



Самоучитель

Александр Леоненков

UML 2



Объектно-ориентированный анализ
и проектирование

Моделирование программных систем

Нотация и семантика
последней версии UML 2

UML в Borland® Together® Designer

Язык объектных ограничений OCL

Александр Леоненков

Самоучитель

UML 2

Санкт-Петербург

«БХВ-Петербург»

2007

УДК 681.3.06
ББК 32.973.26-018.1
Л147

Леоненков А. В.

Л147 Самоучитель UML 2. — СПб.: БХВ-Петербург, 2007. — 576 с.: ил.
ISBN 978-5-94157-878-8

Рассмотрена современная технология объектно-ориентированного анализа и проектирования программных систем и бизнес-процессов в контексте нотации унифицированного языка моделирования UML 2. Подробно изложены все понятия языка UML 2 в полном соответствии с оригинальной спецификацией последней версии этого языка. Приведены конкретные рекомендации по разработке канонических диаграмм языка и рассмотрены особенности разработки моделей с помощью CASE-средства Borland® Together® Designer. Описана нотация OCL — языка объектных ограничений, по которому практически отсутствует информация на русском.

*Для системных и бизнес-аналитиков, архитекторов программ,
руководителей проектов и информационных служб,
корпоративных программистов и студентов*

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.12.06.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 46,44.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

Оглавление

Предисловие	1
Структура книги	3
Рекомендации по изучению языка UML.....	4
Благодарности	5
Постскрипtum	6

ЧАСТЬ I. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО АНАЛИЗА И ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

7

Глава 1. Базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем

9

1.1. Основные понятия моделирования систем и программных приложений.....	10
1.2. Методология объектно-ориентированного анализа и проектирования	15
1.3. Концепция разработки архитектур, управляемых моделями	22
1.4. Основные этапы развития UML 2.0.....	25

Глава 2. Основные элементы нотации языка UML 2.0.....

34

2.1. Назначение языка UML 2.0	34
2.2. Общая структура языка UML 2.0.....	38
2.3. Пакеты в языке UML 2.0	42
2.4. Основные пакеты метамодели языка UML 2.0	44
2.4.1. Пакет <i>Абстракции</i>	45
2.4.2. Пакет <i>Основы</i>	45
2.4.3. Пакет <i>Конструкции</i>	46
2.4.4. Пакет <i>Простейшие Типы</i>	47
Boolean (Логический)	47
Integer (Целочисленный)	48
String (Строка)	48

<i>UnlimitedNatural</i> (Неограниченное натуральное число)	49
2.4.5. Пакет <i>Модели</i>	50
2.5. Особенности спецификации метамодели языка UML 2.0.....	51
2.6. Особенности изображения диаграмм в нотации UML 2.0	60
2.7. Механизмы расширения в языке UML 2.0	65
2.7.1. Стереотип	66
2.7.2. Ограничение.....	67
2.7.3. Помеченное значение.....	69

ЧАСТЬ II. ДИАГРАММЫ ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ ЯЗЫКА UML 2.0.....71

Глава 3. Диаграмма вариантов использования (use case diagram).....73

3.1. Диаграмма вариантов использования — исходная концептуальная модель проектируемой системы	73
3.1.1. Назначение диаграммы вариантов использования.....	74
3.1.2. Субъект вариантов использования	75
3.2. Основные графические элементы диаграммы вариантов использования.....	76
3.2.1. Вариант использования.....	77
3.2.2. Актер.....	79
3.2.3. Комментарий.....	82
3.3. Отношения на диаграмме вариантов использования	83
3.3.1. Отношение ассоциации.....	84
3.3.2. Отношение включения.....	86
3.3.3. Отношение расширения.....	88
3.3.4. Отношение обобщения	92
3.3.5. Пример диаграммы вариантов использования для системы продажи товаров в интернет-магазине	94
3.4. Формализация функциональных требований к системе с помощью диаграммы вариантов использования.....	97
3.4.1. Классификация требований в модели FURPS+	97
3.4.2. Спецификация функциональных требований с помощью текстовых сценариев	98
3.4.3. Пример сценария для системы продажи товаров в интернет-магазине	100

Глава 4. Диаграмма классов (class diagram)	104
4.1. Диаграмма классов — основная логическая модель проектируемой системы	104
4.2. Класс	107
4.2.1. Имя класса	110
4.2.2. Атрибуты класса	112
Вид видимости	114
Кратность	116
4.2.3. Операции класса	117
4.2.4. Параметр	120
4.3. Отношения между классами	123
4.3.1. Ассоциация	124
4.3.2. N-арная ассоциация	130
4.3.3. Ассоциация-класс	131
4.3.4. Квалификатор	133
4.3.5. Обобщение	134
4.3.6. Множество обобщения	137
4.3.7. Агрегация	140
4.3.8. Композиция	142
4.3.9. Зависимость	144
4.3.10. Реализация	145
4.4. Интерфейс	145
4.5. Шаблон	148
4.6. Диаграмма классов для системы продажи товаров в интернет-магазине	150
Глава 5. Диаграмма композитной структуры (composite structure diagram)	154
5.1. Композитная структура	154
5.2. Композитный класс	156
5.2.1. Часть	156
5.2.2. Соединитель	158
5.2.3. Роль в спецификации экземпляра класса	160
5.3. Порт класса	161
5.4. Кооперация	165
5.5. Применение кооперации	167
5.6. Шаблон кооперации	170
Глава 6. Дополнительные диаграммы структуры	175
6.1. Диаграмма пакетов	175
6.1.1. Пакет	176

