

MCS D

Training Kit

Exam 70-300

ANALYZING

REQUIREMENTS

AND DEFINING

MICROSOFT® .NET

SOLUTION ARCHITECTURES

Microsoft Press

Учебный
курс
MCS D

Сертификационный
экзамен 70-300

**АНАЛИЗ ТРЕБОВАНИЙ
И СОЗДАНИЕ
АРХИТЕКТУРЫ РЕШЕНИЙ
НА ОСНОВЕ
MICROSOFT® .NET**

*Официальное пособие Microsoft
для самостоятельной подготовки*

Москва, 2004

 РУССКАЯ РЕДАКЦИЯ

УДК 004.45
ББК 32.973.26-018.2
М59

Microsoft Corporation
М59 Анализ требований и создание архитектуры решений на основе Microsoft .NET. Учебный курс MCSD/Пер. с англ. -- М.: Издательско-торговый дом «Русская Редакция», 2004. — 416 стр.: ил.

ISBN 5-7502-0248-8

Этот учебный курс посвящен созданию архитектуры программных решений в среде Microsoft .NET. Вы познакомитесь с каркасом Microsoft Solutions Framework (MSF), в том числе с моделью процессов MSF и ее этапами: созданием общей картины решения, планированием, разработкой процесса тестирования, стабилизацией и развертыванием приложения. Вы научитесь собирать и анализировать информацию для проектируемого бизнес-решения, планировать архитектуру программного продукта, а также его масштабируемость, доступность, производительность, способность к взаимодействию, поддержку других языков и административные функции. Кроме того, вы узнаете, как планируют тестирование, стабилизацию, развертывание и поддержку готового решения.

Учебный курс предназначен специалистам в области информационных технологий, занимающимся проектированием, разработкой и внедрением ИТ-решений в Windows-среде с использованием инструментов и технологий Microsoft, а также всем, кто хочет получить исчерпывающие знания в области проектирования ПО. Помимо теоретического материала курс содержит упражнения и контрольные вопросы для самопроверки. Он поможет вам подготовиться к экзамену по программе сертификации Microsoft (Microsoft Certified Solution Developer, MCSD) № 70-300: «Analyzing Requirements and Defining Microsoft .NET Solution Architectures».

Книга состоит из 11 глав, приложения и предметного указателя. На прилагаемом компакт-диске находятся демонстрационные файлы, практические задания, словарь терминов и другие справочные материалы.

УДК 004.45
ББК 32.973.26-018.2

Подготовлено к изданию по лицензионному договору с Microsoft Corporation, Редмонд, Вашингтон, США. Active Directory, ActiveX, Autenticode, BizTalk, Microsoft, Microsoft Press, MS-DOS, MSDN, Outlook, PowerPoint, SharePoint, Visio, Visual Basic, Visual C++, Visual C#, Visual Studio, Windows, Windows Media и Windows NT являются товарными знаками или охраняемыми товарными знаками корпорации Microsoft в США и/или других странах. Все другие товарные знаки являются собственностью соответствующих фирм.

Все названия компаний, организаций и продуктов, а также имена лиц, используемые в примерах, вымышлены и не имеют никакого отношения к реальным компаниям, организациям, продуктам и лицам.

- © Оригинальное издание на английском языке, Microsoft Corporation, 2003
- © Перевод на русский язык, Microsoft Corporation, 2004
- © Оформление и подготовка к изданию, издательско-торговый дом «Русская Редакция», 2004

ISBN 0-7356-1894-1 (англ.)
ISBN 5-7502-0248-8

Microsoft Corporation

Анализ требований и создание архитектуры решений на основе Microsoft .NET

Учебный курс MCSD

Перевод с английского под общей

редакцией **А. Р. Врублевского**

Редактор **Ю. П. Леонова**

Технический редактор **Н. Г. Тимченко**

Корректор **Л. А. Панчук**

Компьютерная верстка **В. Ю. Барыбин**

Дизайнер обложки **Е. В. Козлова**

Главный редактор **А. И. Козлов**

Подготовлено к печати издательством

«Русская Редакция»

121087, Москва, ул. Антонова-Овсеенко, д. 13

тел.: (095) 256-5120, тел./факс: (095) 256-4541

e-mail: info@rusedit.ru, http://www.rusedit.ru

 РУССКАЯ РЕДАКЦИЯ

Подписано в печать 24.02.2004 г. Тираж 3000 экз.
Формат 70×100/16 Физ. п. л. 26

Отпечатано в ОАО «Типография «Новости»»
105005, Москва, ул. Фр. Энгельса, 46

Содержание

Об этой книге	XIII
Глава 1. Введение в проектирование бизнес-решений	1
Занятие 1. Базовые сведения о Microsoft Solutions Framework.....	2
Модели процессов.....	2
Как работает модель процессов MSF.....	3
Организация проектных команд.....	5
Дисциплины MSF.....	6
Управление компромиссами.....	9
Порядок применения итераций в проектах.....	11
Занятие 2. Этапы модели процессов MSF.....	13
Этап создания общей картины приложения.....	13
Этап планирования.....	15
Этап разработки.....	17
Этап стабилизации.....	18
Этап развертывания.....	20
Занятие 3. Исходные данные для разработки проекта для Adventure Works Cycles.....	22
Пример — компания Adventure Works Cycles.....	22
Бизнес-задачи.....	22
Требования к системе для компании Adventure Works Cycles.....	27
Резюме.....	29
Закрепление материала.....	30
Глава 2. Сбор и анализ информации	31
Занятие 1. Сбор информации.....	32
Категории информации.....	32
Методы сбора информации.....	34
Источники информации.....	39
Определение стратегии сбора информации.....	40
Занятие 2. Анализ информации.....	42
Информация об архитектуре предприятия.....	42
Высокоуровневые варианты и сценарии использования системы.....	43
Предварительный документ о требованиях.....	43
Внутренняя документация проектной команды.....	47
Занятие 3. Использование нотаций моделирования.....	49
Преимущества моделирования.....	49
Что такое UML.....	50
Что такое ORM.....	52
Занятие 4. Создание ВИС и СИС.....	60
Как создается ВИС.....	60
Что такое ВИС.....	63
Почему следует создавать СИС для текущего состояния.....	64
Как уточняются требования.....	66
Практикум. Сбор и анализ информации.....	68
Упражнение 1. Подготовка к интервью.....	68

Упражнение 2. Создание вариантов использования системы для проекта автоматизации продаж и расширения Web-проекта.....	68
Упражнение 3. Разработка предварительных требований на основе первичного сбора информации.....	69
Упражнение 4. Разработка сценария использования системы.....	69
Резюме.....	69
Закрепление материала.....	70
Глава 3. Создание общей картины решения.....	71
Занятие 1. Стадия создания общей картины решения.....	72
Цель создания общей картины решения.....	72
Роли и обязанности членов команды.....	74
Организация проектных команд.....	75
Результаты этапа создания общей картины решения.....	75
Занятие 2. Создание документа общей картины и области действия решения ...	77
Документ общей картины и области действия решения.....	77
Создание формулировки задач.....	78
Создание документа общей картины решения.....	78
Определение области действия.....	80
Создание концепции решения.....	84
Определение целей проекта.....	85
Проверка документа общей картины и области действия решения.....	86
Занятие 3. Создание документа структуры проекта.....	88
Документ структуры проекта.....	88
Роли и обязанности заказчика и проектной команды.....	89
Решения по связям.....	90
Логистические решения.....	91
Решения по управлению изменениями.....	92
Решения относительно оценки хода проекта.....	92
Занятие 4. Анализ рисков.....	93
Процесс управления риском.....	93
Содержание документа оценки рисков.....	94
Создание документа оценки риска.....	94
Практикум. Разработка документа общей картины и области действия решения.....	96
Исходные условия.....	96
Упражнение 1. Формулировка задачи.....	98
Упражнение 2. Создание общей картины решения.....	98
Упражнение 3. Разработка целей проекта.....	99
Резюме.....	100
Закрепление материала.....	102
Глава 4. Создание концептуального дизайна.....	103
Занятие 1. Основные сведения об этапе планирования.....	105
Этап планирования.....	105
Три типа дизайна; концептуальный, логический и физический.....	107
Роли и обязанности членов команды на этапе планирования.....	108
Контрольные точки и результаты этапа планирования.....	109

Занятие 2. Основные сведения о функциональных спецификациях	111
Что такое функциональные спецификации	111
Цели функциональных спецификаций	112
Риски, возникающие в отсутствие функциональных спецификаций	113
Элементы функциональных спецификаций	114
Занятие 3. Основные сведения о концептуальном дизайне	117
Что такое концептуальный дизайн	117
Цели концептуального дизайна	118
Стадии концептуального дизайна	119
Занятие 4. Разработка концептуального дизайна	121
Стадия анализа в концептуальном дизайне	121
Обновление формулировки требований	122
Классификация требований	123
Уточнение диаграмм вариантов использования системы	126
Архитектура приложения	130
Пример концептуальной модели	132
Занятие 5. Оптимизация концептуального дизайна	134
Оптимизация процессов	134
Оценка реорганизованных процессов	137
Проверка модели концептуального дизайна	137
Практикум. Анализ требований	139
Упражнение 1. Уточнение вариантов использования системы и требований ..	139
Упражнение 2. Просмотр диаграммы концептуальной модели	140
Резюме	141
Закрепление материала	143
Глава 5. Создание логического дизайна	145
Занятие 1. Основные сведения о логическом дизайне	147
Что такое логический дизайн	147
Преимущества логического дизайна	151
Роли команды в процессе логического дизайна	151
Занятие 2. Разработка логического дизайна	153
Уточнение возможных технологий на этапе логического дизайна	153
Определение предварительных бизнес-объектов	157
Что такое сервисы	158
Как определяются атрибуты	160
Как определяются отношения	162
Занятие 3. Документирование результатов логического дизайна	166
Моделирование отношений	166
Как создается логическая модель объектов	168
Как создается логическая модель данных	170
Создание высокоуровневого дизайна пользовательского интерфейса	171
Занятие 4. Оптимизация логического дизайна	173
Уточнение множества объектов	173
Проверка существующей логической модели объектов	173
Выбор модели управления в логическом дизайне	175

