 126 |
| 6.21. Прецеденты в рамках итеративной разработки | 128 |
| 6.22. История | 133 |
| 6.23. Дополнительные ресурсы | 133 |

Глава 7. Другие требования **135**

| | |
|---|-----|
| 7.1. Насколько полны приводимые примеры? | 136 |
| 7.2. Нужно ли детально анализировать требования на начальной стадии разработки? | 136 |
| 7.3. Нужно ли помещать документацию на Web-узле проекта? | 137 |
| 7.4. Пример NextGen: фрагмент дополнительной спецификации | 137 |
| 7.5. Дополнительная спецификация (комментарии) | 140 |
| 7.6. Пример для системы NextGen: видение (фрагмент) | 142 |
| 7.7. Видение (комментарий) | 145 |
| 7.8. Пример системы NextGen: словарь терминов (фрагмент) | 148 |
| 7.9. Комментарии: словарь терминов | 149 |
| 7.10. Пример для системы NextGen: бизнес-правила (правила предметной области) | 150 |
| 7.11. Бизнес-правила (комментарии) | 151 |
| 7.12. Эволюция требований в итеративном процессе разработки | 151 |
| 7.13. Дополнительные ресурсы | 153 |

Часть III. Первая итерация фазы развития — основы **155**

Глава 8. Итерация 1 — основы **157**

| | |
|---|-----|
| 8.1. Требования и акценты первой итерации: основные вопросы ООА/П | 158 |
| 8.2. Начальная фаза и стадия развития | 160 |
| 8.3. Планирование следующей итерации | 163 |

Глава 9. Модели предметной области **165**

| | |
|---|-----|
| 9.1. Пример | 167 |
| 9.2. Что такое модель предметной области | 168 |
| 9.3. Зачем создавать модель предметной области | 171 |
| 9.4. Как создать модель предметной области | 172 |
| 9.5. Как выделить концептуальные классы | 173 |
| 9.6. Пример выделения концептуальных классов | 176 |
| 9.7. Построение диаграммы классов в рамках гибкого процесса разработки | 177 |
| 9.8. Автоматизированное построение модели в рамках гибкого подхода к разработке | 177 |
| 9.9. Элементы отчета: включать ли понятие “товарный чек” в модель | 178 |

| | |
|---|-----|
| 9.10. Имена и модели: стратегия построения карт | 178 |
| 9.11. Как моделировать “нереальный” мир | 179 |
| 9.12. Типичная ошибка при выделении концептуальных классов | 179 |
| 9.13. Когда использовать классы описаний | 180 |
| 9.14. Ассоциации | 182 |
| 9.15. Примеры: ассоциации в моделях предметных областей | 189 |
| 9.16. Атрибуты | 190 |
| 9.17. Атрибуты в моделях предметных областей | 198 |
| 9.18. Насколько корректна модель предметной области | 199 |
| 9.19. Итеративное и эволюционное моделирование предметной области | 200 |
| 9.20. Дополнительные ресурсы | 201 |

Глава 10. Системные диаграммы последовательностей 203

| | |
|--|-----|
| 10.1. Пример СДП для приложения NextGen | 205 |
| 10.2. Системные диаграммы последовательностей | 206 |
| 10.3. Зачем строить СДП | 206 |
| 10.4. Применение UML для построения диаграммы последовательности | 207 |
| 10.5. Системные диаграммы последовательностей и прецеденты | 207 |
| 10.6. Имена системных событий и операций | 208 |
| 10.7. Как моделировать взаимодействие с внешними системами | 208 |
| 10.8. Диаграммы последовательностей и словарь терминов | 209 |
| 10.9. Пример диаграммы последовательности для игры “Монополия” | 209 |
| 10.10. Итеративное и эволюционное построение СДП | 210 |
| 10.11. История и дополнительные ресурсы | 210 |

