



Содержание

Предисловие	25
-------------------	----

ЧАСТЬ I. Нотация UML, концепции проектирования, технологии, жизненные циклы и методы

35

Глава 1. Введение	36
-------------------------	----

1.1. Объектно-ориентированные методы и UML	37
--	----

1.2. Метод и нотация	38
----------------------------	----

1.3. Параллельные приложения	38
------------------------------------	----

1.3.1. Последовательные и параллельные программы	39
--	----

1.3.2. Последовательные и параллельные приложения	39
---	----

1.3.3. Параллельные задачи	40
----------------------------------	----

1.4. Системы и приложения реального времени	40
---	----

1.5. Распределенные системы и приложения	42
--	----

1.6. Резюме	43
-------------------	----

Глава 2. Обзор нотации UML	44
----------------------------------	----

2.1. Диаграммы UML	44
--------------------------	----

2.2. Диаграммы прецедентов	45
----------------------------------	----

2.3. Нотация UML для классов и объектов	45
---	----

2.4. Диаграммы классов	46
------------------------------	----

2.5. Диаграммы взаимодействия	47
-------------------------------------	----

2.5.1. Диаграммы кооперации	47
-----------------------------------	----

2.5.2. Диаграммы последовательности	48
---	----

2.6. Диаграммы состояний	48
--------------------------------	----

2.7. Пакеты	50
-------------------	----

2.8. Диаграммы параллельной кооперации	51
--	----

2.8.1. Обмен сообщениями на диаграммах параллельной кооперации	51
--	----

2.9. Диаграммы развертывания	51
------------------------------------	----

2.10. Механизмы расширения UML	53
--------------------------------------	----

2.11. UML как стандарт	54
2.12. Резюме	55

Глава 3. Концепции проектирования

ПО и архитектуры	56
3.1. Объектно-ориентированные концепции	56
3.1.1. Основные концепции	56
3.1.2. Объекты и классы	57
3.2. Соккрытие информации	58
3.2.1. Соккрытие информации в объектно-ориентированном проектировании	59
3.2.2. Соккрытие информации в применении к внутренним структурам данных	59
3.2.3. Соккрытие информации при проектировании интерфейса с устройствами ввода/вывода	62
3.2.4. Проектирование объектов, скрывающих информацию	63
3.3. Наследование	64
3.4. Активные и пассивные объекты	65
3.5. Параллельная обработка	66
3.5.1. Преимущества параллельного выполнения задач	66
3.5.2. Тяжеловесные и облегченные процессы	67
3.6. Кооперация между параллельными задачами	68
3.6.1. Проблема взаимного исключения	68
3.6.2. Пример взаимного исключения	69
3.6.3. Проблема синхронизации задач	70
3.6.4. Пример синхронизации задач	70
3.6.5. Проблема производителя/потребителя	72
3.6.6. Слабо связанный обмен сообщениями	73
3.6.7. Сильно связанный обмен сообщениями с ответом	74
3.6.8. Сильно связанный обмен сообщениями без ответа	74
3.6.9. Пример обмена сообщениями между производителем и потребителем	75
3.7. Соккрытие информации в применении к синхронизации доступа	76
3.7.1. Классы и объекты, скрывающие информацию	76
3.8. Мониторы	77
3.8.1. Пример монитора	77
3.8.2. Условная синхронизация	78
3.9. Шаблоны проектирования	79

3.10. Программные архитектуры и компонентные системы	80
3.10.1. Компоненты и разъемы	81
3.10.2. Компонентные системы	81
3.11. Резюме	81
Глава 4. Технологии	
параллельных и распределенных систем	83
4.1. Среды для параллельной обработки	83
4.1.1. Мультипрограммная среда	83
4.1.2. Симметричная мультипроцессорная среда	83
4.1.3. Распределенная среда	84
4.2. Поддержка исполнения	
в мультипрограммной и мультипроцессорной средах	85
4.2.1. Сервисы операционной системы	85
4.2.2. Стандарт POSIX	86
4.2.3. Операционные системы реального времени	87
4.3. Планирование задач	88
4.3.1. Алгоритмы планирования задач	88
4.3.2. Состояния задач	89
4.3.3. Контекстное переключение задач	90
4.4. Вопросы ввода/вывода в операционной системе	90
4.4.1. Контроллеры устройств	90
4.4.2. Обработка прерываний	91
4.4.3. Ввод/вывод с опросом	92
4.5. Технологии клиент-серверных и распределенных систем ...	93
4.5.1. Конфигурации клиент-серверных и распределенных систем	93
4.5.2. Коммуникационные сетевые протоколы	95
4.6. Технология World Wide Web	97
4.6.1. Язык Java и World Wide Web	98
4.7. Сервисы распределенных операционных систем	98
4.7.1. Служба имен	99
4.7.2. Связывание клиентов и серверов	99
4.7.3. Сервисы распределенного обмена сообщениями	100
4.7.4. Сервисы сокетов	101
4.7.5. Обмен сообщениями через порты	101
4.7.6. Восстановление после ошибок	101
4.8. ПО промежуточного слоя	102
4.8.1. Платформы для распределенных вычислений	102
4.8.2. Вызовы удаленных процедур	102
4.8.3. Вызов удаленных методов в языке Java	104

4.9. Стандарт CORBA	104
4.9.1. Брокер объектных запросов	104
4.9.2. Язык определения интерфейса в CORBA	105
4.9.3. Статическое и динамическое связывание	106
4.9.4. Сервисы CORBA	107
4.9.5. Интеграция унаследованных приложений в каркас распределенных объектов	107
4.10. Другие компонентные технологии	108
4.10.1. Технология COM	108
4.10.2. Технология JavaBeans	108
4.10.3. Технология Jini	108
4.11. Системы обработки транзакций	109
4.11.1. Характеристики транзакций	109
4.11.2. Мониторы обработки транзакций	110
4.12. Резюме	111
Глава 5. Жизненные циклы и методы разработки программного обеспечения	112
5.1. Определение жизненного цикла ПО	112
5.1.1. Модель водопада	112
5.1.2. Недостатки модели водопада	113
5.1.3. Временные прототипы	114
5.1.4. Создание эволюционирующих прототипов в ходе инкрементной разработки	115
5.1.5. Комбинирование временных прототипов и инкрементной разработки	116
5.1.6. Спиральная модель	117
5.1.7. Унифицированный процесс разработки ПО	118
5.2. Верификация и утверждение проекта	118
5.2.1. Контроль качества ПО	119
5.2.2. Анализ производительности	119
5.3. Тестирование программного обеспечения	119
5.3.1. Автономное тестирование	120
5.3.2. Тестирование сопряжений	120
5.3.3. Комплексное тестирование	120
5.3.4. Приемно-сдаточные испытания	121
5.4. Эволюция методов проектирования ПО	121
5.5. Эволюция методов объектно-ориентированного анализа и проектирования	123

5.6. Обзор современных методов проектирования параллельных систем и систем реального времени	125
5.7. Резюме	126
 ЧАСТЬ II. COMET – метод архитектурного проектирования и моделирования параллельных объектов с применением UML	
127	
Глава 6. Введение в метод COMET	128
6.1. Жизненный цикл разработки объектно-ориентированного ПО в методе COMET	128
6.1.1. Моделирование требований	128
6.1.2. Аналитическое моделирование	128
6.1.3. Проектное моделирование	129
6.1.4. Инкрементное конструирование ПО	130
6.1.5. Инкрементная интеграция ПО	130
6.1.6. Комплексное тестирование	130
6.2. Сравнение жизненного цикла COMET с другими процессами разработки ПО	131
6.2.1. Сравнение жизненного цикла COMET с USDP	131
6.2.2. Сравнение жизненного цикла COMET со спиральной моделью	131
6.3. Модель требований, аналитическая и проектная модели ...	131
6.3.1. Виды деятельности при моделировании требований	132
6.3.2. Виды деятельности при аналитическом моделировании	132
6.3.3. Виды деятельности при проектном моделировании	133
6.4. Основы COMET	134
6.4.1. Разработка модели требований	134
6.4.2. Разработка аналитической модели	134
6.4.3. Разработка проектной модели	135
6.5. Резюме	137
Глава 7. Моделирование прецедентов	138
7.1. Прецеденты	138
7.2. Актеры	139
7.3. Актеры, роли и пользователи	141
7.4. Выявление прецедентов	141