6.1.2. Зависимость пакетов	178
6.1.3. Импорт пакета.....	179
6.1.4. Импорт элемента	180
6.1.5. Слияние пакетов	183
Общие правила слияния пакетов.....	187
Правила для пакетов	188
Правила для классов и типов данных	189
Правила для свойств	189
Правила для ассоциаций.....	190
Правила для операций	191
Правила для перечислений.....	191
Правила для ограничений	192
6.2. Диаграмма объектов	194
6.2.1. Объект.....	194
6.2.2. Спецификация экземпляра.....	195
6.2.3. Слот.....	197
6.2.4. Значение экземпляра	199
Глава 7. Диаграмма последовательности (sequence diagram)	201
7.1. Диаграмма последовательности — основная модель взаимодействия элементов проектируемой системы.....	202
7.2. Линия жизни	205
7.3. Сообщения и сигналы.....	207
7.3.1. Сообщение.....	208
7.3.2. Сигнал.....	211
7.4. Комбинированный фрагмент	213
7.4.1. Альтернативы (<i>alt</i>).....	216
7.4.2. Утверждение (<i>assert</i>)	217
7.4.3. Завершение (<i>break</i>).....	218
7.4.4. Критический регион (<i>critical</i>).....	219
7.4.5. Рассмотрение (<i>consider</i>)	220
7.4.6. Игнорирование (<i>ignore</i>).....	220
7.4.7. Цикл (<i>loop</i>)	222
7.4.8. Отрицание (<i>neg</i>).....	223
7.4.9. Необязательный (<i>opt</i>)	224
7.4.10. Параллельный (<i>par</i>).....	225
7.4.11. Слабое следование (<i>seq</i>).....	225
7.4.12. Строгое следование (<i>strict</i>)	226
7.5. Специальные фрагменты и элементы взаимодействия	226
7.5.1. Использование взаимодействия	226

7.5.2. Декомпозиция части.....	228
7.5.3. Инвариант состояния.....	230
7.5.4. Продолжение.....	232
7.5.5. Шлюз.....	234
7.6. Специальные ограничения на диаграммах последовательности.....	235
7.6.1. Временное выражение	236
7.6.2. Временное событие	236
7.6.3. Действие наблюдения времени	237
7.6.4. Интервал	238
7.6.5. Временное ограничение.....	239
7.6.6. Продолжительность.....	239
7.6.7. Действие наблюдения продолжительности	240
7.6.8. Ограничение на продолжительность.....	241
Глава 8. Диаграмма деятельности (activity diagram).....	245
8.1. Концептуальные основы моделирования деятельности.....	245
8.1.1. Деятельность и действие.....	246
8.1.2. Узлы и дуги деятельности	247
8.1.3. Семантика деятельности.....	251
8.1.4. Семантика действия	252
8.2. Узлы управления	254
8.2.1. Начальный узел.....	254
8.2.2. Узел финала деятельности и потока	255
8.2.3. Узел решения	256
8.2.4. Узел слияния	258
8.2.5. Узел разделения.....	259
8.2.6. Узел соединения	260
8.3. Специальные действия	262
8.3.1. Действие передачи сигнала	262
8.3.2. Действие приема события.....	263
8.4. Узлы потока объектов.....	265
8.4.1. Узел объекта.....	265
8.4.2. Центральный буфер и хранилище данных.....	267
8.4.3. Входные и выходные контакты объектов	269
8.4.4. Узел параметра деятельности.....	271
8.4.5. Множество параметров	273
8.5. Специальные регионы	275
8.5.1. Разбиение деятельности.....	275
8.5.2. Регион прерываемой деятельности.....	278
8.5.3. Обработчик исключения.....	280

Глава 9. Вспомогательные диаграммы взаимодействия.....	285
9.1. Диаграмма коммуникации (communication diagram).....	286
9.1.1. Линия жизни.....	288
9.1.2. Связь.....	289
9.1.3. Сообщение.....	290
9.1.4. Формат записи сообщений.....	292
9.1.5. Модель коммуникации.....	295
9.2. Диаграмма обзора взаимодействия (interaction overview diagram).....	298
9.3. Временная диаграмма (timing diagram).....	301
9.3.1. Основные элементы временной диаграммы.....	302
9.3.2. Первая форма временной диаграммы.....	304
9.3.3. Вторая форма временной диаграммы.....	305
9.3.4. Третья форма временной диаграммы.....	306
 Глава 10. Диаграмма конечного автомата	
(state machine diagram).....	308
10.1. Концептуальные основы моделирования конечных автоматов в языке UML 2.0.....	309
10.2. Простое состояние.....	314
10.2.1. Секция имени.....	315
10.2.2. Секция внутренней деятельности.....	315
10.2.3. Секция внутренних переходов.....	316
10.2.4. Отложенные события.....	317
10.3. Псевдосостояния.....	317
10.3.1. Начальное псевдосостояние.....	318
10.3.2. Узел завершения.....	319
10.3.3. Выбор.....	319
10.3.4. Соединение.....	321
10.3.5. Разделение.....	322
10.3.6. Слияние.....	323
10.3.7. Точка входа.....	324
10.3.8. Точка выхода.....	324
10.3.9. Неглубокая история.....	325
10.3.10. Глубокая история.....	327
10.3.11. Финальное состояние.....	329
10.4. Переход.....	329
10.4.1. Сторожевое условие.....	330
10.4.2. Переходы завершения и события завершения.....	331