Глава 11. Описание операций 211

| | |
|---|-----|
| 11.1. Пример | 213 |
| 11.2. Разделы описания | 213 |
| 11.3. Что такое системная операция | 213 |
| 11.4. Постусловия | 214 |
| 11.5. Постусловия описания операции enterItem | 217 |
| 11.6. Описание операций приводит к изменению предметной области | 217 |
| 11.7. Когда нужны описания операций? | 218 |
| 11.8. Составление описания | 218 |
| 11.9. Пример описания операции для POS-системы NextGen | 219 |
| 11.10. Описание операций для игры “Монополия” | 221 |
| 11.11. Описания операций на UML и OCL | 221 |
| 11.12. Описания операций в рамках UP | 222 |
| 11.13. Немного истории | 223 |
| 11.14. Дополнительные ресурсы | 223 |

Глава 12. Итеративный переход от анализа требований к проектированию 225

| | |
|--|-----|
| 12.1. Занимаясь итеративной разработкой, делайте это правильно | 226 |
| 12.2. Раннее начало всех видов деятельности | 226 |
| 12.3. Возможно, на это потребуются недели? | 226 |

| | |
|--|------------|
| Глава 13. Логическая архитектура и диаграммы пакетов UML | 227 |
| 13.1. Пример | 228 |
| 13.2. Что такое логическая архитектура | 228 |
| 13.3. На каких уровнях сосредоточено описание примеров? | 230 |
| 13.4. Программная архитектура системы | 231 |
| 13.5. Диаграммы пакетов UML | 231 |
| 13.6. Проектирование на основе шаблона Layers | 232 |
| 13.7. Принцип Model-View Separation | 238 |
| 13.8. Взаимосвязь между СДП, системными операциями и уровнями | 240 |
| 13.9. Логическая архитектура и диаграмма пакетов для приложения NextGen | 241 |
| 13.10. Пример логической архитектуры игры “Монополия” | 241 |
| 13.11. Дополнительные ресурсы | 242 |
| Глава 14. Переход к объектному проектированию | 243 |
| 14.1. Гибкое моделирование и облегченный процесс построения диаграмм UML | 244 |
| 14.2. CASE-средства UML | 245 |
| 14.3. Сколько времени занимает рисование до начала кодирования | 245 |
| 14.4. Статические и динамические модели объектов | 246 |
| 14.5. Важнее иметь навыки объектного проектирования, чем знать систему обозначений UML | 247 |
| 14.6. Другие методы проектирования объектов: карты CRC | 248 |
| Глава 15. Диаграммы взаимодействия на UML | 251 |
| 15.1. Диаграммы последовательностей и коммуникации | 252 |
| 15.2. Проблема для новичков | 255 |
| 15.3. Основные обозначения UML для диаграмм взаимодействия | 255 |
| 15.4. Основные обозначения диаграммы последовательностей | 257 |
| 15.5. Основные обозначения диаграммы коммуникации | 269 |
| Глава 16. Диаграммы классов UML | 277 |
| 16.1. Применение UML: система обозначений для диаграммы классов | 278 |
| 16.2. Определение: диаграмма классов проектирования | 278 |
| 16.3. Определение: классификатор | 280 |
| 16.4. Способы представления атрибутов UML: имя атрибута и линии ассоциаций | 280 |
| 16.5. Блоки примечаний: примечания, комментарии, ограничения и тела методов | 284 |
| 16.6. Операции и методы | 285 |
| 16.7. Ключевые слова | 286 |
| 16.8. Стереотипы, профили и метки | 287 |
| 16.9. Свойства и строки свойств | 288 |
| 16.10. Обобщение, абстрактные классы, абстрактные операции | 288 |
| 16.11. Зависимость | 289 |
| 16.12. Интерфейсы | 291 |
| 16.13. Композиция или агрегация | 292 |
| 16.14. Ограничения | 293 |
| 16.15. Уточненная ассоциация | 294 |
| 16.16. Класс ассоциации | 294 |

| | |
|---|-----|
| 16.17. Классы-синглетоны | 295 |
| 16.18. Параметризованные классы и интерфейсы | 295 |
| 16.19. Поля, определяемые пользователем | 296 |
| 16.20. Активный класс | 297 |
| 16.21. Связь между диаграммами классов и взаимодействия | 297 |