7.5. Документирование прецедентов в модели прецедентов	142
7.6. Примеры прецедентов	143
7.6.1. Прецедент «Снять Деньги»	143
7.6.2. Прецедент «Получить Справку»	145
7.6.3. Прецедент «Перевести Деньги»	146
7.7. Отношения прецедентов	147
7.7.1. Отношение расширения	147
7.7.2. Отношение включения	149
7.7.3. Некоторые рекомендации	150
7.8. Пакеты прецедентов	150
7.9. Резюме	151
Глава 8. Статическое моделирование	152
8.1. Ассоциации между классами	152
8.1.1. Изображение ассоциаций на диаграммах классов	153
8.1.2. Кратность ассоциаций	153
8.1.3. Другие ассоциации	155
8.1.4. Атрибуты связи	155
8.1.5. Классы-ассоциации	156
8.2. Иерархии композиции и агрегирования	157
8.3. Иерархия обобщения/специализации	159
8.4. Ограничения	160
8.5. Статическое моделирование и язык UML	160
8.5.1. Статическое моделирование предметной области	161
8.6. Статическое моделирование контекста системы	162
8.6.1. Внешние классы	163
8.6.2. Пример разработки диаграммы классов контекста системы с внешними классами	164
8.6.3. Актеры и внешние классы	165
8.6.4. Пример разработки диаграммы классов контекста системы на основе рассмотрения актеров	165
8.7. Статическое моделирование сущностных классов	166
8.8. Резюме	167
Глава 9. Разбиение на классы и объекты	168
9.1. Критерии разбиения на объекты	168
9.2. Категории классов приложения	169
9.3. Структурирование категорий объектов	170
9.4. Внешние и интерфейсные классы	171
9.4.1. Категории внешних классов	171
9.4.2. Идентификация интерфейсных классов	172

9.5. Интерфейсные объекты	173
9.5.1. Объекты интерфейса устройства	173
9.5.2. Объекты интерфейса пользователя	175
9.5.3. Объекты интерфейса системы	176
9.5.4. Изображение внешних и интерфейсных классов	177
9.6. Сущностные объекты	178
9.7. Управляющие объекты	180
9.7.1. Объекты-координаторы	180
9.7.2. Управляющие объекты, зависящие от состояния	181
9.7.3. Объекты-таймеры	182
9.8. Объекты прикладной логики	182
9.8.1. Объекты бизнес-логики	182
9.8.2. Объекты-алгоритмы	183
9.9. Подсистемы	183
9.9.1. Пакеты для изображения подсистем	184
9.9.2. Вопросы, связанные с разбиением на подсистемы	185
9.10. Резюме	186
Глава 10. Конечные автоматы и диаграммы состояний	187
10.1. Конечные автоматы	187
10.2. События и состояния	188
10.2.1. События	188
10.2.2. Состояния	188
10.3. Конечные автоматы и объекты	188
10.4. Примеры диаграмм состояний	189
10.4.1. Пример диаграммы состояний счета	189
10.4.2. Пример диаграммы состояний банкомата	189
10.4.3. Пример диаграммы состояний системы круиз-контроля	191
10.5. События и условия	191
10.6. Действия	193
10.6.1. Деятельности	194
10.6.2. Действия при входе и выходе	197
10.7. Моделирование различных аспектов системы	200
10.8. Иерархические диаграммы состояний	201
10.8.1. Иерархическая декомпозиция состояний	201
10.8.2. Агрегирование переходов состояний	203
10.9. Параллельные диаграммы состояний	203
10.10. Рекомендации по разработке диаграмм состояний	205

10.11. Построение диаграмм состояний на основе прецедентов	206
10.12. Пример разработки диаграммы состояний на основе прецедента	207
10.12.1. Прецедент «Управление Скоростью»	207
10.12.2. Разработка диаграммы состояний	208
10.12.3. Рассмотрение альтернативных внешних событий	210
10.12.4. Разработка иерархической диаграммы состояний	211
10.12.5. Разработка ортогональной диаграммы состояний	213
10.13. Резюме	214
Глава 11. Динамическое моделирование	215
11.1. Моделирование взаимодействий объектов	215
11.1.1. Диаграммы кооперации	216
11.1.2. Диаграммы последовательности	217
11.1.3. Сравнение диаграмм последовательности и кооперации	218
11.1.4. Прецеденты и сценарии	218
11.1.5. Обобщенные и конкретные формы диаграмм взаимодействия	219
11.2. Сообщения-метки на диаграммах взаимодействия	219
11.2.1. Порядковая нумерация сообщений	220
11.2.2. Описание последовательности сообщений	222
11.3. Динамический анализ	223
11.4. Динамический анализ, не зависящий от состояния	223
11.5. Пример динамического анализа, не зависящего от состояния	225
11.6. Динамический анализ, зависящий от состояния	225
11.6.1. Определение объектов и взаимодействий	226
11.6.2. Исполнение диаграммы состояний	227
11.6.3. Рассмотрение альтернативных последовательностей	229
11.7. Пример динамического анализа, зависящего от состояния: банковская система	229
11.7.1. Главная последовательность	229
11.7.2. Альтернативные последовательности	232
11.8. Пример динамического анализа, зависящего от состояния: система круиз-контроля	237
11.9. Резюме	246

Глава 12. Проектирование архитектуры системы	250
12.1. Архитектурные стили	250
12.1.1. Архитектура клиент-сервер	250
12.1.2. Архитектура с несколькими уровнями абстракции	251
12.1.3. Архитектура взаимодействующих задач	252
12.1.4. Смешение архитектурных стилей	252
12.2. Декомпозиция системы	253
12.3. Рекомендации по выявлению подсистем	255
12.4. Консолидированные диаграммы кооперации	255
12.5. Архитектура подсистем	256
12.6. Разделение обязанностей при проектировании подсистем	258
12.7. Критерии разбиения на подсистемы	260
12.8. Примеры разбиения на подсистемы	263
12.9. Статическое моделирование на уровне проектирования	264
12.10. Резюме	266
Глава 13. Проектирование архитектуры распределенных приложений	268
13.1. Конфигурируемые архитектуры и программные компоненты	269
13.2. Шаги проектирования распределенного приложения	269
13.3. Декомпозиция системы	270
13.3.1. Проектирование распределенных подсистем	270
13.3.2. Агрегированные и составные подсистемы в COMET	272
13.3.3. Проектирование конфигурируемых распределенных подсистем	274
13.3.4. Критерии конфигурируемости распределенных компонентов	274
13.4. Проектирование интерфейсов подсистем	276
13.4.1. Слабо связанный (асинхронный) обмен сообщениями	276
13.4.2. Сильно связанный (синхронный) обмен сообщениями	277
13.4.3. Обмен сообщениями между несколькими клиентами и сервером	277
13.4.4. Коммуникации типа подписка/извещение и групповой обмен сообщениями	278