10.4.3. Составные переходы	332
10.4.4. Передача сигнала	333
10.4.5. Прием сигнала.....	333
10.4.6. Действия на переходе.....	334
10.4.7. Правила разрешения и срабатывания переходов	335
10.4.8. Конфликтующие переходы.....	337
10.5. Композитные состояния и регионы.....	339
10.5.1. Основные определения	339
10.5.2. Вход и выход в простом композитном состоянии	341
10.5.3. Вход и выход в ортогональном композитном состоянии	344
10.5.4. Скрытая секция декомпозиции	345
10.6. Состояние подавтомата	347
10.7. Протокольный конечный автомат	350
10.7.1. Протокольное состояние.....	351
10.7.2. Протокольный переход	352
Глава 11. Диаграмма компонентов (component diagram)	356
11.1. Особенности физического моделирования в языке UML 2.0	356
11.2. Компонент.....	360
11.3. Интерфейс	364
11.4. Порт	366
11.5. Соединитель.....	368
11.5.1. Собирающий соединитель.....	368
11.5.2. Делегирующий соединитель	370
11.6. Зависимость	373
11.7. Реализация	375
11.8. Стереотипы компонентов.....	376
Глава 12. Диаграмма развертывания (deployment diagram)	382
12.1. Узел.....	384
12.1.1. Среда выполнения	385
12.1.2. Устройство	387
12.2. Артефакт	388
12.3. Спецификация развертывания	390
12.4. Отношения на диаграмме развертывания.....	392
12.4.1. Развертывание	393
12.4.2. Манифестация.....	395
12.4.3. Путь коммуникации	396
12.5. Стереотипы узлов.....	397

ЧАСТЬ III. АНАЛИЗ И ПРОЕКТИРОВАНИЕ**С ИСПОЛЬЗОВАНИЕМ НОТАЦИИ UML 2.0
И CASE-СРЕДСТВА BORLAND® TOGETHER®****DESIGNER 2005401****Глава 13. Особенности реализации графической нотации
языка UML 2.0 в среде Borland® Together® Designer 2005403**

13.1. Общая характеристика CASE-средства Borland® Together® Designer 2005	404
13.2. Особенности рабочего интерфейса Borland Together Designer 2005	406
13.2.1. Главное меню	407
13.2.2. Стандартная панель инструментов	408
13.2.3. Окно проекта.....	410
13.2.4. Окно навигатора модели.....	411
13.2.5. Окно навигатора диаграмм.....	412
13.2.6. Окно инспектора.....	413
13.2.7. Окно диаграммы	414
13.2.8. Стандартная панель инструментов окна диаграммы модели.....	415
13.2.9. Специальная панель инструментов диаграммы модели.....	417
13.2.10. Окно истории	418
13.3. Назначение операций главного меню	419
13.3.1. Пункт меню <i>File</i> (Файл).....	419
13.3.2. Пункт меню <i>Edit</i> (Редактирование)	421
13.3.3. Пункт меню <i>Search</i> (Поиск).....	422
13.3.4. Пункт меню <i>View</i> (Вид).....	422
13.3.5. Пункт меню <i>Project</i> (Проект)	423
13.3.6. Пункт меню <i>Diagram</i> (Диаграмма).....	424
13.3.7. Пункт меню <i>Team</i> (Команда).....	425
13.3.8. Пункт меню <i>Tools</i> (Инструменты)	428
13.3.9. Пункт меню <i>Window</i> (Окно)	429
13.3.10. Пункт меню <i>Help</i> (Справка)	429

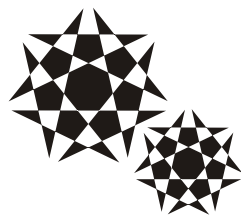
**Глава 14. Организация работы над проектом
в среде Borland Together Designer430**

14.1. Разработка диаграммы вариантов использования в среде Together Designer.....	431
14.1.1. Создание нового проекта и новой диаграммы вариантов использования.....	431
14.1.2. Добавление актеров.....	436
14.1.3. Добавление границы системы	437

14.1.4. Добавление вариантов использования	438
14.1.5. Добавление ассоциаций	439
14.1.6. Добавление зависимостей.....	441
14.1.7. Добавление текстового файла со сценарием варианта использования	443
14.2. Разработка диаграммы классов в среде Together Designer	444
14.2.1. Добавление классов.....	446
14.2.2. Добавление атрибутов классов	447
14.2.3. Добавление операций классов.....	448
14.2.4. Добавление отношений на диаграмму классов	453
14.3. Разработка диаграммы композитной структуры в среде Together Designer.....	455
14.3.1. Добавление классов и частей	457
14.3.2. Добавление портов и интерфейсов	458
14.3.3. Добавление отношений на диаграмму композитной структуры	460
14.4. Разработка диаграммы последовательности в среде Together Designer.....	460
14.4.1. Добавление линий жизни.....	461
14.4.2. Добавление сообщений.....	462
14.4.3. Добавление комбинированных фрагментов	464
14.5. Разработка диаграммы коммуникации в среде Together Designer	468
14.5.1. Добавление линий жизни.....	470
14.5.2. Добавление сообщений.....	471
Глава 15. Завершение разработки проекта в среде Borland Together Designer	474
15.1. Разработка диаграммы деятельности в среде Together Designer.....	474
15.1.1. Добавление действий и деятельностей.....	476
15.1.2. Добавление потока управления.....	477
15.2. Разработка диаграммы конечного автомата в среде Together Designer.....	479
15.2.1. Добавление состояний	481
15.2.2. Добавление переходов	482
15.3. Разработка диаграммы компонентов в среде Together Designer	484
15.3.1. Добавление компонентов.....	485
15.3.2. Добавление отношений на диаграмму компонентов	488
15.4. Разработка диаграммы развертывания в среде Together Designer	488
15.4.1. Добавление узлов, сред выполнения и компонентов.....	490
15.4.2. Добавление отношений на диаграмму развертывания	491
15.5. Генерация документации и программного кода в среде Together Designer.....	493