Глава 17. GRASP: проектирование объектов на основе распределения обязанностей 299

| | |
|---|-----|
| 17.1. UML и принципы проектирования | 300 |
| 17.2. Объектное проектирование – примеры входов, видов деятельности и выходов | 300 |
| 17.3. Обязанности и проектирование на основе обязанностей | 304 |
| 17.4. GRASP – это методический подход к объектному проектированию | 305 |
| 17.5. Обязанности, GRASP и диаграммы UML | 306 |
| 17.6. Шаблоны | 306 |
| 17.7. Где мы находимся? | 309 |
| 17.8. Небольшой пример объектного проектирования на основе шаблонов GRASP | 309 |
| 17.9. Применение шаблонов GRASP для объектного проектирования | 318 |
| 17.10. Шаблон Creator | 319 |
| 17.11. Шаблон Information Expert (или Expert) | 321 |
| 17.12. Шаблон Low Coupling | 326 |
| 17.13. Шаблон Controller | 329 |
| 17.14. Шаблон High Cohesion | 341 |
| 17.15. Дополнительные ресурсы | 345 |

Глава 18. Примеры объектного проектирования на основе шаблонов GRASP 347

| | |
|---|-----|
| 18.1. Реализация прецедентов | 348 |
| 18.2. Комментарии к артефактам | 349 |
| 18.3. Что дальше? | 353 |
| 18.4. Реализация прецедентов для данной итерации разработки системы NextGen | 353 |
| 18.5. Реализация прецедентов для игры “Монополия” | 373 |
| 18.6. Итеративное и эволюционное объектное проектирование | 383 |
| 18.7. Резюме | 385 |

Глава 19. Области видимости 387

| | |
|--------------------------|-----|
| 19.1. Видимость объектов | 388 |
| 19.2. Области видимости | 389 |

Глава 20. Преобразование проектного решения в программный код 393

| | |
|---|-----|
| 20.1. Программирование и итеративный, эволюционный процесс разработки | 394 |
| 20.2. Преобразование результатов проектирования в программный код | 395 |
| 20.3. Создание определений классов на основе диаграмм классов | 395 |
| 20.4. Создание методов на основе диаграмм взаимодействия | 396 |
| 20.5. Классы коллекций в программном коде | 398 |
| 20.6. Исключения и обработка ошибок | 399 |

| | |
|---|------------|
| 20.7. Определение метода <code>Sale.makeLineItem</code> | 399 |
| 20.8. Порядок реализации | 400 |
| 20.9. Программирование на основе тестирования | 400 |
| 20.10. Еще несколько слов о преобразовании проектного решения в код | 401 |
| 20.11. Основное программное решение для POS-системы NextGen | 401 |
| 20.12. Основные программные решения для игры “Монополия” | 404 |
| Глава 21. Разработка на основе тестирования и рефакторинг | 409 |
| 21.1. Разработка на основе тестирования | 410 |
| 21.2. Рефакторинг | 413 |
| 21.3. Дополнительные ресурсы | 416 |
| Часть IV. Вторая итерация фазы развития — дополнительные шаблоны | 419 |
| Глава 22. Средства построения диаграмм UML | 421 |
| 22.1. Прямое, обратное и циклическое проектирование | 422 |
| 22.2. Важнейшие свойства UML-средств | 423 |
| 22.3. Как выбрать систему проектирования | 423 |
| 22.4. Как обновлять диаграммы на основе кода, если они изначально были построены вручную | 424 |
| 22.5. Дополнительные ресурсы | 424 |
| Глава 23. Быстрый дополнительный анализ | 425 |
| 23.1. POS-система NextGen | 426 |
| 23.2. Игра “Монополия” | 428 |
| Глава 24. Вторая итерация и дополнительные шаблоны | 431 |
| 24.1. От первой ко второй итерации | 432 |
| 24.2. Требования для второй итерации | 433 |
| Глава 25. Дополнительные шаблоны GRASP для распределения обязанностей | 437 |
| 25.1. Шаблон Polymorphism | 438 |
| 25.2. Шаблон Pure Fabrication | 445 |
| 25.3. Шаблон Indirection | 449 |
| 25.4. Шаблон Protected Variations | 451 |
| Глава 26. Применение шаблонов проектирования GoF | 459 |
| 26.1. Шаблон Adapter (GoF) | 460 |
| 26.2. Некоторые принципы GRASP как обобщение других шаблонов | 462 |
| 26.3. Анализ на этапе проектирования: модель предметной области | 463 |
| 26.4. Шаблон Factory (GoF) | 464 |
| 26.5. Шаблон Singleton (GoF) | 466 |