13.4.5. Коммуникации с участием брокера	279
13.4.6. Коммуникации с обговариванием условий	281
13.5. Управление транзакциями	283
13.5.1. Протокол двухфазной фиксации в системах обработки транзакций	283
13.5.2. Вопросы проектирования транзакций	284
13.6. Проектирование серверных подсистем	285
13.6.1. Последовательная серверная подсистема	286
13.6.2. Параллельная серверная подсистема	287
13.7. Распределение данных	290
13.7.1. Распределенный сервер	290
13.7.2. Репликация данных	290
13.8. Конфигурирование системы	290
13.8.1. Вопросы конфигурирования системы	290
13.8.2. Пример конфигурирования целевой системы	291
13.9. Резюме	292
Глава 14. Разбиение на задачи	293
14.1. Вопросы разбиения на параллельные задачи	294
14.2. Категории критериев разбиения на задачи	294
14.3. Критерии выделения задач ввода/вывода	295
14.3.1. Характеристики устройств ввода/вывода	295
14.3.2. Асинхронные задачи интерфейса с устройствами ввода/вывода	296
14.3.3. Периодические задачи интерфейса с устройством ввода/вывода	297
14.3.4. Пассивные задачи интерфейса с устройствами ввода/вывода	300
14.3.5. Задачи-мониторы ресурсов	302
14.4. Критерии выделения внутренних задач	302
14.4.1. Периодические задачи	302
14.4.2. Асинхронные задачи	304
14.4.3. Управляющие задачи	306
14.4.4. Задачи интерфейса пользователя	307
14.4.5. Множественные однотипные задачи	308
14.5. Критерии назначения приоритетов задачам	309
14.5.1. Критические по времени задачи	309
14.5.2. Некритические по времени расчетные задачи	310
14.6. Критерии группировки задач	310
14.6.1. Темпоральная группировка	311
14.6.2. Последовательная группировка	314

14.6.3. Группировка по управлению	316
14.6.4. Группировка по взаимному исключению	318
14.7. Пересмотр проекта путем инверсии задач	319
14.7.1. Инверсия нескольких экземпляров задачи	319
14.7.2. Инверсия последовательных задач	321
14.7.3. Темпоральная инверсия задач	321
14.8. Разработка архитектуры задач	324
14.8.1. Начальная диаграмма параллельной кооперации	327
14.9. Коммуникации между задачами и синхронизация	327
14.9.1. Слабо связанный (асинхронный) обмен сообщениями	327
14.9.2. Сильно связанный (синхронный) обмен сообщениями с ответом	329
14.9.3. Сильно связанный (синхронный) обмен сообщениями без ответа	330
14.9.4. Синхронизация по событию	331
14.9.5. Взаимодействие задач с помощью скрывающего информацию объекта	333
14.9.6. Пересмотренная диаграмма параллельной кооперации	334
14.10. Спецификация поведения задачи	334
14.10.1. Пример спецификации поведения для задачи «Банковский Сервер»	336
14.10.2. Пример спецификации поведения для задачи «Интерфейс Устройства Считывания Карточек»	337
14.11. Резюме	338
Глава 15. Проектирование классов	339
15.1. Проектирование классов, скрывающих информацию	339
15.2. Проектирование операций классов	340
15.2.1. Проектирование операций классов на основе модели взаимодействия	341
15.2.2. Проектирование операций классов на основе конечного автомата	342
15.2.3. Проектирование операций классов на основе статической модели	343
15.3. Классы абстрагирования данных	343
15.3.1. Пример класса абстрагирования данных	343
15.4. Классы интерфейса устройства	344
15.4.1. Проектирование операций классов интерфейса устройств	346
15.4.2. Классы интерфейса устройства ввода	346
15.4.3. Классы интерфейса устройства вывода	348
15.5. Классы, зависящие от состояния	350

15.6. Классы, скрывающие алгоритмы	352
15.7. Классы интерфейса пользователя	352
15.8. Классы бизнес-логики	355
15.9. Классы-обертки базы данных	356
15.10. Внутренние программные классы	358
15.11. Применение наследования при проектировании	358
15.11.1. Иерархии классов	358
15.11.2. Абстрактные классы	358
15.11.3. Полиморфизм и динамическое связывание	359
15.12. Примеры наследования	359
15.12.1. Примеры суперклассов и подклассов	359
15.12.2. Пример полиморфизма и динамического связывания	362
15.12.3. Пример наследования абстрактному классу	363
15.13. Спецификация интерфейса класса	364
15.13.1. Пример спецификации интерфейса класса	364
15.14. Резюме	366
Глава 16. Детальное проектирование ПО	367
16.1. Проектирование составных задач	367
16.1.1. Отношения между задачами и классами	367
16.1.2. Разделение обязанностей между задачами и классами	368
16.1.3. Темпоральная группировка и объекты интерфейса устройств	369
16.1.4. Группировка по управлению и объекты, скрывающие информацию	372
16.2. Синхронизация доступа к классам	374
16.2.1. Пример синхронизации доступа к классу	374
16.2.2. Операции класса абстрагирования данных	374
16.2.3. Синхронизация методом взаимного исключения	375
16.2.4. Синхронизация нескольких читателей и писателей	376
16.2.5. Синхронизация нескольких читателей и писателей с помощью монитора	377
16.2.6. Синхронизация нескольких читателей и писателей без ущемления писателей	380
16.3. Проектирование разъемов для межзадачных коммуникаций	381
16.3.1. Проектирование разъема, реализующего очередь сообщений	382
16.3.2. Проектирование разъема, реализующего буфер сообщений	383
16.3.3. Проектирование разъема, реализующего буфер сообщений с ответом	384
16.3.4. Проектирование кооперативных задач с использованием разъемов	385

16.4. Логика упорядочения событий	386
16.4.1. Пример логики упорядочения событий для задач отправителя и получателя	386
16.5. Резюме	387
Глава 17. Анализ производительности проекта параллельной системы реального времени	388
17.1. Теория планирования в реальном времени	388
17.1.1. Планирование периодических задач	389
17.1.2. Теорема о верхней границе коэффициента использования ЦП	389
17.1.3. Теорема о времени завершения	391
17.1.4. Строгая формулировки теоремы о времени завершения	392
17.1.5. Планирование периодических и аperiodических задач	394
17.1.6. Планирование с синхронизацией задач	395
17.2. Развитие теории планирования в реальном времени	395
17.2.1. Инверсия приоритетов	396
17.2.2. Обобщенная теорема о верхней границе коэффициента использования ЦП	397
17.2.3. Обобщенная теорема о времени завершения	398
17.2.4. Планирование в реальном времени и проектирование	398
17.2.5. Пример применения обобщенной теории планирования в реальном времени	399
17.3. Анализ производительности с помощью анализа последовательности событий	400
17.4. Анализ производительности с помощью теории планирования в реальном времени и анализа последовательности событий	401
17.5. Пример анализа производительности с помощью анализа последовательности событий	402
17.6. Пример анализа производительности с применением теории планирования в реальном времени	406
17.7. Анализ производительности по теории планирования в реальном времени и анализа последовательности событий	408
17.7.1. Эквивалентная аperiodическая задача	408
17.7.2. Назначение других приоритетов	410
17.7.3. Детальный анализ аperiodических задач	411
17.8. Пересмотр проекта	414
17.9. Оценка и измерение параметров производительности	415
17.10. Резюме	416

ЧАСТЬ III. Примеры проектирования параллельных приложений, распределенных приложений и приложений реального времени	417
Глава 18. Пример системы управления лифтами	418
18.1. Описание задачи	418
18.2. Модель прецедентов	419
18.2.1. Прецедент «Выбор Этажа Назначения»	419
18.2.2. Прецедент «Вызов Лифта»	420
18.2.3. Абстрактные прецеденты	420
18.2.4. Абстрактный прецедент «Остановка Лифта на Этаже»	421
18.2.5. Абстрактный прецедент «Планирование Лифта»	421
18.2.6. Конкретный прецедент «Выбор Этажа Назначения»	421
18.2.7. Конкретный прецедент «Вызов Лифта»	422
18.3. Статическая модель предметной области	422
18.4. Разбиение на объекты	424
18.5. Динамическая модель	424
18.5.1. Диаграмма кооперации для прецедента «Выбор Этажа Назначения»	425
18.5.2. Диаграмма кооперации для прецедента «Вызов Лифта»	425
18.5.3. Диаграмма кооперации для прецедента «Остановка Лифта на Этаже»	426
18.5.4. Абстрактный прецедент «Отправить Лифт»	430
18.6. Модель состояний	432
18.7. Консолидация диаграмм кооперации	434
18.8. Разбиение на подсистемы	437
18.9. Разбиение системы на задачи	440
18.9.1. Выделение задач в подсистеме лифта	442
18.9.2. Выделение задач в подсистеме этажа	444
18.9.3. Выделение задач в подсистеме планировщика	444
18.9.4. Определение интерфейсов задач	444
18.9.5. Проектирование класса абстрагирования данных	446
18.9.6. Обсуждение альтернативных архитектур	449
18.10. Проект распределенной системы управления лифтами	449
18.10.1. Структура подсистемы лифта	451
18.10.2. Структура подсистемы этажа	451