Практикум. Определение объектов в логическом дизайне.....	178
Упражнение 1. Определение объектов на основании ВИС.....	178
Упражнение 2. Создание матрицы сервисов.....	178
Упражнение 3. Создание диаграммы последовательности.....	178
Резюме.....	179
Закрепление материала.....	180
Глава 6. Создание физического дизайна.....	181
Занятие 1. Основные сведения о физическом дизайне.....	182
Физический дизайн.....	182
Цели физического дизайна.....	185
Роли и обязанности членов команды в процессе физического дизайна.....	185
Результаты физического дизайна.....	186
Стадии физического дизайна.....	186
Стадия исследования.....	188
Занятие 2. Стадия анализа.....	190
Совершенствование UML-моделей.....	190
Создание предварительной модели развертывания.....	194
Занятие 3. Стадия рационализации.....	196
Результаты стадии рационализации.....	196
Создание стратегии распределения и объединения в наборы.....	196
Связность и сочленение.....	198
Объединение компонентов в наборы.....	199
Распределение предварительных компонентов.....	199
Создание модели развертывания.....	201
Проверка и уточнение распределения и объединения в наборы.....	202
Занятие 4. Реализация физического дизайна.....	204
Что такое модель программирования.....	204
Определение интерфейсов компонента.....	206
Физический дизайн модели пользовательского интерфейса.....	207
Физический дизайн модели базы данных.....	208
Практикум. Создание физического дизайна.....	209
Упражнение 1. Создание модели классов.....	209
Упражнение 2. Создание диаграммы модели компонентов.....	209
Резюме.....	210
Закрепление материала.....	212
Глава 7. Проектирование презентационного уровня.....	213
Занятие 1. Основы дизайна пользовательского интерфейса.....	215
Презентационный уровень.....	215
Компоненты пользовательского интерфейса.....	216
Функции компонентов пользовательского интерфейса.....	217
Особенности успешного дизайна интерфейса.....	218
Занятие 2. Проектирование пользовательского интерфейса.....	221
Создание начального проекта пользовательского интерфейса.....	221
Система помощи пользователям.....	222
Выбор модели пользовательского интерфейса.....	224

Выбор клиентской среды	226
Создание прототипа пользовательского интерфейса	228
Проверка дизайна интерфейса на соответствие требованиям пользователя	231
Результаты процесса создания дизайна пользовательского интерфейса	231
Занятие 3. Проектирование компонентов пользовательского процесса	233
Функции компонентов пользовательского процесса	233
Разделение пользовательского процесса и интерфейса	234
Принципы проектирования пользовательских процессов	235
Практикум. Создание пользовательского интерфейса	236
Резюме	237
Закрепление материала	238
Глава 8. Проектирование уровня данных	239
Занятие 1. Проектирование хранилища данных	240
Схема базы данных	240
Определение сущностей и атрибутов	244
Определение таблиц и столбцов	245
Организация связей	248
Занятие 2. Оптимизация доступа к данным	251
Методы оптимизации доступа к данным	251
Индексирование данных	252
Разбиение данных на разделы	253
Нормализация данных	254
Преимущества нормализации	254
Занятие 3. Проверка данных	259
Целостность данных	259
Определение требований к целостности данных	261
Определение бизнес-правил	261
Реализация бизнес-правил в базе данных	262
Реализация проверки данных в компонентах	264
Практикум. Создание схемы данных	265
Упражнение. Создание схемы данных	265
Резюме	266
Закрепление материала	268
Глава 9. Дизайн спецификаций безопасности	269
Занятие 1. Основные сведения о безопасности применительно	
к разработке приложений	270
Типичные бреши в системе безопасности	270
Недостатки традиционных моделей безопасности	271
Принципы создания стратегии защиты	271
Занятие 2. Планирование защиты приложения	273
Роль безопасности в процессе разработки приложений	273
Модель опасностей STRIDE	274
Создание модели опасностей	275
Использование модели опасностей	277
Методы противодействия опасностям	277

Занятие 3. Использование функций безопасности .NET Framework	280
Контроль типов	280
Подписание кода	281
Шифрование и подписание данных	281
Безопасность доступа из кода	282
Безопасность, основанная на ролях	282
Изолированное хранилище	283
Функции безопасности в технологиях .NET	284
Занятие 4. Проектирование механизмов авторизации, аутентификации и аудита	289
Разработка стратегий авторизации и аутентификации	289
Разработка стратегии авторизации в компонентах пользовательского интерфейса	292
Разработка стратегии авторизации в бизнес-компонентах	293
Разработка стратегии авторизации в компонентах доступа к данным	293
Разработка стратегии аутентификации в компонентах пользовательского интерфейса	293
Разработка стратегий аутентификации в компонентах доступа к данным	295
Разработка стратегии аудита	296
Практикум. Моделирование опасностей и меры по их предотвращению	298
Упражнение 1. Определение потенциальных опасностей	298
Упражнение 2. Применение технологий для предотвращения опасности	298
Резюме	298
Закрепление материала	300
Глава 10. Завершение этапа планирования	301
Занятие 1. Реализация требований к проекту	302
Как спроектировать масштабируемое приложение	302
Создание приложения высокой доступности	306
Создание надежного приложения	308
Проектирование производительности	309
Разработка приложения, поддерживающего взаимодействие с другими системами	311
Реализация в проекте требований по поддержке различных языков и локализации	312
Занятие 2. Планирование функций администрирования	316
Планирование мониторинга	316
Планирование переноса данных	317
Создание спецификаций по лицензированию	318
Занятие 3. Планирование будущих этапов	319
Планирование этапа разработки	319
Планирование этапа стабилизации	320
Планирование этапа развертывания	322
Занятие 4. Создание технической спецификации	324
Что такое техническая спецификация	324
Разделы технической спецификации	324

Практикум. Анализ плана тестирования и технической спецификации.....	325
Упражнение 1, Анализ плана тестирования.....	325
Упражнение 2. Анализ технической спецификации.....	325
Резюме.....	326
Закрепление материала.....	328
Глава 11. Стабилизация и развертывание решения.....	329
Занятие 1. Этап стабилизации в MSF.....	330
Результаты этапа стабилизации.....	330
Промежуточные контрольные точки этапа стабилизации.....	331
Задачи ролей команды на этапе стабилизации.....	333
Занятие 2. Тестирование и пилотная эксплуатация.....	334
Рекомендации по тестированию.....	334
Типы тестирования.....	335
Применяемые в тестировании термины.....	335
Классификация и контроль ошибок.....	337
Задачи тестирования.....	340
Пилотная эксплуатация.....	341
Занятие 3. Этап развертывания в MSF.....	346
Контрольные точки и результаты этапа развертывания в MSF.....	346
Основные задачи команды на этапе развертывания.....	346
Сценарии развертывания.....	347
Занятие 4. Развертывание в промышленной среде.....	348
Планирование развертывания.....	348
Базовые компоненты и компоненты, характерные для отдельных мест установки.....	349
Развертывание базовых компонентов.....	350
Развертывание компонентов для индивидуальных мест установки.....	351
Период затишья.....	353
Передача проекта команде сопровождения и поддержки.....	353
Закрытие проекта.....	354
Практикум. Определение важности ошибок.....	355
Резюме.....	357
Закрепление материала.....	358
Приложение. Вопросы и ответы.....	359
Предметный указатель.....	379

Об этой книге

Мы рады представить вам учебный курс MCSE «Анализ требований и создание архитектуры решений на основе Microsoft .NET». Он позволит вам подготовиться к сдаче сертификационного экзамена 70-300; *Analyzing Requirements and Defining Microsoft .NET Solution Architectures*.

Примечание Подробнее о программе сертификации специалистов Microsoft Certified Professional — в разделе «Программа сертификации специалистов Microsoft» далее в этой главе.

Главы состоят из занятий и практикумов. Каждая глава заканчивается разделом «Закрепление материала». Моделируя предлагаемые в этом разделе ситуации, вы сможете оценить степень усвоения материала главы. В занятиях подробно рассказано, как принимать решения в процессе работы над проектом; после большинства занятий предлагаются практикумы, выполняя которые вы сможете приобрести навыки работы.

Книга содержит множество примеров, которые облегчают восприятие и усвоение материала, а также иллюстрируют принятие решения в каждом конкретном случае.

В упражнениях подобраны примеры по темам занятий главы. Они наглядно демонстрируют, как от решений, принимаемых в процессе разработки проекта, зависит эффективность завершеного решения.

Наконец, в разделе «С чего начать» перечислено оборудование и программное обеспечение, необходимое для работы над материалом книги, а также сведения о сетевых конфигурациях, которые требуются для выполнения некоторых практикумов. Внимательно ознакомьтесь с этим разделом до начала занятий.

Кому адресована эта книга

Данный курс предназначен:

- специалистам в области информационных технологий, планирующим проектировать, разрабатывать и внедрять ИТ-решения в Windows-среде с использованием инструментов и технологий Microsoft;
- тем, кто планирует сдать сертификационный экзамен 70-300.

Требования к читателям

Для успешного изучения материала и понимания концепций и задач, изложенных в учебном курсе, требуется предварительная подготовка. Как минимум, необходимо:

- **общее** понимание жизненного цикла разработки программного обеспечения;
- практические знания и опыт разработки в среде Microsoft .NET;
- знание модели процессов MSF;
- базовые знания в области методологий объектного моделирования и моделирования данных;
- опыт работы с Microsoft Visio Professional 2000;
- не менее чем годичный опыт работы в команде, занимающейся разработкой ПО.

Справочные материалы

- «Белая книга» о модели процессов MSF на Web-сайте <http://www.microsoft.com/msf/>.
- Описание методов UML (Unified Modeling Language) в следующих изданиях:
 - книга Грейди Буча (Grady Booch), Айвара Джекобсона (Ivar Jacobson) и Джеймса Рамбо (James Rumbaugh) «The Unified Modeling Language User Guide». Addison-Wesley, 1999;
 - книга Дуга Розенберга (Doug Rosenberg) в соавторстве со Скоттом Кендаллом (Scott Kendall), «Use Case Driven Object Modeling with UML: A Practical Approach». Addison-Wesley, 1999.

Компакт-диск с дополнительными материалами

к курсу

К книге прилагается компакт-диск, на котором записаны:

- полная электронная версия книги с поддержкой контекстного поиска. Подробнее о работе с электронной версией книги — в разделе «Электронный вариант книги»;
- файлы, необходимые при работе над материалом книги, а также примеры проектных документов;
- электронная версия словаря терминов, используемых в настоящем курсе;
- практический тест, состоящий из 100 вопросов. Он поможет выяснить готовность к экзамену, а также лучше разобраться в материале книги.

Структура книги

Каждая глава начинается с раздела «В этой главе», содержащего краткий обзор обсуждаемых тем. Главы состоят из занятий и практикумов. Каждую главу завершает раздел «Закрепление материала», вопросы которого помогут вам проверить, насколько твердо вы усвоили материал. В приложении А «Вопросы и ответы» собраны вопросы всех глав книги и ответы для самопроверки.

Примечания

Практически во всех главах встречаются примечания разных видов.