Заключение.....	497
ПРИЛОЖЕНИЯ	499
Приложение 1. Язык объектных ограничений OCL.....	501
П1.1. Выражения языка OCL	503
П1.2. Основные типы значений и операций в языке OCL	505
П1.3. Операции над отдельными типами значений	507
П1.3.1. Операции с действительными числами	507
П1.3.2. Операции с целыми числами	509
П1.3.3. Операции со строками.....	511
П1.3.4. Операции с булевыми выражениями	512
П1.3.5. Операция <i>@pre</i> для указания предшествующих элементов.....	514
П1.4. Допустимые выражения в языке OCL.....	514
П1.5. Неопределенное выражение.....	515
П1.6. Коллекции значений в языке OCL.....	515
П1.7. Операции над коллекциями значений	516
П1.7.1. Операция выбора <i>select</i>	516
П1.7.2. Операция исключения <i>reject</i>	517
П1.7.3. Операция формирования коллекции <i>collect</i>	517
П1.7.4. Операция "для всех" <i>forAll</i>	518
П1.7.5. Операция "существует" <i>exists</i>	518
П1.7.6. Другие операции над коллекциями значений	519
П1.8. Некоторые операции с множествами, последовательностями и комплектами	520
П1.9. Операции преобразования типов	521
П1.10. Примеры записи выражений языка OCL	522
П1.10.1. Определение значения переменной	522
П1.10.2. Определение возраста сотрудника	522
П1.10.3. Определение кратности значений	522
П1.10.4. Определение коллекции инвариантов	522
Приложение 2. Глоссарий.....	524
Литература	547
Предметный указатель	551

Глава 2



Основные элементы нотации языка UML 2.0

Язык UML 2.0 обладает весьма широкими возможностями, которые предназначены для решения разнообразных задач визуального моделирования из самых различных прикладных областей. Однако, обладая значительной избыточностью, не все его возможности моделирования являются обязательно необходимыми при разработке конкретных приложений. Соответствующая согласованность используемых конструкций языка UML 2.0 обеспечивается его модульным построением. Все это позволяет разработчикам выбирать только те элементы, которые представляют прямой интерес в контексте разрабатываемого проекта.

Язык UML 2.0 основан на некотором числе базовых понятий, которые могут быть изучены и применены большинством программистов и разработчиков, знакомых с методами объектно-ориентированного анализа и проектирования. При этом базовые понятия могут комбинироваться и расширяться таким образом, что специалисты объектного моделирования получают возможность самостоятельно разрабатывать модели больших и сложных систем в самых различных областях приложений.

2.1. Назначение языка UML 2.0

Язык UML 2.0, также как и его предыдущие версии, предназначен для решения следующих задач.

- Предоставить в распоряжение всех пользователей легко воспринимаемую и выразительную нотацию для визуального моделирования, специально предназначенную для разработки и документирования моделей сложных систем самого различного целевого назначения.
 - Речь идет о том, что важным фактором дальнейшего развития и повсеместного использования методологии ООАП является интуитивная

ясность и понятность основных конструкций соответствующего языка моделирования. Язык UML 2.0 включает в себя не только абстрактные конструкции, для представления метамodelей систем, но и целый ряд конкретных понятий, имеющих вполне определенную семантику. Это позволяет языку UML 2.0 одновременно достичь не только универсальности представления моделей для самых различных приложений, но и возможности описания достаточно тонких деталей реализации этих моделей применительно к конкретным системам.

- Для адекватного понимания базовых конструкций языка UML 2.0 важно не только владеть навыками объектно-ориентированного программирования, но и хорошо представлять себе общую проблематику процесса разработки моделей систем. Именно интеграция этих представлений образует новую парадигму в методологии ООАП, практическим следствием и центральным стержнем которой является язык UML 2.0.
- Снабдить исходные понятия языка UML 2.0 возможностью расширения и специализации для более точного представления моделей систем в конкретной предметной области.
- Хотя язык UML 2.0 является формальным языком спецификаций, формальность его описания отличается от синтаксиса как традиционных формально-логических языков, так и известных языков программирования. Разработчики из OMG предполагают, что язык UML 2.0 как никакой другой может быть приспособлен для конкретных предметных областей. Это становится возможным по той причине, что в самом описании языка UML 2.0 заложен механизм расширения базовых понятий, который является самостоятельным элементом и имеет собственное описание в форме правил расширения.
- В то же время разработчики из OMG считают крайне нежелательным переопределение базовых понятий языка. Это может привести к неоднозначной интерпретации их семантики и возможной путанице. Базовые понятия языка UML 2.0 не следует изменять больше, чем это необходимо для их расширения. Все пользователи должны быть способны строить модели систем для большинства обычных приложений с использованием только базовых конструкций языка UML 2.0 без применения механизма расширения. При этом новые понятия и графические обозначения целесообразно применять только в тех ситуациях, когда имеющихся базовых конструкций явно недостаточно для построения моделей системы.
- Язык UML 2.0 допускает также специализацию базовых понятий. Речь идет о том, что в конкретных приложениях пользователи должны уметь

дополнять имеющиеся базовые понятия новыми характеристиками или свойствами, которые не противоречат семантике этих понятий.