18.10.3. Структура подсистемы планировщика	455
18.10.4. Интерфейсы подсистем	455
18.11. Проектирование скрывающих информацию классов	456
18.11.1. Проектирование классов интерфейса устройств	456
18.11.2. Проектирование класса, зависящего от состояния	459
18.12. Разработка детального проекта программы	459
18.12.1. Проектирование объектов-разъемов для лифта	459
18.12.2. Проектирование составных задач	461
18.13. Конфигурирование целевой системы	463
18.14. Анализ производительности	
нераспределенной системы управления лифтами	464
18.14.1. Сценарий для анализа производительности	464
18.14.2. Последовательности событий	464
18.14.3. Назначение приоритетов	466
18.14.4. Планирование в реальном времени	
для нераспределенной архитектуры	468
18.14.5. Последовательность событий «Остановка Лифта на Этаже»	469
18.14.6. Последовательность событий «Выбор Этажа Назначения»	470
18.14.7. Последовательность событий «Вызов Лифта»	471
18.15. Анализ производительности	
распределенной системы управления лифтами	472
18.15.1. Сценарий для анализа производительности	472
18.15.2. Планирование в реальном времени	
для распределенной архитектуры	473
18.15.3. Последовательность событий «Остановка Лифта на Этаже»	477
18.15.4. Последовательность событий «Выбор Этажа Назначения»	479
18.15.5. Последовательность событий «Вызов Лифта»	480
Глава 19. Пример банковской системы	483
19.1. Описание задачи	483
19.2. Модель прецедентов	484
19.2.1. Абстрактный прецедент «Проверить ПИН-код»	485
19.2.2. Конкретный прецедент «Снять Деньги»	485
19.2.3. Конкретный прецедент «Получить Справку»	486
19.2.4. Прецедент «Перевести Деньги»	487
19.3. Статическое моделирование	487
19.3.1. Статическое моделирование предметной области	487
19.3.2. Статическое моделирование контекста системы	488
19.3.3. Статическое моделирование сущностных классов	488

19.4. Разбиение на объекты	491
19.4.1. Выделение клиентской и серверной подсистем	491
19.4.2. Выделение клиентских объектов: интерфейсные объекты	493
19.4.3. Выделение клиентских объектов: объекты, участвующие в прецедентах	493
19.4.4. Выделение объектов в серверной подсистеме	495
19.5. Динамическое моделирование	496
19.5.1. Описание последовательности сообщений для прецедента «Проверить ПИН-код на Клиенте»	497
19.5.2. Описание последовательности сообщений для прецедента «Проверить ПИН-код на Сервере»	500
19.5.3. Описание последовательности сообщений для прецедента «Снять Деньги на Клиенте»	502
19.5.4. Описание последовательности сообщений для прецедента «Снять Деньги на Сервере»	506
19.6. Диаграмма состояний банкомата	508
19.7. Проектирование банковской системы	512
19.8. Создание консолидированной диаграммы кооперации	513
19.9. Разбиение на подсистемы	515
19.10. Проект подсистемы банкомата	517
19.10.1. Проектирование архитектуры задач для подсистемы банкомата	517
19.10.2. Определение интерфейсов задач в подсистеме банкомата	521
19.10.3. Проектирование скрывающих информацию классов в подсистеме банкомата	523
19.10.4. Разработка детального проекта подсистемы банкомата	524
19.11. Проектирование подсистемы банковского сервера	526
19.11.1. Проектирование архитектуры параллельных задач в подсистеме банковского сервера	527
19.11.2. Проектирование скрывающих информацию классов в банковском сервере	528
19.11.3. Проектирование интерфейсов банковского сервера	530
19.12. Конфигурирование банковской системы	534
19.13. Альтернативные варианты	534
19.14. Спецификации поведения задач	534
19.14.1. Пример логики упорядочения событий для задачи «Интерфейс Устройства Считывания Карточек»	534
19.14.2. Пример логики упорядочения событий для задачи «Контроллер Банкомата»	536

19.14.3. Пример логики упорядочения событий для задачи «Банковский Сервер»	538
Глава 20. Пример системы круиз-контроля и мониторинга	540
20.1. Описание задачи	540
20.1.1. Круиз-контроль как задача управления процессами	541
20.2. Модель прецедентов	542
20.2.1. Прецеденты круиз-контроля	543
20.2.2. Прецеденты мониторинга	544
20.3. Описание прецедентов	544
20.3.1. Прецеденты, инициируемые водителем	544
20.3.2. Прецеденты, инициируемые таймером	546
20.3.3. Прецеденты, инициируемые механиком	548
20.4. Статическое моделирование предметной области	548
20.5. Динамическое моделирование	549
20.5.1. Начальное разбиение на подсистемы	550
20.5.2. Динамическое моделирование не зависящих от состояния аспектов подсистемы круиз-контроля	551
20.5.3. Динамическое моделирование зависящих от состояния аспектов подсистемы круиз-контроля	553
20.5.4. Зависящее от состояния динамическое моделирование: управление калибровкой	554
20.5.5. Динамическое моделирование подсистемы мониторинга: прецеденты, связанные со средней скоростью	558
20.5.6. Динамическое моделирование подсистемы мониторинга: прецеденты, связанные с расходом топлива	560
20.5.7. Динамическое моделирование прецедентов технического обслуживания	562
20.6. Разбиение на подсистемы	564
20.6.1. Разбиение подсистемы круиз-контроля на более мелкие подсистемы	564
20.6.2. Разбиение подсистемы мониторинга на более мелкие подсистемы	565
20.6.3. Зависимости между подсистемами	566
20.7. Уточненная статическая модель	571
20.8. Разбиение системы на задачи	572
20.8.1. Определение характеристик устройств ввода/вывода	575
20.8.2. Определение задач в подсистеме вала	575

20.8.3. Определение задач в подсистеме калибровки	575
20.8.4. Определение задач в подсистеме пути и скорости	577
20.8.5. Определение задач в подсистеме автоматического управления	580
20.8.6. Определение задач в подсистеме круиз-контроля	583
20.8.7. Проектирование классов абстрагирования данных	585
20.8.8. Проектирование интерфейсов между задачами и объектами абстрагирования данных	587
20.8.9. Определение задач в подсистеме мониторинга	587
20.8.10. Разработка архитектуры и интерфейсов задач подсистемы мониторинга	592
20.9. Проектирование скрывающих информацию классов	592
20.9.1. Определение классов интерфейса устройств	596
20.9.2. Определение зависящих от состояния классов	598
20.9.3. Определение классов-алгоритмов	600
20.10. Разработка детального проекта программы	602
20.10.1. Проектирование сгруппированных задач	602
20.10.2. Проектирование объектов-разъемов	606
20.11. Проектирование архитектуры распределенной автомобильной системы	607
Глава 21. Пример распределенной системы автоматизации производства	609
21.1. Описание задачи	609
21.2. Модель прецедентов	611
21.3. Концептуальная статическая модель предметной области	613
21.4. Разбиение на объекты	614
21.5. Динамическая модель	617
21.5.1. Диаграммы кооперации для клиент-серверных прецедентов дежурного оператора	617
21.5.2. Диаграммы кооперации для прецедентов подписки на извещения	618
21.5.3. Диаграммы кооперации для клиент-серверных прецедентов управления производством	620
21.5.4. Прецеденты распределенного управления	623
21.5.5. Диаграмма состояний контроллера линейной рабочей станции	624
21.5.6. Прецеденты изготовления детали	626
21.6. Разбиение на подсистемы	630