- **Совет** — поясняет возможный результат или описывает альтернативный метод решения задачи.
- **Внимание!** — предупреждает вас о возможной потере данных или содержит сведения, необходимые для выполнения поставленной задачи.
- **Примечание** — содержит дополнительную информацию.
- **Совет по планированию** — здесь приведена полезная информация о планировании развертывания в конкретной среде.

Обозначения

- *Курсивом* выделяются важные термины, положения и другие **существенные** сведения, после которых в скобках, как правило, указан их английский эквивалент.
- *Курсивом* также выделяются имена файлов, папок и каталогов, расширения файлов.
- Аббревиатуры обозначаются **ЗАГЛАВНЫМИ БУКВАМИ**.
- Примеры кода, текста, отображаемого на экране и вводимого в командной строке, набраны моноширинным шрифтом.
Значками на полях помечены конкретные разделы.

Значок	Описание
	Практикум, выполняя задание которого вы закрепите навыки, приобретенные при изучении материала
	Вопросы, отвечая на которые вы проверите, насколько твердо и безошибочно усвоили изложенный материал. Вопросы обычно сгруппированы в конце главы в разделе «Закрепление материала». Ответы для самопроверки находятся в приложении А «Вопросы и ответы»



Практикум, выполняя задание которого вы **закрепите** навыки, приобретенные при изучении материала



Вопросы, отвечая на которые вы проверите, насколько твердо и безошибочно усвоили изложенный материал. Вопросы обычно сгруппированы в конце главы в разделе «Закрепление материала». Ответы для самопроверки находятся в **приложении А «Вопросы и ответы»**

Обзор глав и приложений

Материал этого учебного курса посвящен основам разработки архитектуры решений в среде Microsoft .NET. Конечно, курс предназначен для последовательного изучения, однако вы можете работать с ним так, как вам удобнее, скажем, проработать лишь отдельные главы. Советуем в этом случае обращать внимание на раздел «Прежде всего» в начале каждой главы, где указаны предварительные требования для выполнения упражнений.

- Во вводной главе собраны сведения о содержании книги и данные о структурных единицах и условных обозначениях, принятых в ней. Внимательно прочитайте эту главу — это поможет вам эффективнее работать с материалами курса, а также выбрать интересующие вас темы. Здесь также подробно описана установка программ и файлов, необходимых для успешного выполнения упражнений данного курса.
- В главе 1 «Введение в проектирование бизнес-решений» описывается каркас Microsoft Solutions Framework (MSF). Глава начинается с краткого рассказа о модели процессов MSF и ее этапах, об основных операциях, выполняемых в процессе проектирования приложения, и их результатах. Кроме того, в этой главе указаны исходные данные проекта, который используется в качестве примера на протяжении всей книги. Именно для него создаются все методики и проектные документы.
- В главе 2 «Сбор и анализ информации» речь идет о сборе и анализе информации в процессе проектирования бизнес-решения. Глава начинается описанием типов данных, которые необходимо собрать, источников информации и некоторых методов (таких, как интервьюирование, наблюдение за действиями пользователя и создание прототипов), которые обычно применяются для сбора информации. Далее рассказано, как анализировать собранную информацию, а также перечислены методы анализа информации — варианты и сценарии использования системы.
- Глава 3 «Создание общей картины решения» посвящена этапу создания общей картины решения в процессе разработки проектов в MSF. Вначале описаны цели этого этапа, затем — роли и обязанности членов команды. Далее обсуждаются основные задачи этапа, создание единого видения проекта и анализ связанных с ним рисков.
- В главе 4 «Создание концептуального дизайна» вы познакомитесь с процессом концептуального дизайна этапа планирования. Сначала обсуждаются цели этапа и три процесса дизайна, из которых состоит этап планирования: концептуальный, логический и физический, а также рассказывается о функциональной спецификации. Затем подробно описывается концептуальный дизайн и три его стадии: исследование, анализ и оптимизация.
- Глава 5 «Создание логического дизайна» посвящена процессу логического дизайна, выполняемому на этапе планирования. Она начинается кратким описанием целей и преимуществ логического дизайна, а также рассказом о составе проектной команды и роли каждого из ее членов на этой стадии. Затем подробно описывается порядок создания логического дизайна бизнес-решения, инструменты и методы документирования результатов логического дизайна, а также рекомендации по оптимизации логического дизайна, в частности проверка на предмет соответствия требованиям представляется как наиболее результативный метод оптимизации.

- В главе 6 «Создание физического дизайна» рассказывается о стадии физического дизайна на этапе планирования. Вначале обсуждаются цели физического дизайна, затем его результаты и задачи, выполняемые при завершении физического дизайна: исследование, анализ, рационализация и реализация.
- Глава 7 «Проектирование презентационного уровня» посвящена процессу проектирования презентационного уровня приложения. Здесь кратко описан презентационный уровень и два его компонента: пользовательский интерфейс и процесс. Далее вы узнаете, как создается дизайн пользовательского интерфейса и компоненты пользовательского процесса приложения. Завершается глава рекомендациями по проектированию и дизайну презентационного уровня.
- Глава 8 «Проектирование уровня данных» посвящена проектированию уровня данных приложения. Здесь также обсуждается, как оптимизировать доступ к данным и реализовать проверку данных в приложении.
- В главе 9 «Дизайн спецификаций безопасности» рассказано об основных принципах создания спецификаций безопасности приложения, предлагаются инструменты и методы оценки и предотвращения опасностей, грозящих приложению. В главе также описываются некоторые возможности обеспечения безопасности в Microsoft .NET, а также мероприятия и рекомендации по проектированию аутентификации, авторизации и аудита в приложении.
- Из главы 10 «Завершение этапа планирования» вы узнаете о задачах, над которыми работает проектная команда при завершении этапа планирования. Здесь описаны принципы и рекомендуемые методы проектирования для обеспечения масштабируемости, доступности, надежности, высокой производительности, способности к взаимодействию, поддержки других языков и локализации, а также обсуждаются вопросы планирования административных функций, таких, как мониторинг, перенос данных и управление лицензиями. Кроме того, из главы вы узнаете о планах: разработки, тестирования, пилотной эксплуатации, развертывания и переноса данных из унаследованных систем — а также о цели и содержании документа технической спецификации.
- В главе 11 «Стабилизация и развертывание решения» описаны задачи, выполняемые в процессе стабилизации и развертывания решения. Глава начинается обсуждением этапа стабилизации, а также различных типов тестирования, применяемых для стабилизации продукта, затем речь пойдет о порядке пилотной эксплуатации и процессе развертывания. Вы узнаете, как планировать развертывание, и познакомитесь с различными стратегиями, которые для этого применяются.
- В приложении «Вопросы и ответы» приведены ответы на вопросы из упражнений и разделов «Закрепление материала» всех глав учебного курса,

Материалы для подготовки к экзаменам

Далее перечислены темы сертификационного экзамена 70-300: *Analyzing Requirements and Defining Microsoft .NET Solution Architectures* и главы настоящего учебного курса, где обсуждаются соответствующие вопросы.

Примечание Конкретная программа любого экзамена определяется Microsoft и может быть изменена без предварительного уведомления.

Тема	Где обсуждается	
	Глава	Занятие
Создание общей картины решения		
Разработка концепции решения	3	2
Анализ реализуемости решения:	3	2
<ul style="list-style-type: none"> • анализ реализуемости решения с точки зрения бизнеса; • анализ реализуемости решения с технической точки зрения; • анализ квалификации и доступности ресурсов 		
Анализ и уточнение области действия проекта	3	2
Определение основных рисков, связанных с проектом	3	4
Сбор и анализ бизнес-требований		
Сбор и анализ бизнес-требований:	2 4	1–2 и 4
<ul style="list-style-type: none"> • анализ текущего состояния предприятия; • анализ бизнес-требований решения 		
Сбор и анализ пользовательских требований:	2 4	1–2 и 4
<ul style="list-style-type: none"> • определение вариантов использования системы; • определение требований по поддержке других языков; • определение требований по локализации; • определение требований по поддержке специальных возможностей 		
Сбор и анализ системных требований:	2 4	1–2 и 4
<ul style="list-style-type: none"> • определение требований по сопровождению и поддержке; • определение требований по масштабированию; • определение требований по доступности; • определение требований по надежности; • определение требований по развертыванию; • определение требований по безопасности 		
Сбор и анализ требований к оборудованию, ПО и сетевой инфраструктуре:	2 4	1–2 и 4
<ul style="list-style-type: none"> • определение требований по интеграции; • анализ ИТ-среды, в том числе существующих и будущих приложений, а также текущего и планируемого оборудования, системного ПО и сетевой инфраструктуры; • анализ влияния решения на ИТ-среду 		

(продолжение)

Тема	Где обсуждается	
	Глава	Занятие
Разработка спецификаций		
Создание функциональных спецификаций на основе требований. Учет таких параметров проекта, как производительность, возможность поддержки и сопровождения, расширяемость, масштабируемость, доступность, удобство развертывания и безопасность	4 6 10	Все занятия 3 и 1
Создание технических спецификаций на основе функциональных требований. Учет таких параметров проекта, как производительность, возможность поддержки и сопровождения, расширяемость, масштабируемость, доступность, удобство развертывания и безопасность:	10	4
<ul style="list-style-type: none"> • выбор стратегии разработки; • выбор стратегии развертывания; • выбор стратегии обеспечения безопасности; • выбор стратегии поддержки и сопровождения; • создание плана тестирования; • создание плана обучения пользователей 		
Концептуальный дизайн		
Создание концептуального дизайна бизнес-требований и требований к данным с использованием методов Object Role Modeling (ORM):	4 2	Все занятия и 3
<ul style="list-style-type: none"> • преобразование внешней информации в элементарные факты; • проверка популяции элементарных фактов; • определение примитивных типов объектов в концептуальной модели; • применение ограничений уникальности к концептуальной модели; • применение к концептуальной модели ограничений обязательной принадлежности к роли; • добавление к концептуальной модели ограничений на значения, на сравнение множеств и на подтипы; • добавление к концептуальной модели круговых ограничений 		
Логический дизайн		
Создание концептуального дизайна решения:	5	Все занятия
<ul style="list-style-type: none"> • обеспечение аудита и контроля; • обеспечение обработки ошибок; • обеспечение интеграции; • обеспечение поддержки других языков; • обеспечение локализации; 		

(см. следующую страницу)

(продолжение)