- Описание языка UML 2.0 должно поддерживать такую спецификацию моделей, которая не зависит от конкретных языков программирования и инструментальных средств проектирования программных систем.
 - Речь идет о том, что конструкции языка UML 2.0 не должны зависеть от особенностей их реализации на той или иной платформе или языке программирования. Другими словами, хотя отдельные понятия языка UML 2.0 семантически связаны с последними, их жесткая интерпретация в форме конструкций программирования не может быть признана корректной. Разработчики из OMG считают необходимым свойством языка UML 2.0 его контекстно-программную и платформенную независимость.
 - С другой стороны, язык UML 2.0 должен обладать потенциальной возможностью реализации своих конструкций на том или ином языке программирования. Конечно, в первую очередь имеются в виду языки, поддерживающие концепцию ООП, такие как Java, C++, C#, Object Pascal/Delphi. Именно это свойство языка UML 2.0 делает его современным и конструктивным средством решения задач моделирования сложных систем. В то же время, предполагается, что для программной поддержки конструкций языка UML 2.0 могут быть разработаны специальные инструментальные CASE-средства. Наличие последних имеет принципиальное значение для широкого распространения и использования языка UML 2.0.
- Описание языка UML 2.0 должно включать в себя семантический базис для понимания общих особенностей ООАП.
 - Говоря об этой особенности, имеют в виду самодостаточность языка UML 2.0 для понимания не только его базовых конструкций, но что не менее важно — понимания общих принципов ООАП. В этой связи необходимо отметить, что поскольку язык UML 2.0 не является языком программирования, а служит средством для решения задач объектно-ориентированного моделирования систем, описание языка UML 2.0 должно по возможности включать в себя все необходимые понятия методологии ООАП.
 - С другой стороны, какие бы то ни было ссылки на дополнительные источники, содержащие важную для понимания языка UML 2.0 информацию, по мнению разработчиков из OMG, должны быть исключены. Это позволит избежать неоднозначного толкования принципиальных особенностей методологии ООАП и их реализации в форме базовых конструкций языка UML 2.0.

- Поощрять развитие рынка программных инструментальных средств, поддерживающих методологию ООАП и концепцию MDA.
 - Более 800 ведущих производителей программных и аппаратных средств, усилия которых сосредоточены в рамках OMG, видят перспективы развития современных информационных технологий и основу своего коммерческого успеха в широком продвижении на рынок инструментальных средств, поддерживающих объектные технологии. С этой точки зрения языку UML 2.0 отводится роль базового концептуального средства для описания и документирования различных объектных технологий.
- Способствовать распространению объектных технологий и соответствующих понятий ООАП.
 - Если попытаться составить объективную картину возможностей языка UML 2.0, то можно прийти к следующему заключению. Следует признать, что усилия достаточно большой группы разработчиков, направленные на интеграцию в рамках языка UML 2.0 многих известных техник визуального моделирования, привели к серьезному усложнению языка UML 2.0 по сравнению с другими известными нотациями и его предыдущими версиями. Платой за сложность является действительно высокая гибкость и широкие изобразительные возможности языка UML 2.0. В свою очередь, использование языка UML 2.0 для решения всевозможных практических задач будет только способствовать его дальнейшему совершенствованию, а значит, и дальнейшему развитию объектных технологий и практики ООАП.
- Интегрировать в себя новейшие и наилучшие достижения практики ООАП.

Язык UML непрерывно совершенствуется разработчиками, и основой этой работы является его дальнейшая интеграция с современными технологиями моделирования. При этом различные методы системного моделирования получают свое прикладное осмысление в контексте методологии ООАП. В последующем эти методы могут быть включены в состав очередных версий языка UML в форме дополнительных базовых понятий, наиболее адекватно и полно отражающие наилучшие достижения практики ООАП.

Чтобы решить указанные выше задачи, за свою историю язык UML претерпел определенную эволюцию. В результате описание самого языка UML 2.0 стало нетривиальным, поскольку семантика базовых понятий включает в себя целый ряд перекрестных связей с другими понятиями и конструкциями языка. В связи с этим последовательное рассмотрение основных конструкций

языка UML 2.0 стало практически невозможным, т. к. одни и те же понятия могут использоваться при построении различных диаграмм и представлений. В то же время каждое из представлений модели обладает собственными семантическими особенностями, которые накладывают отпечаток на семантику базовых понятий языка в целом.

Как результат, разработчикам нет необходимости знать весь язык UML 2.0, чтобы эффективно его использовать. Кроме того, большинство единиц языка может быть разделено на многократные приращения, каждое из которых добавляет к предыдущим дополнительные возможности моделирования. Такая дискретная декомпозиция языка UML 2.0 предназначена для того, чтобы сделать более легким его изучение и использование на практике. При этом отдельные приращения описываются в форме слияния пакетов, что находит свое выражение при рассмотрении отдельных типов канонических диаграмм.

Учитывая эти особенности, принятая в книге последовательность изучения языка UML 2.0 основывается на рассмотрении основных графических средств моделирования, а именно — канонических диаграмм. Все понятия, необходимые для построения соответствующих диаграмм, вводятся по мере необходимости. В данной главе приводится описание общих особенностей языка UML 2.0, которые в той или иной степени оказывают влияние на понимание всех его базовых конструкций и канонических диаграмм.

2.2. Общая структура языка UML 2.0

С самой общей точки зрения описание языка UML 2.0 состоит из двух взаимодействующих частей: семантики и нотации.