21.7. Архитектура распределенной программы	634
21.7.1. Применение критериев конфигурируемости компонентов	635
21.7.2. Обмен сообщениями между подсистемами	638
21.8. Конфигурирование системы	641
Глава 22. Пример системы электронной коммерции	644
22.1. Задача электронной коммерции	644
22.2. Модель прецедентов	644
22.3. Агенты, поддерживающие систему электронной коммерции	645
22.4. Поддержка системы электронной коммерции со стороны брокеров объектов	647
22.5. Статическое моделирование предметной области	648
22.6. Модель кооперации	648
22.6.1. Модель кооперации для просмотра каталога	649
22.6.2. Модель кооперации для размещения заявки	649
22.6.3. Модель кооперации для обработки заказа	651
22.6.4. Модель кооперации для подтверждения отгрузки	651
22.6.5. Модель кооперации для подтверждения доставки	652
22.6.6. Модель кооперации для отправки счета-фактуры	653
22.7. Архитектура распределенной программы	655
Глоссарий	663
Библиография	675
Предметный указатель	684



Предисловие

Предисловие Питера Фримена

Недавние успехи в области разработки аппаратного обеспечения и средств связи привели к лавинообразному росту числа параллельных и распределенных систем, а также систем реального времени. Это, в свою очередь, способствовало изменению требований, предъявляемых к программному обеспечению. Широкое распространение объектно-ориентированных методов, а теперь еще и применение языка UML, разумеется, влияют на сложившуюся практику, однако темпы, к сожалению, отстают от требований времени.

Одна из причин такого отставания – отсутствие добротных практических руководств по объектно-ориентированному анализу и проектированию параллельных приложений (в особенности таких, которые одновременно являются распределенными и/или должны работать в режиме реального времени). Книга, которую вы держите в руках, призвана восполнить этот пробел.

Я не знаю человека, который мог бы написать авторитетный учебник по данному предмету лучше, чем Хассан Гома. Более 20 лет Хассан изучал природу систем реального времени, параллельных и распределенных, работая проектировщиком, проводя исследования в области новых методов проектирования подобных систем и преподавая в университете. Результаты его деятельности перед вами.

Книга прекрасно организована, поскольку написал ее опытный преподаватель. В ней продемонстрирована такая глубина понимания технологии, которая появляется только в результате длительных исследований в данной области. А рисунки и практические рекомендации свидетельствуют о наличии у автора непосредственного опыта проектирования программных систем.

Надеюсь, что вы получите от чтения этой книги такое же удовольствие, как и я, и будете пользоваться ею еще много лет.

Предисловие Брана Селика

В условиях недостаточности данных допускается куда меньше ошибок, чем при полном отсутствии данных.

Чарльз Бэббидж

Недавно, произведя поиск на Web-сайте, специализирующемся в области книготорговли, я обнаружил 1188 названий в категории «проектирование программного обеспечения», и число книг на эту тему постоянно растет. Несмотря на такое

изобилие, в большинстве подобных изданий почти ничего не говорится собственно о теории и практике проектирования, а потому огромное количество программ в наши дни создается людьми, плохо знакомыми с предметом.

В традиционных инженерных дисциплинах всегда использовались последние достижения прикладной математики, гарантирующие, что проект будет отвечать поставленным требованиям при приемлемых затратах. Основываясь на оценках, полученных с помощью математической модели, можно достаточно уверенно приступать, например, к конструированию моста. Однако применительно к программному обеспечению проектирование оказывается преимущественно неформальным процессом, для которого зачастую нет моделей и методов прогнозирования результата. С этой точки зрения весьма поучительно было бы сравнить эволюцию программного и аппаратного обеспечения на протяжении нескольких последних десятилетий. Если аппаратные устройства со временем становились миниатюрнее, быстрее и дешевле, то программы, напротив, оказывались все более объемными, медленными, дорогими и менее надежными. Одна из основных причин такого положения состоит в том, что в основе проектирования современной аппаратуры лежит использование прогностических моделей.

Отсутствие фундаментальных инженерных принципов в практике разработки ПО можно отчасти объяснить изменчивой, в чем-то даже хаотичной природой программ, что сильно затрудняет математическое моделирование. Тем не менее имеется немало весьма полезных аналитических методик. Особенно активно такие методики развивались в области создания систем реального времени: здесь надежное предсказание временных характеристик программы часто оказывается критически важным, поскольку от этого зависит человеческая жизнь. И все же, вопреки доказанной на опыте эффективности, подобные методы распространены не очень широко. По сути дела, многие разработчики систем реального времени даже не подозревают об их существовании.

Написание программы – это в основном интеллектуальное упражнение, не ограниченное такими физическими факторами, как необходимость обрабатывать сырье и затрачивать много энергии. У большинства практикующих программистов, соблазненных кажущимся отсутствием сопротивления, создается впечатление, что важен только сам код, а потому они не разделяют процессы проектирования и кодирования. Как это ни странно, те же самые люди прекрасно понимают разницу между проектированием и сборкой авиалайнера.

Дополнительным препятствием к внедрению вышеупомянутых методик в практику создания ПО служит то, что некоторые из них ориентированы на традиционное процедурное программирование. Хотя никаких фундаментальных зависимостей от такой модели не имеется, проблема привязки к более современной объектно-ориентированной модели все же существует, и тем, кто ценит преимущества, заложенные в новой парадигме, это мешает пользоваться старыми методами проектирования.

Книга Хассана Гома – первая из известных мне, в которой эти вопросы рассматриваются систематически и всесторонне. Она представляет собой не собрание разнородных паттернов и методик, а ясное и подробное изложение способов

объединения традиционных методов инженерного проектирования с унифицированным языком моделирования (UML), который уже обрел статус стандарта. Кроме того, демонстрируется роль данных методов в процессе создания конкретных систем: параллельных, распределенных и реального времени. (Опытные разработчики знают, какие трудности стоят за каждым из этих терминов по отдельности, а создание систем, обладающих всеми тремя характеристиками, – одна из наиболее сложных инженерных задач.)

В соответствии с хорошо известным принципом, что обучаться лучше всего на практике, значительную часть книги занимают детально продуманные нетривиальные примеры. Вам будет очень полезно полностью разобраться хотя бы в одном из них, тогда вы по-настоящему оцените и удобство предлагаемого подхода, и то, как на самом деле должно выглядеть инженерное проектирование программного обеспечения.

Нотация UML и методы проектирования программ

В этой книге рассматривается объектно-ориентированный анализ и проектирование параллельных приложений, в частности распределенных систем и систем реального времени. Объектно-ориентированные концепции играют огромную роль в анализе и проектировании программного обеспечения, поскольку касаются фундаментальных вопросов адаптируемости и развития. На протяжении некоторого времени существовало множество разнообразных нотаций и методов объектно-ориентированного анализа и проектирования программных систем, но потом появился унифицированный язык моделирования (UML), который представляет стандартизованную нотацию для описания таких моделей. Однако UML лучше использовать в сочетании с методами объектно-ориентированного анализа и проектирования.

В большинстве изданий, посвященных объектно-ориентированному анализу и проектированию, рассматриваются только последовательные системы, а важные вопросы, которые возникают в связи с разработкой распределенных систем и систем реального времени, опускаются. Для создания такого рода приложений нужно объединить объектно-ориентированные концепции с методами параллельной обработки. Поскольку на сегодняшний день стандартной нотацией для описания подобных моделей является UML, то именно этим языком мы и будем пользоваться далее.

COMET – метод архитектурного проектирования и моделирования параллельных объектов

COMET (Concurrent Object Modeling and Architectural Design Method) – это метод разработки параллельных приложений, в частности распределенных систем и систем реального времени. В основе создания объектно-ориентированного ПО по методу COMET лежит концепция прецедентов, а его жизненный цикл характеризуется большим числом итераций. На этапе моделирования требований

(Requirements Modeling) система рассматривается как черный ящик. Формируется модель прецедентов, где определяются функциональные требования к системе в терминах актеров и прецедентов.