| | |
|---|-----|
| 26.6. Еще несколько слов о внешних службах с разными интерфейсами | 469 |
| 26.7. Шаблон Strategy (GoF) | 471 |
| 26.8. Шаблон Composite (GoF) и другие принципы проектирования | 475 |
| 26.9. Шаблон Facade (GoF) | 483 |
| 26.10. Шаблон Observer/Publish-Subscribe/Delegation Event Model (GoF) | 485 |
| 26.11. Выводы | 493 |
| 26.12. Дополнительная литература | 494 |

Часть V. Третья итерация фазы развития **495**

Глава 27. Третья итерация – вспомогательные вопросы **497**

| | |
|---|-----|
| 27.1. Требования третьей итерации для POS-системы NextGen | 498 |
| 27.2. Требования третьей итерации для игры “Монополия” | 498 |

Глава 28. Диаграммы видов деятельности UML **499**

| | |
|---|-----|
| 28.1. Пример | 500 |
| 28.2. Как применять диаграммы видов деятельности | 501 |
| 28.3. Дополнительные обозначения диаграмм видов деятельности UML | 503 |
| 28.4. Рекомендации | 503 |
| 28.5. Пример: диаграмма видов деятельности для приложения NextGen | 504 |
| 28.6. Диаграммы видов деятельности в UP | 505 |
| 28.7. Другие подходы | 506 |

Глава 29. Диаграммы состояний UML и моделирование **507**

| | |
|--|-----|
| 29.1. Пример | 508 |
| 29.2. События, состояния и переходы | 508 |
| 29.3. Как применять диаграммы состояний | 509 |
| 29.4. Дополнительные обозначения для диаграмм состояний | 511 |
| 29.5. Моделирование навигации с помощью диаграммы состояний | 512 |
| 29.6. Пример: диаграмма состояний для прецедента системы NextGen | 513 |
| 29.7. Диаграммы состояний и UP | 513 |
| 29.8. Дополнительная литература | 513 |

Глава 30. Взаимосвязь прецедентов **515**

| | |
|--|-----|
| 30.1. Отношение включает | 516 |
| 30.2. Новые термины: конкретный, абстрактный, основной и дополнительный прецеденты | 519 |
| 30.3. Отношение расширяет | 520 |
| 30.4. Отношение обобщает | 521 |
| 30.5. Диаграммы прецедентов | 521 |

Глава 31. Дополнительные диаграммы последовательностей и описание операций **523**

| | |
|--------------------------|-----|
| 31.1. Приложение NextGen | 524 |
|--------------------------|-----|