Тема	Где обсуждается	
	Глава	Занятие
<ul style="list-style-type: none"> • обеспечение безопасности; • реализация ограничений для поддержки бизнес-правил; • логический дизайн презентационного уровня, в том числе пользовательского интерфейса; • логический дизайн сервисов и компонентов; • логический дизайн управления состоянием; • логический дизайн архитектур синхронного и асинхронного управления 		
Создание логической модели данных:	5	3
<ul style="list-style-type: none"> • определение таблиц и столбцов; • нормализация таблиц; • определение отношений; • определение основных и внешних ключей; • определение XML-схемы 		
Проверка предлагаемого логического дизайна:	5	4
<ul style="list-style-type: none"> • анализ предлагаемого логического дизайна на предмет соответствия таким бизнес-требованиям, как производительность, возможность поддержки, расширяемость, масштабируемость, доступность, простота развертывания и безопасность; • проверка предлагаемого логического дизайна на предмет соответствия сценариям использования системы; • создание варианта продукта для проверки верности концепции 		
Физический дизайн		
Выбор технологий для физического дизайна решения	5	2;
Создание физического дизайна решения:	6	Все занятия,
	7	Все занятия и
	9	3
<ul style="list-style-type: none"> • создание спецификаций аудита и журналирования; • создание спецификаций обработки ошибок; • создание спецификаций физической интеграции; • создание спецификаций безопасности; • реализация ограничений для поддержки бизнес-правил; • дизайн презентационного уровня, в том числе пользовательского интерфейса и онлайн-системы поддержки пользователей; • дизайн сервисов и компонентов; • дизайн управления состоянием 		

(окончание)

Тема	Где обсуждается	
	Глава	Занятие
Создание физического дизайна развертывания:	10	2
<ul style="list-style-type: none"> • создание спецификаций развертывания, в том числе сосуществование с унаследованными системами и распределение; • создание спецификаций по лицензированию; • создание спецификаций по переносу данных; • определение пути обновления 		
Создание физического дизайна поддержки:	10	2
<ul style="list-style-type: none"> • создание механизма мониторинга приложения 		
Создание физического дизайна модели данных:	8	2
<ul style="list-style-type: none"> • создание спецификации индексирования; • деление данных на разделы; • денормализация таблиц 		
Проверка физического дизайна:	6	3—4
<ul style="list-style-type: none"> • анализ соответствия предложенного физического дизайна бизнес-требованиям; • проверка вариантов исценариев использования системы, сценариев прогона и диаграмм последовательностей; • создание варианта продукта для проверки верности концепции 		
Определение стандартов и процессов		
Определение стандартов, в том числе стандартов разработки документации, исходного кода, анализа кода, пользовательского интерфейса и тестирования	11	2
Определение процессов, в том числе процессов проверки документации разработчика, кода, создания сборок, отслеживания ошибок, управления исходным кодом, управления изменениями и версиями и задач по сопровождению. Методы можно создавать на основе шаблонов Microsoft Visual Studio .NET Enterprise	11	2
Определение метрик качества и производительности, применяемых для оценки управления проектом, организационной производительности и окупаемости	11	1—2

Начало работы

Этот курс содержит практические занятия, выполняя которые вы получите навыки создания архитектуры решений в среде Microsoft .NET.

Требования к аппаратному обеспечению

Все компьютеры должны удовлетворять указанным далее требованиям к оборудованию, кроме того, все аппаратное обеспечение должно быть указано в списке совместимого оборудования Hardware Compatibility List (HAL) для Microsoft Windows 2000 Professional (требование совместимости оборудования);

- процессор Pentium II с частотой 266 МГц или выше;
- 128 Мбайт оперативной памяти;
- жесткий диск размером не менее 4 Гбайт;
- дисковод для компакт-дисков;
- Microsoft Mouse или другое совместимое координатно-указательное устройство.

Программное обеспечение

Для выполнения упражнений и практикумов данного курса необходимо следующее ПО;

- Microsoft Windows 2000 Professional с Service Pack 3 или Microsoft Windows XP Professional;
- Microsoft Office 2000 Professional с пакетом исправлений Service Pack 3 или более поздним;
- Microsoft Visio 2000 Professional или более поздняя версия.

Подготовка компьютера к выполнению практических заданий

Настройте компьютер в соответствии с рекомендациями производителей соответствующего оборудования и ПО.

Установка примеров и вспомогательных материалов

На прилагаемом к книге компакт-диске хранятся файлы, необходимые для выполнения заданий курса. Перед началом работы следует скопировать эти файлы на жесткий диск.

► Установка примеров и вспомогательных материалов

1. Вставьте компакт-диск, прилагаемый к книге, в привод CD-ROM.

Примечание Если на компьютере отключен автоматический запуск компакт-дисков, найдите в корневом каталоге компакт-диска и запустите на исполнение файл StartCD.exe или ознакомьтесь с содержанием справочного файла Readme.txt.

2. Выберите в меню пользовательского интерфейса команду Solution Documents и перейдите в папку с материалами соответствующей главы
- ИЛИ
3. скопируйте вспомогательные материалы на жесткий диск своего компьютера и обращайтесь к ним по мере изучения курса.

Электронный вариант книги

На компакт-диске, прилагаемом к книге, записан электронный вариант оригинального издания этого учебного курса, который доступен для просмотра при наличии Microsoft Internet Explorer 5 или более поздней версии. Если ваша система не удовлетворяет этим требованиям, перед установкой электронного варианта книги вы можете установить Internet Explorer 6 SP1 с прилагаемого компакт-диска.

► **Установка электронной версии книги**

1. Вставьте компакт-диск, прилагаемый к книге, в привод CD-ROM.

Примечание Если на компьютере отключен автоматический запуск компакт-дисков, ознакомьтесь с содержанием справочного файла `Readme.txt`.

2. Выберите в меню пользовательского интерфейса команду eBook и следуйте рекомендациям мастера.

Программа сертификации специалистов Microsoft

Программа сертификации специалистов Microsoft (Microsoft Certified Professional, MCP) — отличная возможность подтвердить ваши знания современных технологий и программных продуктов этой фирмы. Лидер отрасли в области сертификации, Microsoft разработала современные методы тестирования. Экзамены и программы сертификации подтвердят вашу квалификацию разработчика или специалиста по реализации решений на основе технологий и программных продуктов Microsoft. **Сертифицированные Microsoft профессионалы квалифицируются** как эксперты и высоко ценятся на рынке труда.

Программа сертификации специалистов предлагает несколько типов сертификации по разным специальностям.

- **Сертифицированный специалист Microsoft (Microsoft Certified Professional, MCP)** — предполагает глубокое и доскональное знание по крайней мере одной операционной системы Microsoft. MCP обладает квалификацией, необходимой для развертывания продуктов и технологий Microsoft, входящих в состав **бизнес-решения** для предприятия.
- **Сертифицированный разработчик программных решений на основе продуктов Microsoft (Microsoft Certified Solution Developer, MCS D)** — отвечает за анализ, разработку и создание прикладных приложений с применением инструментальных средств, технологий и платформ Microsoft, в том числе Microsoft .NET Framework.
- **Сертифицированный разработчик программного обеспечения на основе продуктов (Microsoft Certified Application Developer, MCAD)** — профессиональный разработчик, отвечающий за разработку, тестирование, развертывание и поддержку сложных приложений с применением инструментальных средств, технологий и платформ Microsoft, в том числе Visual Studio .NET и XML Web-сервисов.
- **Сертифицированный системный инженер Microsoft (Microsoft Certified Systems Engineer)** — предполагает умение анализировать бизнес-требования, проектировать и реализовывать **инфраструктуру бизнес-решений** на базе Microsoft Windows и операционной системы Microsoft Server 2003.
- **Сертифицированный системный администратор Microsoft (Microsoft Certified Systems Administrator, MCSA)** — предполагает умение управлять и устранять неполадки **действующих** сетевых систем на базе Microsoft Windows и операционной системы Microsoft Server 2003.
- **Сертифицированный администратор баз данных Microsoft (Microsoft Certified Database Administrator, MCDBA)** — должен уметь разрабатывать **физическую** структуру и логические модели данных, создавать физические базы данных, службы доступа к данным с использованием Transact-SQL, управлять и поддерживать базы данных, настраивать и управлять системой защиты, выполнять мониторинг и оптимизацию баз данных, а также установку и настройку Microsoft SQL Server.
- **Сертифицированный преподаватель Microsoft (Microsoft Certified Trainer, MCT)** — предполагает теоретическую и практическую подготовку для ведения соответствующих курсов в авторизованных учебных центрах Microsoft (Certified Technical Education Center, CTEC).

Преимущества программы сертификации Microsoft

Программа сертификации Microsoft — один из самых строгих и полных тестов оценки знаний и навыков в области проектирования, разработки и сопровождения программного обеспечения. Звание **сертифицированного специалиста Microsoft** присваивается лишь тому, кто демонстрирует умение решать конкретные задачи, применяя продукты компании. Программа тестирования позволяет не только оценить квалификацию специалиста, но и служит **ориентиром** для всех, кто стремится достичь современного уровня знаний в этой области. Как и любой другой тест или экзамен, сертификация Microsoft служит критерием определенного уровня знаний специалиста, что важно как при трудоустройстве, так и для карьерного роста в организации.

Для специалистов. Звание Microsoft Certified Professional дает следующие преимущества:

- **официальное признание** знаний и опыта работы с продуктами и технологиями Microsoft;
- **подписку на MSDN.** MCP получает скидку на годовую подписку на Microsoft Developer Network (msdn.microsoft.com/subscriptions/ в первый год после сертификации. (Конкретные условия зависят от региона. Подробнее — в Welcome Kit.);
- **доступ к технической информации о продуктах Microsoft в защищенной области Web-узла MCP.** Для этого следует открыть страницу по адресу <http://www.microsoft.com/trainingandservices/> развернуть узел Certification и щелкнуть ссылку For MCPs Only;
- **эксклюзивные скидки на продукты и услуги различных компаний.** Подробнее об этом — на Web-странице по адресу <http://www.microsoft.com/trainingandservices/>. Разверните узел Certification, щелкните ссылку For MCPs Only и далее — ссылку Other Benefits;
- **эмблемы, сертификат, отчет об экзамене, карточка для бумажника и закладка на лацкан пиджака, свидетельствующие о наличии звания Microsoft Certified Professional.** После того как вы успешно выдержите экзамен, вы получите право загрузить электронные копии эмблемы и отчета об экзамене с защищенной области Web-узла MCP. Для этого следует открыть страницу по адресу <http://www.microsoft.com/trainingandservices/> развернуть узел Certification и щелкнуть ссылку For MCPs Only;
- **приглашения** на конференции, семинары и мероприятия, предназначенные для специалистов Microsoft;
- **бесплатную подписку на MSDN Online Certified Membership.** Вы получите доступ к лучшим техническим ресурсам, новостям сообщества MCP и другим полезным ресурсам и службам (отдельные элементы узла MSDN Online доступны лишь на английском языке или в некоторых странах — недоступны вообще). Для получения списка услуг, доступных сертифицированным специалистам, обратитесь на Web-узел MSDN;
- **скидку на членство в PASS (Professional Association for SQL Server),** сообществе, поддерживаемом Microsoft и предлагающем своим членам массу образовательных материалов (подробнее — на Web-странице <http://www.microsoft.com/traincert/mcp/mcpsecure.asp>).