На этапе аналитического моделирования (Analysis Modeling) строятся статическая и динамическая модели системы. Статическая модель описывает структурные отношения между классами предметной области. Для выявления объектов, рассматриваемых в аналитической модели, применяется критерий разбиения на объекты. После этого разрабатывается динамическая модель и уточняются описанные в модели требований прецеденты с целью представить объекты, участвующие в каждом прецеденте, и взаимодействия между ними. В динамической модели с помощью диаграмм состояний определяются объекты, зависящие от состояния.

На этапе проектного моделирования (Design Modeling) продумывается архитектура системы. Аналитическая модель, в которой основное внимание уделялось предметной области, соотносится со средой, где будет эксплуатироваться программа, и с проектной моделью, где акцент ставится на область решения. Формулируются критерии разбиения системы на подсистемы. В случае распределенной системы наиболее важным является разделение ответственности между клиентами и серверами, в том числе с точки зрения централизации и распределения данных и управления. Кроме того, проектируются интерфейсы для обмена сообщениями, рассматриваются синхронные, асинхронные, групповые коммуникации и брокерские сервисы. Затем наступает черед проектирования отдельных подсистем. Проектирование параллельных приложений, в том числе и систем реального времени, сводится в основном к выделению параллельно выполняемых объектно-ориентированных задач. Создаются интерфейсы для обмена данными между задачами и синхронизации. При анализе производительности новой системы реального времени используется метод монотонного анализа частот, предложенный Институтом проектирования программных систем (Software Engineering Institute).

Особенности книги

Существует несколько книг, описывающих концепции и методы объектно-ориентированного анализа для приложений общего вида. Однако распределенные системы и системы реального времени имеют специфические особенности, которые в большинстве изданий рассматриваются лишь поверхностно. Здесь вы найдете исчерпывающие сведения о том, как фундаментальные объектно-ориентированные концепции применяются к анализу и проектированию распределенных программ (включая клиент-серверные приложения) и программ реального времени. Помимо таких объектно-ориентированных концепций, как сокрытие информации, классы и наследование, в этой книге рассказывается о конечных автоматах, параллельных задачах, технологии распределенных объектов и планировании в режиме реального времени. Подробно описывается метод СОМЕТ, представляющий собой специализацию основанного на UML подхода к объектно-ориентированному анализу и проектированию параллельных и распределенных систем, а также систем реального времени. Чтобы продемонстрировать применение СОМЕТ на

практике, в издание включено несколько примеров из разных областей: система реального времени, система клиент-сервер и распределенная система.

Отличительные черты данной книги состоят в следующем:

- здесь рассказывается о критериях разбиения, призванных помочь программисту выявлять подсистемы, объекты и параллельно выполняемые задачи на разных стадиях процесса анализа и проектирования;
- с помощью динамического моделирования взаимодействия объектов и конечных автоматов описываются способы совместного использования диаграмм кооперации и диаграмм состояний;
- делается акцент на параллельности – приводятся характеристики активных и пассивных объектов;
- рассматриваются проектирование распределенных приложений и методы взаимодействия распределенных компонентов;
- анализируется производительность системы реального времени с помощью теории планирования в реальном времени;
- приводятся развернутые примеры различных приложений, иллюстрирующие применение изложенных концепций и методов.

Структура книги

Книга состоит из трех частей. Первая часть содержит обзор различных концепций, технологий, жизненных циклов и методов проектирования систем реального времени, параллельных и распределенных систем. Глава 1 начинается с краткого описания различий между методом и нотацией, за которым следует обсуждение характеристик распределенных систем и систем реального времени. В главе 2 излагаются те аспекты нотации UML, которые задействованы в методе COMET. Далее рассматриваются важные концепции проектирования (глава 3) и необходимой технологической поддержки (глава 4) для параллельных и распределенных систем. Затем в главе 5 рассказывается о жизненных циклах ПО и методах проектирования.

Во второй части речь идет о методе COMET. В главе 6 анализируется жизненный цикл ПО, созданного с использованием COMET. В главе 7 говорится об этапе моделирования требований в COMET и, в частности, о моделировании прецедентов. В главах 8–11 описываются этапы аналитического моделирования в COMET. Главы 12–16 посвящены этапу проектного моделирования. Предмет главы 17 – расчет производительности созданной системы с помощью теории планирования в реальном времени и метода монотонного анализа частот.

В третьей части метод COMET демонстрируется на пяти подробных примерах проектирования распределенных приложений (две системы реального времени, система клиент-сервер и две распределенные системы). В главе 18 рассматривается система управления лифтами и предлагаются два решения: одно распределенное, другое – нет. В главе 19 исследуется банковская клиент-серверная система. Глава 20 посвящена автомобильной системе круиз-контроля, глава 21 – распределенной системе автоматизации производства, а глава 22 – распределенной системе электронной коммерции.

Как работать с книгой

Эту книгу можно читать по-разному. Допустимо читать подряд: главы 1–5 являются введением в используемые концепции и технологии, в главе 6 приводится обзор метода СОМЕТ, главы 7–17 посвящены углубленному изучению порядка проектирования приложений с помощью СОМЕТ, а главы 18–22 включают примеры и их анализ.

Первая часть – вводная. Если вы владеете предметом, можете пропустить ее и сразу перейти к описанию метода СОМЕТ во второй части. Тем, кто знаком с языком UML, не понадобится глава 2. Зная концепции проектирования ПО, вы не станете читать главу 3. Если вы обладаете сведениями о технологиях, применяемых в параллельных и распределенных системах, пропустите главу 4; если же вам известны жизненные циклы ПО и методы проектирования, пропустите главу 5. Читателям, которые заинтересованы главным образом в методе СОМЕТ, удобнее сразу перейти ко второй и третьей частям книги. Тем, чья работа связана с проектированием распределенных приложений, следует прочесть главы 4, 12 и 13 (дополнительная информация о параллельных системах приводится в главах 14–16), а также изучить примеры распределенных приложений в главах 18, 19, 21 и 22. Желая ознакомиться с проектированием систем реального времени и теорией планирования в реальном времени необходимо прочесть главы 4, 14–17 и разобрать примеры в главах 18 и 20.

Опытные проектировщики могут использовать это издание в качестве справочника, обращаясь к тем или иным главам на разных этапах анализа или проектирования проекта. Главы мало связаны друг с другом. Например, краткое описание прецедентов вы найдете в главе 7, при проектировании диаграмм состояний можете заглянуть в главу 10, а при разработке динамической модели – в главу 11, проектирование распределенных компонентов рассматривается в главе 13, проектирование параллельно выполняемых задач – в главе 14, а информацию о планировании в реальном времени вы найдете в главе 17. Разобраться в том, как используется метод СОМЕТ, несложно и путем изучения примеров, поскольку в каждом из них объясняются мотивы принятия решений на каждом этапе проектирования.

Благодарности

Я очень признателен рецензентам ранних редакций рукописи. Особая благодарность Джеффу Мейджи (Jeff Magee), Ларри МакАлистеру (Larry McAlister), Кевину Миллсу (Kevin Mills), Роберту Дж. Петтиту IV (Robert G. Pettit IV) и Марии Эрикссон (Maria Ericsson) за пронизательные замечания. Хочу также поблагодарить Антуана К. Дина (Antuan Q. Dinh), Гулама Ахмада Фаруха (Ghulam Ahmad Farukh), Йохана Галле (Johan Galle), Келли Хьюстона (Kelli Houston), Жишну Мукерджи (Jishnu Mukerji), Лесли Пробаско (Leslee Probasco), Санджива Сетия (Sanjeev Setia) и Думинду Виджесекера (Duminda Wijesekera) за ценные рецензии.

Спасибо Кевину Миллсу за вклад в использование стереотипов в COMET, Шигеру Оцуки (Schigeru Otsuki) – за помощь в работе над материалом по паттернам проектирования, Роджеру Александру (Roger Alexander) – за участие в работе над одним из примеров главы 15, Ларри МакАлистеру, помогавшему сделать рис. 21.1. Особая благодарность – Тиррелл Элбо (Tyrrell Albaugh) за самоотверженный труд по координации процесса выпуска книги, Кристин Эрикссон (Kristin Ericsson) – за организацию издательского процесса и Мелинде МакКейн (Melinda McCain) – за тщательную корректуру рукописи.