| | |
|--|------------|
| Глава 32. Уточнение модели предметной области | 529 |
| 32.1. Новые понятия модели предметной области системы NextGen | 530 |
| 32.2. Обобщение | 532 |
| 32.3. Определение концептуальных суперклассов и подклассов | 533 |
| 32.4. Когда нужно определять концептуальный подкласс | 536 |
| 32.5. Когда нужно определять концептуальный суперкласс | 538 |
| 32.6. Иерархия концептуальных классов POS-системы NextGen | 538 |
| 32.7. Абстрактные концептуальные классы | 541 |
| 32.8. Моделирование изменения состояний | 542 |
| 32.9. Иерархия классов и наследование | 543 |
| 32.10. Классы ассоциаций | 543 |
| 32.11. Агрегация и композиция | 546 |
| 32.12. Временные интервалы и цены товаров: устранение ошибки первой итерации | 548 |
| 32.13. Имена ролей ассоциаций | 549 |
| 32.14. Роли в форме понятий и роли, представленные в ассоциации | 550 |
| 32.15. Производные элементы | 551 |
| 32.16. Составные ассоциации | 552 |
| 32.17. Рефлексивные ассоциации | 553 |
| 32.18. Использование пакетов для организации элементов модели предметной области | 553 |
| 32.19. Уточненная модель предметной области для игры “Монополия” | 559 |
| Глава 33. Архитектурный анализ | 561 |
| 33.1. Когда нужно приступать к архитектурному анализу | 562 |
| 33.2. Точки вариации и эволюции | 562 |
| 33.3. Архитектурный анализ | 563 |
| 33.4. Общие методы архитектурного анализа | 564 |
| 33.5. Определение и анализ архитектурных факторов | 565 |
| 33.6. Пример: фрагмент таблицы архитектурных факторов POS-системы NextGen | 568 |
| 33.7. Определение архитектурных факторов | 570 |
| 33.8. Выводы по архитектурному анализу | 575 |
| 33.9. Итеративный архитектурный анализ в UP | 576 |
| 33.10. Дополнительная литература | 577 |
| Глава 34. Уточнение логической архитектуры | 579 |
| 34.1. Пример: логическая архитектура приложения NextGen | 580 |
| 34.2. Взаимодействия на основе шаблона Layers | 585 |
| 34.3. Другие вопросы использования шаблона Layers | 591 |
| 34.4. Принцип Model-View Separation | 596 |
| 34.5. Дополнительная литература | 597 |
| Глава 35. Новые проектные решения на основе шаблонов GoF | 599 |
| 35.1. Пример: POS-система NextGen | 600 |
| 35.2. Переход к локальным службам и обеспечение локальной буферизации | 600 |
| 35.3. Обработка отказов | 604 |

| | |
|--|-----|
| 35.4. Взаимодействие с локальными службами на основе шаблона Proxy (GoF) | 611 |
| 35.5. Реализация нефункциональных или качественных требований | 615 |
| 35.6. Доступ к внешним физическим устройствам с помощью шаблона Adapter | 615 |
| 35.7. Шаблон Abstract Factory (GoF) для семейства взаимосвязанных объектов | 617 |
| 35.8. Обработка платежей на основе шаблонов Polymorphism и Do It Myself | 621 |
| 35.9. Пример: игра “Монополия” | 627 |
| 35.10. Заключение | 630 |

Глава 36. Проектирование на основе пакетов 631

| | |
|---|-----|
| 36.1. Рекомендации по организации пакетов | 632 |
| 36.2. Дополнительная литература | 637 |

Глава 37. Диаграммы развертывания и компонентов UML 639

| | |
|-------------------------------|-----|
| 37.1. Диаграммы развертывания | 639 |
| 37.2. Диаграммы компонентов | 641 |

Глава 38. Проектирование каркаса взаимодействия с базой данных на основе шаблонов 643