Семантика (semantics) — система правил и соглашений, определяющих толкование и придание смысла конструкциям некоторого языка.

Нотация (notation) — система условных обозначений, принятая в некотором языке для изображения и визуализации модели.

В рамках языка UML 2.0 семантика и нотация тесно взаимосвязаны. Семантика языка UML 2.0 описывается с использованием некоторого подмножества нотации UML 2.0. В свою очередь, нотация UML 2.0 описывает соответствие или отображение графической нотации в базовые понятия семантики. Таким образом, с функциональной точки зрения эти две части дополняют друг друга. При этом семантика языка UML 2.0 описывается на основе некоторой метамодели, имеющей три отдельных представления: абстрактный

синтаксис, правила корректного построения выражений и семантику. Рассмотрение семантики языка UML 2.0 предполагает некоторый "полуформальный" стиль изложения, который объединяет естественный и формальный языки для представления базовых понятий и правил их расширения.

Семантика определяется для двух категорий объектных моделей: структурных моделей (статических моделей) и моделей поведения (динамических моделей).



Структурные модели (structured models) — модели, предназначенные для описания статической структуры сущностей или элементов некоторой системы, включая их классы, интерфейсы, атрибуты и отношения.

Модели поведения (behavioral models) — модели, предназначенные для описания процесса функционирования элементов системы, включая их методы и взаимодействие между ними, а также процесс изменения состояний отдельных элементов и системы в целом.

Для решения самого широкого диапазона задач моделирования в языке UML 2.0 разработана достаточно полная семантика для всех элементов графической нотации. При этом требования семантики языка UML 2.0 конкретизируются при построении отдельных типов диаграмм, последовательное рассмотрение которых служит темой *части II* книги. Нотация языка UML 2.0 включает в себя описание отдельных семантических элементов, которые могут применяться при построении диаграмм.

Формальное описание самого языка UML 2.0 основывается на некоторой общей иерархической структуре модельных представлений, состоящей из четырех уровней:

- мета-метамодель (M3);
- метамодель (M2);
- модель (M1);
- объекты или экземпляры (M0).

Уровень *мета-метамодели* образует исходную формально-логическую основу для всех возможных метамодельных представлений. Главное предназначение этого уровня состоит в том, чтобы определить язык для спецификации метамодели. Мета-метамодель определяет модель языка UML 2.0 на самом высоком уровне абстракции и является наиболее компактным ее описанием. С другой стороны, мета-метамодель может специфицировать несколько метамоделей, чем достигается потенциальная гибкость включения дополни-

тельных понятий. Хотя в книге этот уровень не рассматривается, он наиболее тесно связан с теорией формальных языков. Примерами понятий этого уровня служат метакласс, метаатрибут, метаоперация.

Метамодель является экземпляром или конкретизацией мета-метамодели. Главная задача этого уровня — определить язык для спецификации моделей. Данный уровень является более конструктивным, чем предыдущий, поскольку обладает более развитой семантикой базовых понятий. Все основные понятия языка UML 2.0 — это понятия уровня метамодели. Примерами таких понятий являются класс, атрибут, операция, компонент, ассоциация и многие другие. Рассмотрению семантики и графической нотации понятий уровня метамодели собственно и посвящена настоящая книга.

Модель в контексте языка UML 2.0 является экземпляром метамодели в том смысле, что любая конкретная модель системы должна использовать только понятия метамодели, конкретизировав их применительно к данной ситуации. Это уровень для описания информации о конкретной предметной области. Однако поскольку при построении модели используются понятия языка UML 2.0, то необходима полная согласованность понятий уровня модели с базовыми понятиями языка UML 2.0 уровня метамодели. Примерами понятий уровня модели могут служить, например, имена полей проектируемой базы данных, такие как: имя и фамилия сотрудника, возраст, должность, адрес, телефон. При этом данные понятия используются лишь как имена соответствующих информационных атрибутов.

Конкретизация понятий модели происходит на уровне *объектов* или *экземпляров*. При этом совокупность объектов рассматривается как отдельный экземпляр модели времени выполнения, поскольку содержит конкретную информацию относительно того, чему в действительности соответствуют те или иные понятия модели. Примером объекта может служить следующая запись в проектируемой базе данных: "Игорь Греков, 40 лет, модельер, ул. Системная, 1-2, 123-4567". Рассмотренную 4-уровневую структуру модельных представлений языка UML 2.0 можно проиллюстрировать следующим образом (рис. 2.1).

Описание семантики языка UML 2.0 предполагает рассмотрение базовых понятий только уровня *метамодели* (M_2), который представляет собой пример или частный случай уровня мета-метамодели. Метамодель UML 2.0 является по своей сути скорее логической моделью, чем физической или моделью реализации. Особенность логической модели заключается в том, что она концентрирует внимание на декларативной или концептуальной семантике, опуская детали конкретной физической реализации моделей. При этом отдельные реализации, использующие данную логическую метамодель, должны быть согласованы с ее семантикой, а также поддерживать возможности импорта и экспорта отдельных логических моделей.