Я признателен студентам, принимавшим участие в семинаре по проектированию программных систем в Университете Джорджа Мейсона, за энтузиазм, преданность и ценные замечания, а также Институту проектирования программных систем (Software Engineering Institute – SEI) за предоставленный материал по планированию в реальном времени, частично положенный в основу главы 17. Необходимо поблагодарить и Консорциум по повышению производительности разработки ПО (Software Productivity Consortium) за спонсорскую помощь, оказанную при работе над первой редакцией глав 9–11 этой книги. Спасибо Арману Анвару (Arman Anwar), Хуа Линь (Hua Lin) и Майклу Шину (Michael Shin) за изготовление ранних версий рисунков.

И, разумеется, я хочу поблагодарить свою жену Джилл за понимание, ободрение и поддержку.

Типографские соглашения и альтернативная нотация

В этом разделе описываются использованные в книге типографские соглашения. Здесь же приводится альтернативная нотация UML для обозначения стереотипов и активных объектов.

Принятые обозначения

Для улучшения восприятия соглашения, применяемые для записи имен классов, объектов и т.д. на рисунках, иногда отличаются от записи тех же имен в тексте. На рисунках имена набраны шрифтом Times Roman, а в тексте – шрифтом Courier (этим же моноширинным шрифтом в книге набраны листинги – фрагменты программного кода). Основные термины в тексте выделены *курсивом*, нотации UML отмечены **полужирным шрифтом**. Используемые обозначения во многом определяются этапом проектирования. Заглавные буквы в названиях употребляются по-разному в зависимости от того, об аналитической (менее формальной) или проектной (более формальной) моделях идет речь.

Моделирование требований

Имена прецедентов набраны шрифтом Courier, первые буквы каждого слова в имени – заглавные, например Снять Деньги.

Аналитическая модель

Ниже приведены соглашения, применяемые в аналитических моделях.

Классы

Имена классов набраны шрифтом *Courier* с начальными заглавными буквами. На рисунках отдельные слова в именах классов не разделяются пробелами (ЧековыйСчет), а в тексте – разделяются для удобства чтения (Чековый Счет).

Имена атрибутов начинаются с маленькой буквы, например баланс. Если имя атрибута состоит из нескольких слов, то на рисунках они не разделяются пробелами, а в тексте – разделяются. Первое слово имени начинается с маленькой буквы, остальные – с большой, допустим номер Счета.

Тип атрибута пишется с большой буквы, скажем Boolean, Integer или Real.

Объекты

Имена объектов записываются по-разному. На рисунках, в отличие от текста, они всегда подчеркиваются. Кроме того, применяются следующие соглашения:

- отдельный именованный объект. В этом случае первая буква в первом слове имени – маленькая, а в последующих словах – большая. На рисунках имя объекта выглядит так: чековыйСчет или другойЧековыйСчет. В тексте те же объекты обозначаются иначе: чековый Счет или другой Чековый Счет;
- отдельный неименованный объект. Иногда объекты на рисунках показаны как неименованные экземпляры класса, например : ЧековыйСчет. В тексте такой объект будет обозначен как Чековый Счет. Для удобства восприятия двоеточие опускается, а между словами в составном имени вставляются пробелы.

Таким образом, в зависимости от способа изображения объекта на рисунке его имя в тексте иногда начинается с большой, а иногда с маленькой буквы.

Сообщения

В аналитической модели все сообщения являются простыми (см. рис. 2.11), поскольку решение об их типе еще не принято. Имена сообщений начинаются с большой буквы. Если имя состоит из нескольких слов, то они разделяются пробелами, например Имя Простого Сообщения.

Обозначения на диаграммах состояний

Имена состояний, событий, условий, действий и деятельности начинаются с большой буквы, слова в составном имени разделяются запятыми, например состояние Ожидание ПИН-кода, событие Наличные Выданы и действие Выдать Наличные.

Проектная модель

Ниже приведены соглашения, применяемые в проектных моделях.

Активные и пассивные классы

Соглашения об именовании активных и пассивных классов такие же, как для классов в аналитической модели.

Активные и пассивные объекты

Соглашения об именовании активных и пассивных объектов такие же, как для объектов в аналитической модели.

Сообщения

В проектной модели первое слово в имени сообщения начинается с маленькой буквы, остальные – с большой. На рисунках слова не разделяются пробелами (тревожноеСообщение), а в тексте пробелы между ними есть (тревожное Сообщение).

Имена параметров сообщений начинаются с маленькой буквы, например скорость. Если имя состоит из нескольких слов, то на рисунках оно записывается без разделительных пробелов, а в тексте – с ними. Первое слово в составном имени начинается с маленькой буквы, остальные – с большой, скажем полныйПробег (на рисунке) или полный Пробег (в тексте).

Альтернативная нотация для стереотипов

В UML нотация, принятая для стереотипов, позволяет адаптировать элементы модели к особенностям конкретной предметной области. С одной стороны, стереотипы можно обозначать с помощью двойных угловых кавычек внутри элемента модели (например, класса или объекта), как показано на рис. 1а. С другой стороны, в UML допускается изображение стереотипа в виде графического символа. Наиболее распространенные на сегодняшний день пиктограммы были введены Джекобсоном [Jacobson 1992] и применяются в унифицированном процессе разработки программ [Jacobson, Booch, and Rumbaugh 1999] на этапе аналитического моделирования. Стереотипы используются для представления классов-сущностей (entity), граничных классов (boundary, в методе СОМЕТ им соответствуют классы со стереотипом «интерфейс») и управляющих классов (control). На рис. 1б с помощью графических стереотипов Джекобсона изображены сущностный класс

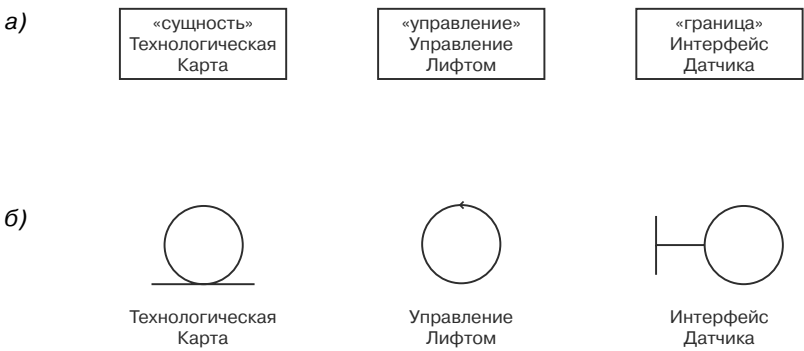


Рис. 1. Альтернативные нотации для стереотипов в UML:

а – стандартная нотация UML для стереотипов; б – нотация для стереотипов, применяемая в унифицированном процессе разработки программ

Технологическая Карта, управляющий класс Управление Лифтом и граничный класс Интерфейс Датчика.

Альтернативная нотация для активных объектов

В этой книге применяется стандартная нотация UML для обозначения активных (в методе COMET они называются задачами) и пассивных объектов. В UML прямоугольник активного объекта рисуется с жирной рамкой, а прямоугольник пассивного объекта – с тонкой (см. главу 2). Некоторые CASE-средства не поддерживают такой стандарт, поэтому активные и пассивные объекты оказываются визуально неразличимыми. В COMET слово «задача» обычно не употребляется в имени стереотипа активного объекта. Как показано на рис. 2а на примере объекта Генератор Отчетов, для этой цели достаточно прямоугольника с жирной рамкой. Если же такой визуальный эффект не поддерживается, то в имя стереотипа можно вставить слово «задача» (см. рис. 2б).



Рис. 2. Альтернативные нотации для активных объектов (задач) в UML: а – стандартная нотация UML для активных объектов; б – вариант, используемый, если стандартная нотация активных объектов не поддерживается

ЧАСТЬ I



**Нотация UML,
концепции
проектирования,
технологии,
жизненные циклы
и методы**

Глава 1.