Кроме того, специалисты Microsoft Certified System Engineer получают дополнительное преимущество:

- **50% скидку на годовую подписку на компакт-диски TechNet или TechNet Plus в течение года после сдачи экзамена.** (Конкретные условия зависят от региона. Подробнее — в Welcome Kit.) Кроме того, бесплатный доступ к Web-узлу TechNet (<http://www.microsoft.com/technet/>) на котором опубликовано 95% материалов с указанных компакт дисков.

Специалисты Microsoft Certified System Database Administrators получают дополнительные преимущества:

- **50% скидку на годовую подписку на компакт-диски TechNet или TechNet Plus в течение года после сдачи экзамена.** (Конкретные условия зависят от региона. Подробнее — в Welcome Kit.) Кроме того, бесплатный доступ к Web-узлу TechNet (<http://www.microsoft.com/technet/>) на котором опубликовано 95% материалов с указанных компакт дисков.
- **годовую подписку на SQL Server Magazine.** Создаваемый экспертами в своей области, это журнал является источником знаний, полезных советов и информации для тех, чья работа неразрывно связана с SQL Server.

Для работодателей и организаций. Сертифицированные специалисты эффективно работают с новейшими технологиями Microsoft, а значит, организация не только быстро окупит затраты на приобретение этих технологий, но и сможет достаточно быстро получить прибыль. Исследования показывают, что сертификация сотрудников по программам Microsoft:

- быстро окупается за счет стандартизации требований к обучению специалистов и методов оценки их квалификации;
- позволяет повысить эффективность обслуживания клиентов и производительность труда, а также снизить расходы на сопровождение операционных систем;
- предоставляет надежные критерии найма специалистов и их продвижения по службе;
- позволяет разработать гибкие методы оценки профессионального уровня и премирования персонала;
- предоставляет возможность переподготовки сотрудников для обучения новым технологиям;
- позволяет оценить профессиональный уровень сторонних фирм — партнеров по бизнесу.

Требования к соискателям

Требования к соискателям определяются специализацией, а также служебными функциями и задачами сотрудника.

Соискатель сертификата Microsoft должен сдать экзамен, подтверждающий его глубокие знания в области программных продуктов Microsoft. Экзаменационные вопросы, подготовленные с участием ведущих специалистов компьютерной отрасли, отражают реалии применения программных продуктов компании Microsoft.

- **Сертифицированный специалист Microsoft** — кандидаты на это звание сдают один сертификационный экзамен Microsoft. Кандидат может сдать дополнительные экзамены, которые подтвердят его право на работу с другими про-

дуктами Microsoft, инструментальными средствами или прикладными программами.

- **Сертифицированный разработчик программных решений на основе продуктов Microsoft** — кандидаты сдают три экзамена по основам технологий и один специальный экзамен.
- **Сертифицированный разработчик программного обеспечения на основе продуктов** — кандидаты сдают два базовых и один специальный экзамен.
- **Сертифицированный системный инженер Microsoft** — кандидаты на это звание сдают пять базовых экзаменов и два специальных экзамена.
- **Сертифицированный системный администратор Microsoft** — кандидаты на это звание сдают три базовых экзамена и один специальный экзамен, на которых должны продемонстрировать теоретические и практические знания технологий Microsoft.
- **Сертифицированный администратор баз данных Microsoft** — кандидаты на это звание сдают три основных экзамена и один — по выбору.
- **Сертифицированный преподаватель Microsoft** — соискателю надо подтвердить свою теоретическую и практическую подготовку для ведения соответствующих курсов в авторизованных учебных центрах Microsoft. Более подробные сведения о сертификации по этой программе вам сообщат сотрудники местного отделения компании Microsoft или на Web-сайте http://www.microsoft.com/tra_in_cert/mct/.

Подготовка к экзаменам

Предлагаются три режима подготовки: самостоятельная работа, интерактивный режим, а также занятия с инструктором в авторизованных центрах подготовки,

Самостоятельная подготовка

Самостоятельная работа — наиболее эффективный метод подготовки для инициативных соискателей. Издательство «Microsoft Press» и «Microsoft Developer Division» предлагают широкий спектр учебных пособий для подготовки к экзаменам по программе сертификации специалистов Microsoft. Учебные курсы для самостоятельного изучения, адресованные специалистам компьютерной отрасли, содержат теоретические и практические материалы, мультимедийные презентации, упражнения и необходимое ПО. Все эти пособия позволяют изучить весь необходимый материал, чтобы успешно сдать любой из сертификационных экзаменов.

Интерактивная подготовка

Интерактивная подготовка средствами Интернета — альтернатива занятиям в учебных центрах. Вы можете выбрать наиболее удобный распорядок занятий в виртуальном классе, где научитесь работать с продуктами и технологиями компании Microsoft и подготовитесь к сдаче экзаменов. В интерактивном виде доступно множество курсов Microsoft — как обычные официальные тесты, так и специальные, разработанные лишь для интерактивного обучения. Интерактивные ресурсы доступны круглосуточно в сертифицированных центрах подготовки.

Сертифицированные центры технического обучения Microsoft

Сертифицированные центры технического обучения Microsoft (Certified Technical Education Center, СТЕС) — самый простой способ изучить материалы курса под руководством опытного инструктора для сдачи экзамена на звание сертифицированного специалиста Microsoft. Всемирная сеть учебных центров Microsoft СТЕС позволяет специалистам повысить технический потенциал под руководством сертифицированных преподавателей Microsoft.

Подробнее о центрах СТЕС в США и Канаде — на Web-узле компании Microsoft по адресу <http://www.microsoft.com/CTEC/default.htm> (на русском языке — <http://www.microsoft.com/rus/CTEC/default.htm>).

Техническая поддержка

Мы постарались сделать все от нас зависящее, чтобы и сам учебный курс, и прилагаемый к нему компакт-диск не содержали ошибок. Если все же у вас возникнут вопросы или вы захотите поделиться своими предложениями или комментариями, обращайтесь в издательство Microsoft Press по одному из указанных ниже адресов.

Электронная почта: TKINPUT@MICROSOFT.COM

Почтовый адрес: Microsoft Press

Attn: *MCSD Self-Paced Training Kit: Analyzing Requirements and Defining Microsoft .NET Solution Architectures, Editor*

One Microsoft Way

Redmond, WA 98052-6399

Издательство Microsoft Press постоянно обновляет список исправлений и дополнений к своим книгам, опубликованный на узле <http://mspress.microsoft.com/support/>.

Учтите, что по указанным почтовым адресам техническая поддержка не предоставляется. Для получения подробной информации о технической поддержке программных продуктов Microsoft обращайтесь на Web-узел компании Microsoft по адресу <http://www.microsoft.com/support/>.

Подробнее о получении полных версий программных продуктов Microsoft вы можете узнать, позвонив в службу Microsoft Sales по телефону (800) 426-9400 или по адресу www.microsoft.com.

ГЛАВА 1

Введение в проектирование бизнес-решений

Занятие 1. Базовые сведения о Microsoft Solutions Framework	2
Занятие 2. Этапы модели процессов MSF	13
Занятие 3. Исходные данные для разработки проекта для Adventure Works Cycles	22
Резюме	29
Закрепление материала	30

В этой главе

Из этой главы вы узнаете о Microsoft Solutions Framework (MSF) — наборе моделей, принципов и рекомендаций по проектированию прикладных программ, а также о модели процессов MSF и ее этапах. Здесь также рассказывается о ключевых операциях, выполняемых при проектировании приложений, и их результатах. В конце главы описан пример, который будет использоваться для иллюстрирования концепций и операций на протяжении всей книги.

Прежде всего

Для изучения материалов этой главы необходимо иметь общее представление о:

- жизненном цикле разработки ПО;
- технологиях Microsoft.

Занятие 1. Базовые сведения о Microsoft Solutions Framework

MSF представляет собой набор **моделей**, принципов и рекомендаций по проектированию и разработке решений масштаба предприятия, который позволит вам успешно управлять такими составляющими проекта, как люди, процессы и инструментальные средства. В MSF также предлагаются проверенные на практике методы планирования, проектирования, разработки и развертывания корпоративных решений. На этом занятии вы узнаете о модели процессов и модели команд в MSF.

Изучив материал этого занятия, вы сможете:

- ✓ описать этапы в модели процессов MSF;
- ✓ описать роли в модели процессов MSF;
- ✓ описать дисциплины MSF: управление риском, готовностью и проектами;
- ✓ рассказать, как, управляя компромиссами, удастся обеспечить успех корпоративного проекта;
- ✓ описать цель итерации в корпоративном проекте;
- ✓ описать водопадную и спиральную модели.

Продолжительность занятия — около 15 минут.

Модели процессов

Microsoft постаралась помочь разработчикам и максимально повысить вероятность успеха корпоративных проектов, предоставив им доступный унифицированный набор рекомендаций по эффективному проектированию, разработке, развертыванию, использованию и поддержке решений, причем не только основанных на технологиях Microsoft. Этот «пакет знаний» базируется на информации, полученной при реализации проектов для Microsoft, при разработке и обслуживании крупномасштабного ПО, полученного от других поставщиков, на опыте консультантов Microsoft, который они приобрели, выполняя проекты для клиентов компании, и на основе других знаний в области информационных технологий.

Модель процессов определяет порядок проектирования и описывает жизненный цикл проекта. Исторически сложились два основных типа модели процессов: водопадная и спиральная.

Водопадная и спиральная модели

На рис. 1-1 показаны контрольные точки каскадов водопадной модели и «винтовая» организация процессов в спиральной модели.

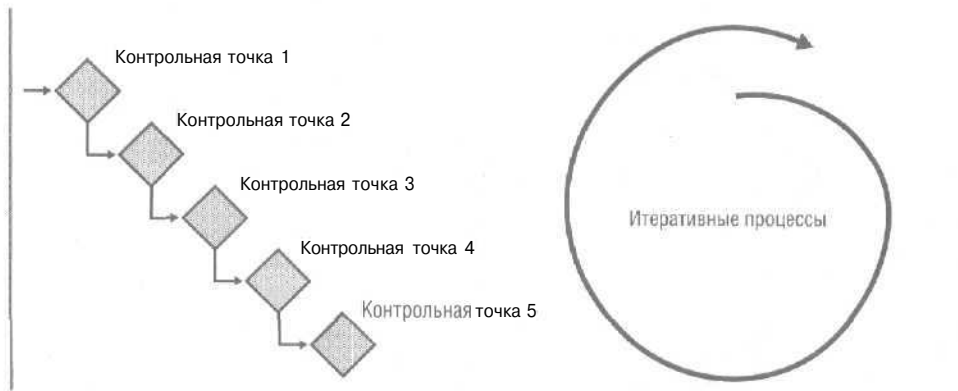


Рис. 1-1. Водопадная и спиральная модели

Эти модели представляют два разных подхода к организации жизненного цикла проекта.