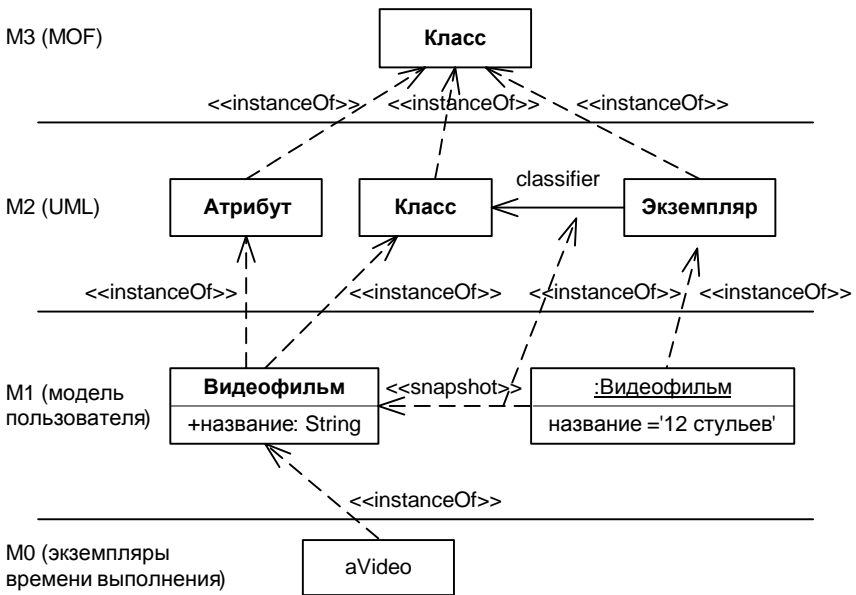


Рис. 2.1. Иллюстрация 4-уровневой структуры модельных представлений языка UML 2.0

В то же время, логическая метамодель может быть реализована различными способами для обеспечения требуемого уровня производительности и надежности соответствующих инструментальных средств. В этом заключается недостаток логической модели, которая не содержит на уровне семантики требований, обязательных для ее эффективной последующей реализации. Однако согласованность метамодели с конкретными моделями реализации является обязательной для всех разработчиков программных средств, обеспечивающих поддержку языка UML 2.0.

Метамодель языка UML 2.0 имеет довольно сложную структуру, которая включает в себя метаклассы, метаассоциации и стереотипы, число которых возрастает с появлением новых версий языка. Чтобы справиться с этой сложностью языка UML 2.0, все его элементы организованы специальным образом в логически связанные подмножества, получившие название пакетов. Поэтому рассмотрение языка UML 2.0 на метамодельном уровне заключается в описании его наиболее общих логических блоков или *пакетов*.



Говоря о пакетах в контексте общего описания языка, мы, по сути дела, приступаем к рассмотрению графической нотации языка UML 2.0. При этом следует заметить, что для описания языка UML 2.0 используются средства самого этого языка. Одним из таких средств является *пакет*, который в общем случае служит для группировки любых элементов модели. Пакеты могут находиться в различных отношениях и быть вложенными друг в друга.

2.3. Пакеты в языке UML 2.0

Пакет — основной способ организации элементов метамодели языка UML 2.0. Каждый пакет владеет всеми своими элементами, т. е. теми элементами, которые включены в него. Про соответствующие элементы пакета говорят, что они принадлежат пакету или входят в него. В языке UML 2.0 для визуализации пакетов разработана специальная символика или графическая нотация, которой мы и будем пользоваться в дальнейшем. Именно с описания этой системы обозначений мы приступим к изучению основных элементов языка.

Для графического изображения пакетов на диаграммах применяется специальный графический символ — большой прямоугольник с небольшим прямоугольником, присоединенным к левой части верхней стороны первого (рис. 2.2). Визуально символ пакета напоминает пиктограмму папки на экране монитора. Внутри большого прямоугольника могут изображаться элементы, которые принадлежат этому пакету. Если такие элементы не указаны, то внутри большого прямоугольника записывается имя пакета, которое должно быть уникальным в пределах рассматриваемой модели (рис. 2.2, а). Если же элементы пакета указаны, то имя пакета записывается в верхнем маленьком прямоугольнике (рис. 2.2, б).



Рис. 2.2. Графическое изображение пакетов в языке UML 2.0

Говоря об имени пакета, следует остановиться на общем соглашении об именах в языке UML 2.0. В данном случае именем пакета может быть строка текста, содержащая произвольное число букв, цифр и некоторых специальных знаков. С целью удобства спецификации пакетов принято в качестве их имен использовать существительные, возможно, с пояснительными словами, например, контроллер, графический интерфейс, форма ввода данных.

Сами по себе пакеты находят ограниченное применение, поскольку содержат лишь информацию о входящих в их состав элементах модели. Не менее важно представить графически отношения, которые могут иметь место между отдельными пакетами. Как и в теории графов, для визуализации отношений

в языке UML 2.0 применяются отрезки линий, внешний вид которых имеет смысловое содержание или семантику.

Одним из типов отношений между пакетами является отношение вложенности или включения пакетов друг в друга. С одной стороны, в языке UML 2.0 это отношение может быть изображено без использования линий простым размещением одного пакета-прямоугольника внутри другого пакета-прямоугольника (рис. 2.3). Так, в данном случае `Пакет А` содержит в себе два подпакета: `Пакет В` и `Пакет С`.