| | |
|--|-----|
| 38.1. Проблема: объекты, подлежащие постоянному хранению | 644 |
| 38.2. Решение: каркас интерфейса с базой данных | 645 |
| 38.3. Каркасы | 645 |
| 38.4. Требования к каркасу интерфейса с базой данных | 646 |
| 38.5. Основные принципы | 647 |
| 38.6. Шаблон представления объектов в виде таблиц | 647 |
| 38.7. Профиль моделирования данных UML | 648 |
| 38.8. Шаблон Object Identifier | 648 |
| 38.9. Доступ к службе взаимодействия с базой данных на основе шаблона Facade | 649 |
| 38.10. Объекты-преобразователи: шаблон Database Mapper или Database Broker | 650 |
| 38.11. Разработка каркаса на основе шаблона Template Method | 652 |
| 38.12. Материализация на основе шаблона Template Method | 654 |
| 38.13. Настройка преобразователей с помощью объекта MapperFactory | 659 |
| 38.14. Шаблон Cache Management | 659 |
| 38.15. Объединение и сокрытие операторов SQL в одном классе | 660 |
| 38.16. Состояние транзакции и шаблон State | 661 |
| 38.17. Обработка транзакций на основе шаблона Command | 664 |
| 38.18. Пассивная материализация на основе шаблона Virtual Proxy | 667 |
| 38.19. Представление отношений в таблицах | 670 |
| 38.20. Суперкласс PersistentObject | 670 |
| 38.21. Нерешенные вопросы | 671 |

Глава 39. Документирование архитектуры с помощью N+1 представления 673

| | |
|--|-----|
| 39.1. Документ SAD и архитектурные представления | 674 |
| 39.2. Структура документа SAD | 677 |
| 39.3. Пример: описание архитектуры POS-системы NextGen | 677 |
| 39.4. Пример: документ SAD для каркаса Struts | 682 |

| | |
|--|------------|
| 39.5. Итеративное документирование архитектуры | 686 |
| 39.6. Дополнительная литература | 686 |
| Часть VI. Специальные вопросы | 687 |
| Глава 40. Еще раз об итеративной разработке и гибком управлении проектом | 689 |
| 40.1. Как спланировать итерацию | 690 |
| 40.2. Адаптивное и предиктивное планирование | 690 |
| 40.3. Планы для фазы и итерации | 692 |
| 40.4. Как составить план итерации с учетом прецедентов и сценариев | 693 |
| 40.5. Приблизительность начальных оценок | 695 |
| 40.6. Организация артефактов проекта | 695 |
| 40.7. Вы не поняли принципов итеративного планирования, если... | 696 |
| 40.8. Дополнительная литература | 697 |
| Приложение А. Артефакты унифицированного процесса, шаблоны GRASP и условные обозначения языка UML | 699 |
| Приложение Б. Словарь терминов | 705 |
| Литература | 711 |
| Предметный указатель | 717 |

Конкретные примеры

Существует совсем немного вещей, которые придумать труднее, чем хороший пример.

Марк Твен (Mark Twain)

Основная задача

- Ознакомиться с конкретными примерами, рассматриваемыми в этой книге.

Введение

Рассматриваемые в этой книге конкретные примеры выбраны именно потому, что многие знакомы с подобными задачами. В книге рассматриваются интересные и сложные проектные решения, позволяющие сконцентрироваться на фундаментальных принципах ООА/П, анализе требований, UML и шаблонах, а не на постановке задач и предметной области.

Что дальше?

После ознакомления с итеративным процессом в данной главе рассматриваются конкретные примеры и основные элементы уровня логики приложения, которые будут рассматриваться в книге. Следующая глава посвящена стадии начала разработки учебных приложений. При этом подчеркивается, что начало *не* является фазой каскадного процесса, на которой проводится ранний анализ всех требований



3.1. Какие вопросы рассматриваются и не рассматриваются при изучении конкретных примеров

Архитектура типичной информационной системы включает графический интерфейс пользователя, взаимодействие с базой данных, уровень логики приложения и взаимосвязь с другими программными системами или архитектурными компонентами (рис. 3.1).

Хотя объектно-ориентированную технологию можно применять ко всем уровням, в данной книге основное внимание уделяется вопросам анализа и проектирования базового уровня логики приложения, а другие уровни рассматриваются только поверхностно.

При рассмотрении других уровней (например, интерфейса пользователя) мы, в основном, сосредоточимся на разработке их интерфейса с уровнем логики приложения.

Почему основное внимание в процессе ООА/П уделяется базовому уровню логики приложения? На этот вопрос можно ответить следующим образом.