- **Водопадная модель.** Здесь оценка и переход проекта на следующий этап выполняется в контрольных точках. Для перехода на следующий этап необходимо завершить все задачи предыдущего. Водопадная модель лучше всего подходит для проектов, в которых проектные требования поддаются четкой формулировке и не изменяются в дальнейшем. Модель предусматривает четкий переход от этапа к этапу, поэтому в ней очень просто контролировать графики и четко формулировать обязанности и ответственность различных сотрудников и ролей.
- **Спиральная модель** используется, когда необходимо непрерывно корректировать требования и параметры проекта. Эта модель эффективна при быстрой разработке приложений в небольших проектах. В такой ситуации команда разработчиков и клиент работают в тесном сотрудничестве, так как клиент привлекается на всех этапах, высказывая свое мнение о системе и одобряя успешно разработанные компоненты. Однако в спиральной модели отсутствуют четко определенные контрольные точки, поэтому есть риск, что процесс разработки станет хаотическим.

Как работает модель процессов MSF

Модель процессов MSF описывает общую последовательность действий по созданию и развертыванию решений уровня предприятия. Модель достаточно гибка и адаптируется в соответствии с самыми различными требованиями в проектах самого разного масштаба. Модель процессов MSF ориентирована на этапы, и управление в ней организовано на основе контрольных точек, а итерационный подход применяется при разработке и развертывании традиционных прикладных программ, корпоративных решений электронной коммерции и распределенных прикладных Web-программ.

Модель процессов MSF

В модели процессов MSF собрано все лучшее из водопадной и спиральной моделей: планирование на основе промежуточных контрольных точек и предсказуемость из водопадной модели наряду с обратной связью и коллективным творческим подходом, характерными для спиральной модели (рис. 1-2).



Рис. 1-2. Модель процессов MSF

Этапы модели процессов MSF

Модель процессов MSF состоит из пяти четко определенных этапов:

- создания общей картины приложения;
- планирования;
- разработки;
- стабилизации;
- развертывания.

Каждый этап завершается контрольной точкой. Рис. 1-3 иллюстрирует этапы и контрольные точки модели процессов MSF. Подробнее об этапах рассказывается на следующем занятии.

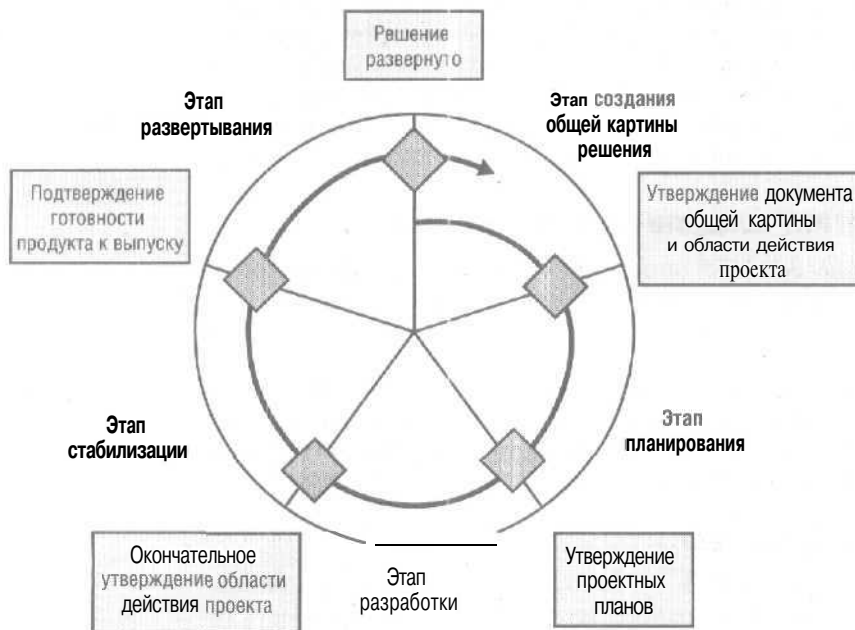


Рис. 1-3. Этапы и контрольные точки модели MSF

Организация проектных команд

Наряду с моделью процессов в MSF предусмотрена модель команд (MSF Team Model), которая применяется для организации проектных команд. Модель команд MSF подчеркивает важность четкого определения ролей, обязанностей и задач отдельных членов для успеха проекта, а также предполагает **повышенную ответственность** каждого члена команды. Благодаря гибкости она легко **адаптируется** в соответствии с потребностями и контекстом проекта, размером команды и опытом членов команды. Эта модель позволяет создавать эффективные, гибкие и успешные проектные команды.

Роли в модели команд MSF

Проект корпоративного решения предусматривает большое количество операций, а сам проект отличается разносторонностью. Для решения этих задач в модели команд MSF определены шесть специальных ролей, обязанности и задачи которых четко определены.

Внимание! Команда работает над решением одной задачи, а ее члены равноправны. В пределах команды каждая роль вносит свой вклад и разделяет **ответственность** за успех проекта.

Далее перечислены роли в модели команд MSF.

- **Менеджер решения** (product management) отвечает за управление связями с клиентом. На этапе проектирования на эту роль возлагается сбор клиентских требований и контроль за тем, чтобы они соответствовали и **удовлетворяли** все потребности бизнеса. Менеджер решения также работает над планом связей с клиентом в процессе создания и реализации проекта, в том числе над организацией встреч с клиентом, маркетинговых акций, демонстраций и представления продукта.
- **Менеджер программы** (program management) несет ответственность за разработку и поставку решения заказчику в полном соответствии с ограничениями проекта.
- **Разработчик** (development) обеспечивает разработку технологического решения в соответствии со спецификациями, предоставленными менеджерами решения.
- **Тестировщик** (testing) отвечает за выявление и устранение всех неполадок и проблем с качеством продукта и дает окончательное **«добро»** на выпуск и поставку решения. Эта роль оценивает и проверяет корректность набора функций и его соответствие общей концепции и области **действия** проекта.
- **Менеджер по выпуску** (release management) отвечает за развертывание и работу продукта. Он проверяет корректность инфраструктуры и определяет возможность развертывания и поддержки продукта.
- **Специалист по удобству использования** (user experience) анализирует потребности и проблемы, **возникающие** у пользователей, и оценивает продукт на предмет соответствия таким потребностям.

В небольшом проекте отдельные члены проектной команды часто совмещают несколько ролей. Имейте в виду, что объединение ролей проекта довольно рискованно, поэтому важно назначать участникам проекта «совместимые» роли. Например, не рекомендуется назначать одного сотрудника на роль менеджера программы и разработчика.

Минимизировать риски вам поможет таблица 1-1, в которой указаны разрешенные и не рекомендуемые комбинации ролей.

Таблица 1-1. Совмещение ролей в рамках команды

Роль	Менеджер решения	Менеджер программы	Разработчик	Тестирующий	Специалист по удобству использования	Менеджер по выпуску
Менеджер решения	Н	Н	В	В	Н/Ж	
Менеджер программы	Н		Н	Н/Ж	Н/Ж	В
Разработчик	Н	Н		Н	Н	Н
Тестирующий	В	Н/Ж	Н		В	В
Специалист по удобству использования	В	Н/Ж	Н	В		Н/Ж
Менеджер по выпуску	Н/Ж	В	Н	В	Н/Ж	

Условные обозначения: В — возможно, Н/Ж — нежелательно, Н — не рекомендуется.

Другие члены команды

Кроме уже описанных ролей, в проектную команду также входят заинтересованные в проекте лица (stakeholders), хотя они и не учитываются в модели команд MSF.

- **Инициатор, или спонсор проекта**, — одно или несколько лиц, иницирующих проект и одобряющих как сам проект, так и его результаты.
- **Заказчик (или бизнес-спонсор)** — одно или несколько лиц, которые планируют получить выгоду — повышение ценности бизнеса — от внедрения решения.
- **Конечный пользователь** — одно или несколько лиц или систем, которые непосредственно взаимодействуют с **решением**.
- **Организация, занимающаяся поддержкой решения**, — отвечает за сопровождение решения после его развертывания.

Дисциплины MSF

Руководящие принципы MSF предусматривают управление людьми, процессами и технологическими элементами, которые характерны для большинства проектов. Три ключевые дисциплины MSF — управление рисками, готовностью и проектом.

Управление рисками в MSF

Этот процесс предусматривает активное управление рисками, непрерывную их оценку и принятие решений на всех стадиях жизненного цикла проекта. Команда непрерывно оценивает, контролирует и активно управляет рисками до полного их устранения или пока они не приводят к появлению проблем, которые приходится решать.

Процесс управления рисками в MSF состоит из шести логических стадий, на каждой из которых проектная команда управляет текущими рисками, планирует, реализует и документирует стратегию управления рисками.

1. **Определение рисков** позволяет выявлять их, благодаря чему команда получает информацию о возможных проблемах.
2. **Анализ рисков** — преобразование оценок и информации о конкретных проектных рисках, обнаруженных на предыдущей стадии, в форму, пригодную для принятия решения об их важности.
3. **Планирование рисков** — использование результатов предыдущей стадии для формулировки стратегий, планов и мероприятий.
4. **Мониторинг рисков** — наблюдение за конкретными рисками и документирование планов мероприятий по управлению ими или устранению.
5. **Управление рисками** — процесс исполнения планов мероприятий по управлению и устранению рисков и отчетность о состоянии дел в этой сфере.
6. **Извлечение уроков** — формализация приобретенного опыта и соответствующих проектных документов и инструментальных средств и сохранение приобретенных знаний в форме, доступной для их повторного использования в командах и во всей компании.

Примечание Подробнее об управлении рисками в MSF рассказывается в главе 3.

Управление готовностью в MSF

Это процесс разработки знаний, навыков и возможностей, необходимых для создания и управления проектами и решениями. На рис. 1-4 показаны четыре стадии процесса управления готовностью: *определение, оценка, изменение* и *анализ*.