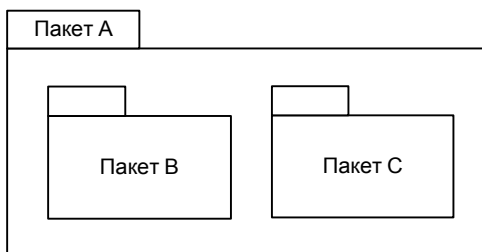


Рис. 2.3. Графическое изображение вложенности пакетов

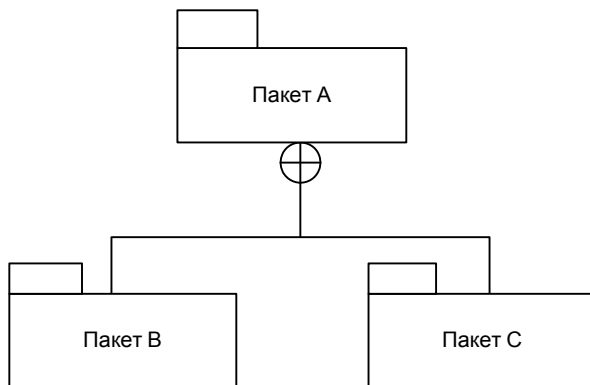


Рис. 2.4. Альтернативное графическое изображение вложенности пакетов

С другой стороны, это же отношение может быть изображено с помощью отрезков линий аналогично графическому представлению дерева. В этом случае пакет, содержащий другие пакеты, или пакет-контейнер изображается в верхней части рисунка, а его подпакеты — уровнем ниже. Пакет-контейнер соединяется с подпакетами сплошной линией, на конце которой, примыкающей к пакету контейнеру, изображается специальный символ — \oplus . Этот сим-

вол означает, что подпакеты являются "собственностью" или частью контейнера, и, кроме этих подпакетов, контейнер не содержит никаких других подпакетов. Рассмотренный выше пример (см. рис. 2.3) может быть представлен альтернативно следующим образом (рис. 2.4).

На графических диаграммах между пакетами могут указываться и другие типы отношений, которые будут описаны при рассмотрении диаграммы пакетов (см. главу 6).

2.4. Основные пакеты метамодели языка UML 2.0

Основой представления языка UML 2.0 на метамодельном уровне является спецификация его основных логических частей в форме пакетов: Элементы ядра, Абстракции, Основы, Конструкции и Простейшие Типы (рис. 2.5).

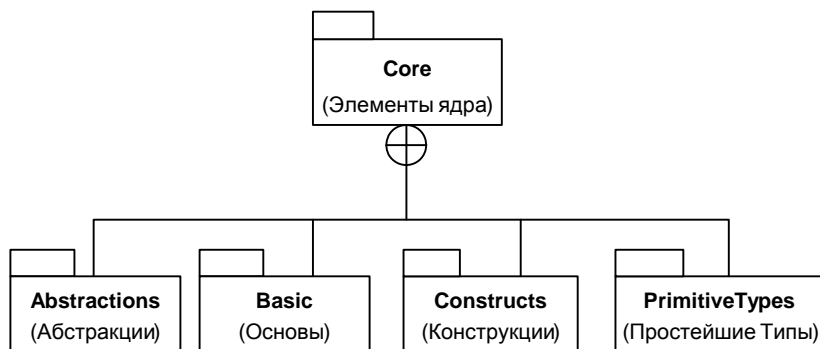


Рис. 2.5. Основные пакеты метамодели языка UML 2.0

Эти пакеты, в свою очередь, также включают в себя другие подпакеты, которые содержат описание фундаментальных понятий языка UML 2.0. Далее дается краткая характеристика элементов каждого из перечисленных подпакетов, входящих в состав пакета Элементы ядра. Более полное рассмотрение отдельных пакетов метамодели будет представлено в главах, посвященных изучению отдельных видов канонических диаграмм. Последние аккумулируют в себе не только различные представления моделируемой системы, но и более детально раскрывают семантические особенности применения базовых конструкций языка UML 2.0 в процессе построения конкретных моделей.

2.4.1. Пакет Абстракции

Пакет Абстракции является наиболее фундаментальным из всех подпакетов, которые входят в пакет Элементы ядра языка UML 2.0. Он определяет основные абстрактные элементы, необходимые для разработки объектных моделей. При этом абстрактные элементы метамодели не имеют экземпляров или примеров и используются исключительно для уточнения других элементов модели.

В этот пакет входят основные подпакеты языка UML 2.0: Elements (Элементы), Ownerships (Права собственности), Namespaces (Пространства имен), Comments (Комментарии), Relationships (Отношения), Visibilities (Видимости), Expressions (Выражения), StructuralFeatures (Структурные характеристики), BehavioralFeatures (Характеристики поведения), Classifiers (Классификаторы), Constraints (Ограничения), Generalizations (Обобщения), Instances (Экземпляры), MultiplicityExpressions (Выражения кратности), Redefinitions (Переопределения), Changeabilities (Способности к изменениям), Super (Верхний уровень), Literals (Литералы), Multiplicities (Кратности), TypedElements (Типизированные элементы).

2.4.2. Пакет Основы

Пакет Основы содержит минимальное подмножество элементов языка UML 2.0, которые называются *метаклассами* и используются для дальнейшей конкретизации при моделировании структуры систем. Метаклассы данного пакета и отношения между ними описывают следующие диаграммы:

- диаграмма типов, которая специфицирует метаклассы, имеющие отношение к именованным и типизированным элементам: Type (Тип), Element (Элемент), TypedElement (Типизированный элемент), NamedElement (Именованный элемент), Comment (Комментарий);
- диаграмма классов, которая определяет конструкции, необходимые для моделирования классов: Class (Класс), Property (Свойство), Operation (Операция), Parameter (Параметр), MultiplicityElement (Элемент кратности);
- диаграмма типов данных, которая специфицирует метаклассы, которые определяют типы данных: Type (Тип), PrimitiveType (Простейший Тип), Enumeration (Перечисление), DataType (Тип Данных);
- диаграмма пакетов, которая определяет базовые конструкции, имеющие отношение к пакетам и их содержимому: Package (Пакет) и Type (Тип).