- Другие уровни обычно в значительной мере зависят от платформы или от реализации. Например, чтобы изучить вопросы объектно-ориентированного проектирования Web-интерфейса или интерфейса пользователя на Java, необходимо детально разобраться с такими каркасами, как Struts или Swing. Если же в качестве платформы разработки выбрана .NET или Python, базовые знания должны быть совершенно другими.
- В отличие от этого объектно-ориентированное проектирование уровня логики приложения одинаково для всех технологий.
- Основные знания и навыки, приобретенные в процессе изучения ООП в контексте уровня логики приложения, применимы ко всем другим уровням или компонентам.
- Подход к проектированию (в том числе на основе шаблонов) быстро изменяется с появлением новых каркасов или технологий. Например, в середине 1990-х разработчикам приходилось создавать свои собственные замороженные объектно-реляционные уровни доступа к базам данных. А спустя несколько лет они предпочли использовать свободно распространяемые решения с открытым кодом, такие как Hibernate (для технологии Java).

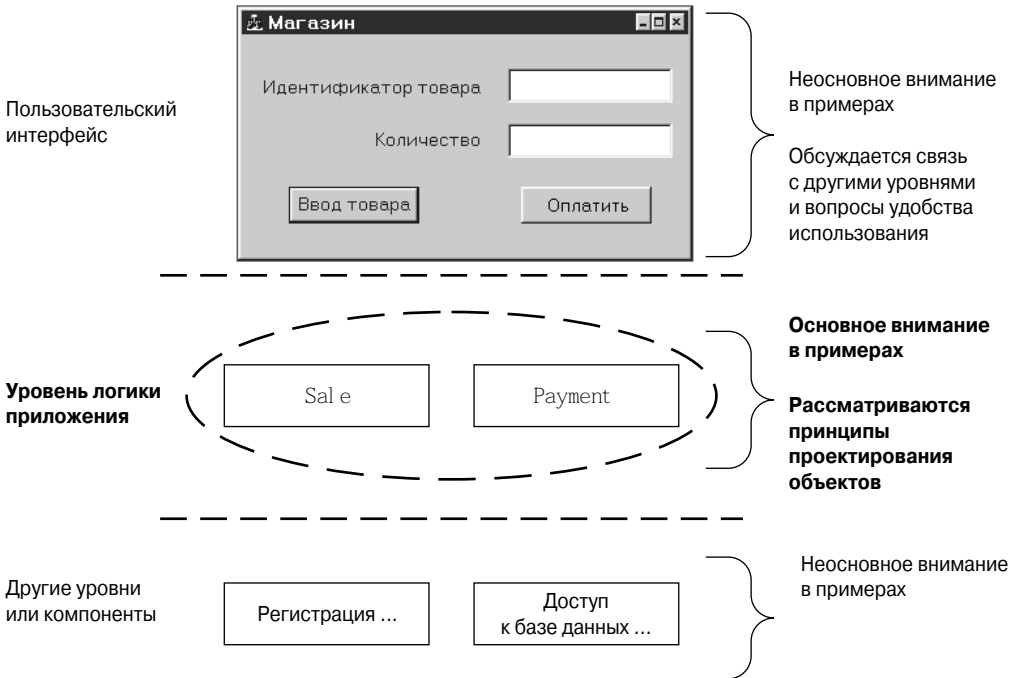


Рис. 3.1. Уровни и объекты типичной объектно-ориентированной системы с указанием степени детализации их описания в данной книге

3.2. Стратегия описания конкретных примеров: итеративная разработка + итеративное изучение

Структура материала в этой книге основывается на итеративной стратегии разработки. Объектно-ориентированный анализ и проектирование применяются к разработке конкретных систем в процессе реализации нескольких итераций. На первой итерации обеспечивается выполнение некоторых базовых функций, а на последующих расширяется функциональность систем (рис. 3.2).