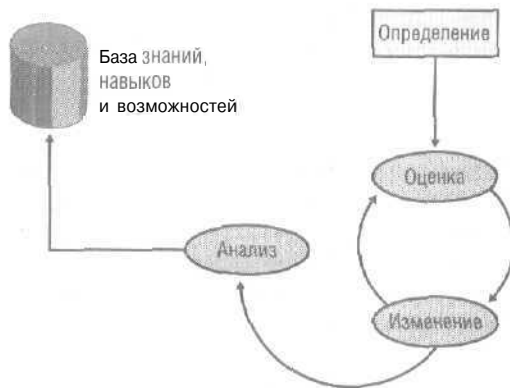


Рис. 1-4. Управление готовностью в MSF

Каждая стадия процесса содержит ряд задач, которые помогают перейти к следующей контрольной точке.

- **Определение** (define). На этой стадии команда определяет порядок, уровень компетентности и опыта, необходимые для успешного планирования, создания и управления решением. В это же время определяют, какой уровень компетентности и навыков необходим для каждой роли в организации. В описании роли указывается, какими знаниями и навыками должен обладать сотрудник, которому она предназначена.
- **Оценка** (assess). Именно на этой стадии в команде начинается анализ текущих навыков и их связи с различными ролями. Цель — определить набор навыков каждой роли, который затем сравнивается с набором, определенным

на предыдущей стадии. Сравнение текущей квалификации с требуемой, необходимо для разработки плана обучения, который «подтянет» членов команды до заданного уровня.

- **Изменение** (change). На этой стадии члены команды совершенствуют свои знания и навыки в процессе структурированного обучения, необходимого для **повышения** квалификации до **указанного** уровня. Эта стадия состоит из:
 - *обучения* ~ наставничества и тренингов в соответствии с планом обучения;
 - *мониторинга прогресса*, который позволяет отслеживать индивидуальную и общую готовность на **любом** этапе жизненного цикла проекта. Мониторинг позволяет вносить необходимые корректировки в план обучения.
- **Анализ** (evaluate). На этой стадии определяют эффективность планов обучения и выясняют, применяются ли полученные знания при работе над проектом.

Процесс управления готовностью в MSF — это непрерывный итеративный подход к решению задачи обеспечения готовности, применимый и в крупных, и в небольших проектах. Описанные процедуры позволяют эффективно управлять решением задач обеспечения достаточного уровня знаний, навыков и компетенции отдельных членов проектной команды и организации в целом.

Управление проектом в MSF

Чтобы создать продукт в рамках ограничений проекта, необходим значительный опыт. Управление проектом — это процесс, в котором набор навыков и методов применяется для решения следующих задач:

- обеспечения единых принципов планирования и управления изменениями;
- определения и управления областью действия проекта;
- подготовки бюджета и управления затратами;
- подготовки и контроля графиков;
- выделения для проекта подходящих ресурсов;
- управления контрактами и поставщиками и поставки ресурсов для проекта;
- поддержки внешних связей и общения внутри команд;
- поддержки процесса управления рисками;
- документирования и мониторинга процесса управления качеством.

В модели команд MSF не предусмотрена роль менеджера проекта, однако большинство функций по управлению проектом отведены роли менеджеру программы.

Отличительная особенность MSF в области руководства проектом заключается в том, что функции и операции по управлению проектом не организованы в иерархическую структуру **принятия** решений. Природе MSF *противен* твердый, централизованный стиль управления проектом, так как он препятствует эффективному формированию команды равных по положению членов, а это основное условие успеха MSF.

В MSF все роли в команде решают свои задачи и считаются одинаково важными. Ключевые решения принимаются при согласии членов, составляющих костяк команды. Если согласованного мнения достичь не удастся, менеджер программы принимает окончательное решение проблемы, взяв на себя роль ответственного за принятие решения — так обеспечивается безостановочная работа над проектом. Принимая решение, менеджер программы ориентируется на требования заказчика и необходимость поставить продукт в оговоренный срок.

После принятия решения команда работает в обычном режиме, предполагающем равноправие ее членов.

Управление компромиссами

Иногда в проектах возможны нарушение графиков или перерасход бюджета. Главная причина подобных проблем — нечетко описанная область действия проекта. *Область действия* (score) определяет, какие задачи решает продукт, а какие не относятся к его компетенции. Для эффективного определения рамок проекта и управления ими необходимо:

- определить ограничения проекта;
- организовать управление компромиссами;
- организовать управление изменениями;
- обеспечить мониторинг проекта.

В процессе определения и управления компромиссами не обязательно сужать функциональность решения, но все же зачастую компромисса избежать не удастся и набор функций приходится *сокращать*. Управление компромиссами — это структурированный метод достижения баланса всех составных частей проекта, когда в команде осознают, что в выделенное время не удастся решить всех задач. И команда, и заказчик должны анализировать компромиссы и быть готовыми к нелегкому выбору. Для этого в MSF предлагаются особые инструменты: треугольник компромиссов и матрица компромиссов проекта.

Треугольник компромиссов

В проектах существует вполне определенная связь между такими параметрами проекта, как ресурсы, график и функциональность проекта (рис. 1-5).



Рис. 1-5. Треугольник компромиссов

Как видно на рисунке, любое изменение одного из компонентов требует корректировки других компонентов. Ключ к разработке продукта, который соответствует требованиям заказчика, — определить и обеспечить соблюдение корректного баланса между ресурсами, датой поставки и функциональным наполнением.

Очень часто проектные команды неохотно соглашаются на сокращение функций. Треугольник компромиссов позволяет разобраться в ограничениях и предложить варианты решения.

Примечание Если в качестве четвертого измерения добавить качество, треугольник превратится в тетраэдр. Снижая требования к качеству, можно одновременно сократить объем ресурсов, ускорить дату поставки и расширить функциональность, но это очень плохой и, поэтому не рекомендуемый подход к разработке продукта.

Матрица компромиссов проекта

Матрица компромиссов проекта — инструмент, который команда и заказчик используют при принятии решений о компромиссах. Подобные решения следует принимать на ранних стадиях проекта. Пример матрицы компромиссов проекта показан на рис. 1-6.



Рис. 1-6. Матрица компромиссов проекта

Матрица позволяет разделить функции на обязательные, необязательные, но желательные, и факультативные, которые можно при необходимости исключить или отложить до следующей версии, чтобы удовлетворить первые два параметра. Чтобы лучше понять, как работает матрица компромиссов, представьте себе, что переменные — ресурсы, график и функциональность — необходимо разместить на свободных местах в следующем предложении:

Учитывая, что зафиксировано _____, мы определим _____ и в случае необходимости скорректируем _____,

Вот некоторые возможности.

- Учитывая, что зафиксированы ресурсы, мы определим график и в случае необходимости скорректируем функциональность.
- Учитывая, что зафиксированы ресурсы, мы определим функциональность и в случае необходимости скорректируем график.
- Учитывая, что зафиксирована функциональность, мы определим необходимые ресурсы и в случае необходимости скорректируем график.
- Учитывая, что зафиксирована функциональность, мы определим график и в случае необходимости скорректируем ресурсы.
- Учитывая, что зафиксирован график, мы определим необходимые ресурсы и в случае необходимости скорректируем функциональность.
- Учитывая, что зафиксирован график, мы определим функциональность и в случае необходимости скорректируем ресурсы.

Внимание! Чтобы извлечь пользу из компромиссов, проектная команда и заказчик должны согласовать и формально одобрить матрицу компромиссов проекта. Этот процесс еще называют *окончательным одобрением* (signing off).

Порядок применения итераций в проектах

В процессе разработки решения каждая итерация процесса приближает продукт к окончательному виду, предусмотренному в проекте. Итерации на каждой стадии продолжаются, пока не достигнута цель этой стадии. Кроме того, итерации позволяют разрабатывать проект за меньшее число шагов, так как число последующих итераций определяется на основании успешного или неудачного завершения предыдущего шага. Результаты — документ общей картины решения и другие документы, код, проекты и планы — разрабатываются по итерационному методу. Есть несколько видов итеративного подхода к разработке проекта:

- версии;
- обновляемые документы;
- периодические сборки (еженедельные или ежедневные).

Версии

Разрабатывая решения по методологии MSF, команда создает, тестирует и развертывает базовые функции, а затем в каждой последующей версии расширяет функциональность. Подобный подход называют стратегией, основанной на выпуске версий.

Рис. 1-7 иллюстрирует расширение набора функций с каждой последующей версией. Временной интервал между версиями зависит от размера и границ проекта.

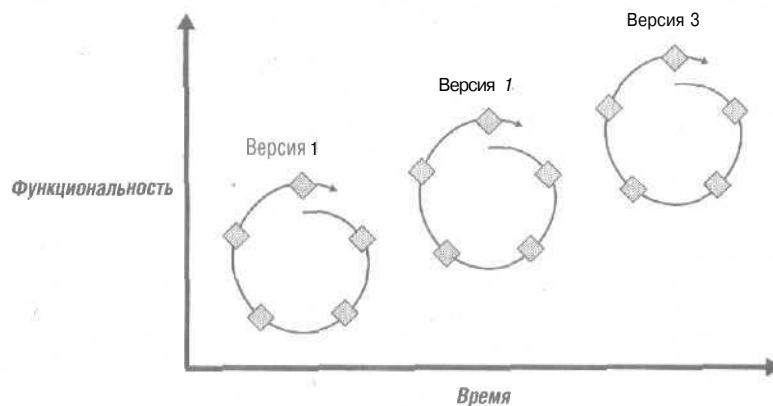


Рис. 1-7. Функциональность версий

Метод версий обеспечивает более тесную связь проектной команды с клиентом и гарантирует, что самые удачные идеи найдут отражение в продукте. Клиенту легче принять предложение отложить функцию до следующего выпуска, если он доверяет команде и уверен, что разработчик вовремя предоставит исходную и последующие версии решения. Следуя нескольким рекомендациям, удастся значительно упростить работу по методу версий.

- **Создание плана выпуска множественных версий.** При планировании реализации всех функций в нескольких версиях команде проще принимать решения о том, какие функции реализовать в текущей версии и что можно отложить до следующей. Это позволяет более эффективно использовать имеющиеся ресурсы и время. Кроме того, удастся предотвратить нежелательное расширение области действия проекта, создав четкий график развития функциональности продукта.

- **Ускорение итераций.** Важное преимущество метода версий в том, что клиент быстро получает работоспособный продукт, набор функций которого со временем расширяется. Четко определяйте область действия проекта, чтобы итерации укладывались в разумные временные рамки.
- **Поставка в первой версии только базовых функций.** Предоставление базового, но надежного и пригодного к использованию решения намного эффективнее, чем разработка продукта, который клиент сможет использовать только через недели или месяцы. Предоставив клиенту основные функции на первых этапах реализации решения, разработчики активно подключают клиента к участию в корректировке и совершенствовании продукта на последующих итерациях.
- **Создание в первую очередь рискованных функций.** В процессе оценки риска команда определяет самые рискованные функции. Именно их следует реализовать в первую очередь при создании варианта продукта, разрабатываемого для проверки верности концепции. В случае проблем, требующих коренных изменений архитектуры, вы сможете внести изменения на ранних стадиях проекта, что сэкономит значительные бюджетные средства.