Введение

Глава 2.

Обзор нотации UML

Глава 3.

**Концепции
проектирования ПО
и архитектуры**

Глава 4.

**Технологии параллельных
и распределенных систем**

Глава 5.

**Жизненные циклы и методы
разработки программного
обеспечения**



Глава 1. Введение

Значительное понижение цен на микропроцессоры и полупроводниковые микросхемы и столь же существенное увеличение производительности микропроцессоров, наблюдаемые на протяжении нескольких лет, сделали рентабельными распределенные системы и системы реального времени на базе микрокомпьютеров. Сегодня большинство коммерческих, промышленных, военных, медицинских и потребительских продуктов снабжаются микропроцессорами и целиком либо в значительной части управляются программами. Такие системы встречаются в микроволновых печах и видеомагнитофонах, в телефонах и телевизорах, в автомобилях и самолетах, в подводных лодках и космических кораблях, в автоматах по продаже газированных напитков и банкоматах, в системах диагностики пациентов и системах управления производством, в контроллерах роботов и системах управления лифтами, в системах управления городским и воздушным транспортом, в электронной почте и коммерции, в «интеллектуальных» транспортных и информационных магистралях... Список можно продолжать до бесконечности. Все это параллельные системы, а многие из них являются к тому же распределенными или системами реального времени.

Объектно-ориентированные концепции особенно важны для анализа и проектирования программного обеспечения, поскольку они касаются фундаментальных вопросов адаптируемости и развития. Сравнительно недавно появившийся унифицированный язык моделирования (UML) предлагает стандартизованную нотацию для описания объектно-ориентированных моделей [Booch, Rumbaugh, Jacobson 1998; Eriksson and Penker 1998; Fowler and Scott 1999; Jacobson, Booch and Rumbaugh 1999; Rumbaugh, Booch and Jacobson 1999]. Однако в большинстве книг, посвященных данным темам, рассматриваются только последовательные приложения и опускаются существенные вопросы, связанные с проектированием систем реального времени, параллельных и распределенных систем. Объединение концепций объектно-ориентированного проектирования с концепциями параллельного выполнения необходимо для успешного создания распределенных приложений, работающих в реальном масштабе времени. Поскольку UML содержит стандартную нотацию для описания объектно-ориентированных моделей, в книге будет использоваться именно этот язык. Особое внимание уделяется моделированию динамики системы, представляющему интерес для приложений реального времени и распределенных приложений.

В книге рассматривается объектно-ориентированный анализ и проектирование параллельных систем с использованием нотации UML. В частности, описываются две важнейшие категории параллельных приложений: распределенные

и реального времени. Издание предназначено для тех, кто хочет научиться проектировать и оценивать программное обеспечение параллельных и распределенных систем и систем реального времени.

В этой главе вводятся объектно-ориентированные концепции и обсуждаются общие характеристики трех вышеупомянутых категорий программных систем.

1.1. Объектно-ориентированные методы и UML

Объектно-ориентированные методы основаны на концепциях сокрытия информации, классов и наследования [Wegner 1990]. *Соккрытие информации* [Parnas 1972, Parnas 1979] позволяет получить замкнутые, а оттого в большей степени поддающиеся модификации и сопровождению системы. *Наследование* [Meyer 1987, Meyer 1997, Wegner 1990] – это систематический способ адаптации классов.

Язык UML, пришедший на смену многочисленным системам нотации и методикам проектирования [Booch 1994; Coad and Yourdon 1991; Jacobson 1992; Rumbaugh et al. 1991; Shlaer and Mellor 1988; Wirfs-Brock, Wickerson, and Wiener 1990 и др.], предложил нотацию для описания объектно-ориентированных моделей, которая стала промышленным стандартом. Однако для эффективного применения нотации UML необходимо сочетать ее с каким-либо методом объектно-ориентированного анализа и проектирования.

В описываемом методе COMET сочетаются прецеденты использования [Jacobson 1992], статическое моделирование [Rumbaugh et al. 1991, Booch 1994], диаграммы состояний [Harel 1988, Harel and Politi 1998, Coleman et al. 1994, Rumbaugh et al. 1991] и диаграммы последовательности событий, которые встречаются в нескольких методах [Booch 1994, Goma 1993, Jacobson 1992, Rumbaugh et al. 1991]. Применяемая нотация основана на UML [Booch, Rumbaugh, and Jacobson 1998; Eriksson and Penker 1998; Fowler and Scott 1999; Jacobson, Booch and Rumbaugh 1999; Rumbaugh, Booch and Jacobson 1999]. В ходе *моделирования прецедентов использования* определяются функциональные требования к системе в терминах актеров и прецедентов. *Статическая модель* предлагает статический взгляд на информационные аспекты системы. Класс определяется в терминах своих атрибутов и взаимоотношений с другими классами. Результатом *динамического моделирования* является динамический взгляд на систему. Уточняются сформулированные ранее прецеденты с целью показать взаимодействие объектов, участвующих в каждом из них. Разрабатываются диаграммы кооперации и последовательности, отражающие кооперацию объектов в каждом прецеденте. Зависящие от состояния аспекты системы описываются с помощью диаграмм состояний, причем для каждого объекта составляется своя диаграмма.

В аналитической модели основное внимание уделяется пониманию проблемы: выявлению объектов предметной области и передаваемой между ними информации. Объекты и классы предметной области группируются на основе критериев выделения объектов. Рассмотрение вопросов, активен объект или пассивен, синхронно или асинхронно посылаемое сообщение и какая вызывается операция у объекта-получателя, откладывается до стадии проектирования.

На этапе проектирования среди объектов выявляются активные (их называют задачами) и пассивные (их называют объектами). Для определения задач применяются критерии разбиения на задачи. Разрабатываются также интерфейсы задач и описываются операции каждого пассивного класса.

1.2. Метод и нотация

Метод проектирования и нотация проектирования – это разные вещи. Нотация проектирования ПО предназначена для описания самого проекта. Хотя она и предполагает наличие определенного подхода к проектированию, сам подход остается за ее рамками. Метод проектирования ПО представляет собой систематическое описание этапов создания проекта.

Нотация проектирования ПО описывает проект программы в графическом или текстовом виде. В частности, диаграммы классов – это графическая нотация, а псевдокод – текстовая.

Концепция проектирования ПО – это фундаментальная идея, применимая к проектированию всей системы, например сокрытие информации.

Стратегия проектирования ПО – общий план и методика выполнения проекта. Одной из стратегий является объектно-ориентированная декомпозиция.

Критерии структурирования ПО – это эвристические или формальные правила, помогающие проектировщику разбить систему на отдельные компоненты. Так, критерии разбиения на объекты – это правила декомпозиции системы на объекты.

Метод проектирования ПО описывает последовательность шагов, выполняемых при работе над проектом при условии, что требования к системе уже сформулированы. Он помогает выявить, какие решения предстоит принять, в каком порядке это следует делать и на основе каких критериев. Метод проектирования базируется на наборе соответствующих концепций, использует одну или несколько стратегий, а также ту или иную нотацию для документирования результатов. При выполнении определенных шагов метод может подсказать разработчику, какие критерии наиболее удобны для декомпозиции системы.

В методе СОМЕТ для описания проекта применяется нотация UML. Метод основан на следующих концепциях: сокрытие информации, наследование и параллельные задачи. Стратегия проектирования параллельно работающих объектов состоит в разбиении системы на активные и пассивные объекты и определении интерфейсов между ними. Метод СОМЕТ также содержит критерии разбиения, помогающие выделить объекты в системе на этапе анализа, а затем на этапе проектирования выявить отдельные подсистемы и параллельно выполняемые задачи.

1.3. Параллельные приложения

На заре развития вычислительной техники компьютерные приложения являлись в основном пакетными заданиями. Все программы были последовательными и работали в автономном режиме. Теперь же, когда имеется огромное число