В соответствии с итерационной стратегией разработки в книге будут последовательно представлены главы, посвященные объектно-ориентированному анализу и проектированию, системе обозначений языка UML и шаблонам. В главах, посвященных первой итерации, будут рассмотрены базовые вопросы анализа и проектирования, а также основные обозначения, принятые в UML. В главах, посвященных второй итерации, будут описаны новые идеи, новые обозначения UML, шаблоны и т.д. Это же касается и третьей итерации.

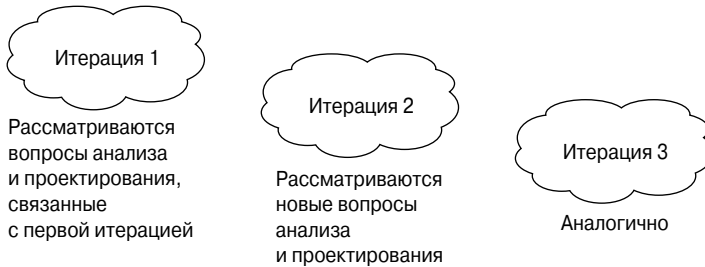


Рис. 3.2. Последовательность изучения материала соответствует итеративному процессу разработки приложения

3.3. Пример 1: POS-система NextGen

Первым примером, рассматриваемым в этой книге, является система автоматизации торговой точки – POS-система NextGen. Рассматриваемая задача очень интересна своими требованиями и проблемами, возникающими в процессе проектирования. Кроме того, она очень реалистична. Многие организации действительно разрабатывают POS-системы (point-of-sale system) на основе объектной технологии.

POS-система – это компьютеризированное приложение, предназначенное для организации товарооборота и обработки платежей в обычных магазинах. Система автоматизации торговли включает аппаратные компоненты (компьютер и устройство считывания штрихкода), а также программное обеспечение, выполняющее основные задачи системы. Это приложение связано с различными служебными программами, например с программой вычисления налогов, разработанной сторонними производителями, или с системой складского учета товаров. Подобные системы должны быть устойчивыми к сбоям, т.е. работоспособными при временном выходе из строя удаленных служб (например, системы складского учета товаров). В критических ситуациях они должны обслуживать продажу товаров и обеспечивать обработку хотя бы платежей наличными (чтобы хозяин магазина не обанкротился).



POS-система должна поддерживать различные типы клиентских терминалов и интерфейсов, в том числе клиентский терминал с Web-браузером (“тонкого” клиента), обычный персональный компьютер с графическим интерфейсом пользователя типа Java Swing, сенсорный ввод информации, беспроводный интерфейс и т.п.

Более того, коммерческие POS-системы должны уметь работать с различными категориями потребителей, для которых определены отдельные бизнес-правила. Для каждого потребителя может быть предусмотрена своя логика выполнения отдельных операций в рамках сценария использования системы, например, специфические действия при добавлении нового товара или создании новой продажи. Следовательно, необходимо предусмотреть механизм обеспечения этой гибкости и настройки системы.

На основе итеративной стратегии разработки мы выполним все необходимые этапы создания системы: формулировку требований, объектно-ориентированный анализ, проектирование и реализацию.

3.4. Пример 2: игра “Монополия”

Чтобы показать применимость основных приемов ООА/П к различным задачам, в качестве второго примера рассмотрим программную версию игры “Монополия”. Хотя предметная область и требования к этой системе не имеют ничего общего с коммерческими системами типа POS-системы NextGen, при разработке программы применяются те же методы моделирования предметной области, объектно-ориентированного проектирования, шаблоны и обозначения UML. Как и система автоматизации розничной торговли, программная версия игры “Монополия” реально существует и продается. Она также поддерживает разнообразное клиентское программное обеспечение и Web-интерфейс.

Мы не станем здесь описывать правила игры “Монополия”. Наверное, каждый человек в любой стране, будучи ребенком или подростком, играл в эту игру. Если вас миновала эта участь, правила игры можно найти на многих Web-узлах.

Программная версия игры работает в режиме моделирования. Один человек начинает игру и указывает количество моделируемых игроков, а затем наблюдает за ходом игры, отслеживая действия своих соперников и выполняя собственные.