Обновляемые документы

Чтобы не тратить слишком много времени на совершенствование проекта на начальной стадии, рекомендуется создавать *обновляемые документы* (living documents), то есть такие, что изменяются вместе с корректировкой проекта. «Живые» документы позволяют команде оперативно обновлять любую часть дизайна проекта и продолжать разработку на основании обновленных требований. Такой процесс гарантирует, что конечное решение будет удовлетворять финальным, а не только начальным требованиям. Документы в MSF-проекте разрабатываются итерационно.

Например, модель процессов MSF рекомендует, чтобы начальные версии документов по планированию были максимально обобщенными, без обширных подробностей. Их предлагают оценить членам команды и другим заинтересованным лицам на начальных стадиях проекта. По мере развития решения и перехода его на другие этапы документы уточняются и детализируются. Планы пересматриваются и изменяются итерационно. Тип и число планов определяются размером проекта.

Периодические сборки

MSF рекомендует частую сборку компонентов решения с последующим тестированием и анализом. Этот метод применяется при кодировании, а также при сборке аппаратных и программных компонентов. Ежедневные сборки дают четкое понимание стабильности всего решения и возможность накопить достаточно результатов тестирования перед финальным выпуском продукта.

Ежедневная сборка особенно эффективна в больших, сложных проектах, которые разделены на более мелкие. Отдельные команды работают и тестируют подсистемы, а затем объединяют их в единое решение. Сначала реализуются базовые функции, а позже добавляются дополнительные. Разработка и тестирование выполняется непрерывно и параллельно. Ежедневная сборка гарантирует совместимость всего кода и позволяет подкомандам уверенно переходить к следующей итерации разработки и тестирования.

Занятие 2. Этапы модели процессов MSF

На занятии 1 вы узнали, что модель процессов MSF ориентирована на контрольные точки и состоит из пяти этапов: создания общей картины приложения, планирования, разработки, стабилизации и развертывания. Набор действий проектной команды на каждом этапе четко определен. На этом занятии подробно рассказывается о каждом этапе, его контрольных точках и результатах.

Изучив материал этого занятия, вы сможете:

- ✓ перечислить задачи, контрольные точки и результаты всех этапов модели процессов MSF.

Продолжительность занятия — около 30 минут.

Этап создания общей картины приложения

Процесс MSF начинается с *создания общей картины приложения* (envisioning) — самого общего описания целей и ограничений проекта. На этом этапе определяют состав команды, а также выясняют, что она должна создать для заказчика. Цель этого этапа — выработать единое понимание проекта среди всех основных его участников.

На этапе создания общей картины приложения менеджмент управления программой определяет задачи и результаты, которые удовлетворяют **требования** и цели проекта. **Кульминация** этапа — достижение контрольной точки «Утверждение документа общей картины и области действия проекта», которое свидетельствует о том, что клиент и команда достигли согласия относительно целей и направления проекта.

Процесс создания общей картины приложения

На этапе создания общей картины приложения команда решает различные задачи.

- **Определение состава команды**, в которой должны быть представлены все роли, предусмотренные моделью команд MSF. (Сотрудника, ответственного за создание команды, обычно назначает руководство компании.) При организации команды важно учесть навыки, опыт и эффективность работы отдельных ее членов. Кроме того, не забудьте о практических соображениях, таких, как наличие и доступность ресурсов и бюджета.
- **Определение структуры проекта** — определение административной структуры проектной команды и стандартов управления проектом.
- **Определение бизнес-целей** — анализ бизнес-задачи и возможностей для выявления целей создания продукта.
- **Оценка существующей ситуации** — анализ текущего состояния и оценка разрыва между реальным и ожидаемым положением дел. Цель подобного анализа — сформулировать перечень задач и определить направление развития проекта.
- **Создание документа общей картины и области действия проекта** — разработка концепции решения, которым должна руководствоваться проектная команда, чтобы достичь долгосрочных бизнес-целей проекта, и ее документирование. Область действия проекта определяет, что включается в контекст проекта, а что выходит за его рамки.

Примечание Одобренная всеми участниками и четко сформулированная картина решения — залог успеха проекта.

- **Определение требований и профилей пользователей** — определение всех заинтересованных сторон, конечных пользователей и спонсоров проекта, а также документирование их требований к решению. Эта информация помогает «набросать» черновик общей картины и границ проекта, а также создать концепцию решения.
- **Разработка концепции решения** — создание базовой концепции решения, то есть «костяка» решения, которое станет основой будущего продукта. Концепция создается на основе собранных требований.
- **Оценка риска** — определение и выяснение важности различных видов риска для проекта, а также разработка мероприятий по устранению или снижению рисков. Это итерационная процедура, выполняемая на всех этапах жизненного цикла продукта.
- **Закрытие этапа создания общей картины решения** — завершение этапа, которое подтверждается документом общей картины и области действия решения, одобренным всеми заинтересованными лицами и проектной командой,

Контрольные точки этапа создания общей картины решения

Каждый этап модели процессов MSF содержит несколько промежуточных и одну основную контрольную точку. Первые относятся к операциям, которые выполняются на этапе, например создание команды и разработка документа общей картины и области действия. По достижении основной контрольной точки команда вправе перейти на следующий этап модели процессов MSF. Например, основная контрольная точка этапа создания общей картины решения — «Утверждение документа общей картины и области действия проекта». Далее команда переходит к этапу планирования.

Ниже перечислены промежуточные контрольные точки рассматриваемого этапа.

- **Организован костяк команды.** Определены ключевые члены команды. На этом этапе вам не нужен полный поименный список команды. Документ структуры проекта определяет роли и обязанности каждого члена команды, а также описывает иерархию отчетности и ответственности в команде, точки контакта с заказчиком и структуру команды.
- **Создана общая картина решения.** Первая версия документа общей картины решения создана и представлена на рецензию членам команды, заказчикам и участникам. Документ проходит стадии получения откликов (рецензий), обсуждения и модификации.

Этап создания общей картины решения завершается контрольной точкой «Утверждение документа общей картины и области действия решения». На этой стадии проектная команда и заказчик согласовали направление проекта, область действия решения и общий график поставки продукта.

Результаты этапа создания общей картины решения

Результаты решения каждой задачи этапа формируют контекст и направление последующих этапов проекта, а также общую картину и область действия решения, которые предоставляются заказчику. Вот цели, которых команда достигает на этапе создания общей картины решения.

- **Общая картина и область действия решения:**
 - формулировка задач и бизнес-целей;
 - анализ существующих процессов;
 - наиболее общее определение требований пользователей;
 - профили пользователей, определяющие, кто будет работать с продуктом;
 - документ общей картины и определение области действия;
 - концепция решения, описывающая способ планирования проекта;
 - стратегии проектирования решения.
- **Структура проекта:**
 - описание всех ролей команд MSF и списки членов команд;
 - структура проекта и стандарты процессов, которых должна придерживаться команда.
- **Оценка риска:**
 - « предварительная оценка риска;
 - список предварительно определенных рисков;
 - планы устранения или снижения влияния выявленных рисков.

Этап планирования

На этапе планирования команда решает, что следует разработать, и создает планы реализации продукта. Команда готовит функциональную спецификацию, создает дизайн решения и планы работы, а также оценивает стоимость и сроки получения запланированных результатов.

На этапе планирования выполняется анализ требований, которые делятся на бизнес-требования, пользовательские, функциональные и системные требования. Они необходимы для проектирования продукта и его функций, а также для проверки корректности проекта.

После сбора и анализа требований команда создает проект решения. Создаются профили, которые определяют пользователей продукта и их роли и обязанности. Затем команда формирует сценарии использования системы. Сценарий использования системы (СИС) — это описание процесса, выполняемого пользователями определенного типа. Команда создает отдельные СИС для всех пользовательских профилей. Затем формируются варианты использования системы (ВИС), которые определяют последовательность шагов, выполняемых пользователем в СИС.

Стадии дизайна

Этап планирования состоит из трех стадий.

- **Концептуальный дизайн.** Задача рассматривается с точки зрения пользовательских и бизнес-требований и определяется в виде сценариев использования системы.
- **Логический дизайн.** Задача рассматривается с точки зрения проектной команды, и решение определяется как набор сервисов.
- **Физический дизайн.** Задача рассматривается с точки зрения разработчиков (программистов). На этой стадии уточняются технологии, интерфейсы компонентов и сервисы решения.

Дизайн решения документируется в виде функциональной спецификации, которая описывает поведение и вид каждой функции будущего продукта, а также определяет архитектуру и дизайн всех функций.

Задачи этапа планирования

На этапе планирования команда выполняет ряд задач.

- **Разработка дизайна и архитектуры решения** — определение бизнес-требований, пользовательских требований и требований к технологиям, а также порядка использования этой информации для проектирования предлагаемой модели приложения.
- **Создание функциональной спецификации**, где описываются требования, которым должно удовлетворять будущее решение.
- **Разработка планов проекта** — определение и планирование задач, которые будет выполнять проектная команда, и их объединение в генеральный план проекта. Последний включает и другие элементы, такие, как методика, зависимости и допущения будущей системы.
- **Разработка календарного графика проекта** — создание генерального календарного графика проекта. Он состоит из выраженных в виде контрольных точек календарных графиков для каждой из ролей в проектной команде.
- **Создание среды разработки, тестирования и пилотной эксплуатации** — создание отдельной среды, в которой будет разрабатываться и тестироваться решение. Она не должна зависеть от среды, в которой в конечном итоге планируется развернуть продукт.
- **Закрытие этапа планирования** — завершение процесса одобрения контрольных точек. Документирование результатов выполнения задач на этапе планирования.

Контрольные точки этапа планирования

- **Окончательное одобрение используемых технологий.** На этой стадии команда оценивает продукты и технологии, которые предполагается использовать для создания или развертывания решения, а также изучает существующую производственную среду заказчика, в том числе конфигурации серверов, сеть, программное обеспечение персональных компьютеров и все имеющееся оборудование.
- **Завершение функциональной спецификации.** Эта контрольная точка предусматривает завершение работы над функциональной спецификацией и представление ее на рецензию заказчику и другим участникам проекта. Следует помнить, что документ дизайна отличается от функциональной спецификации, разрабатывается проектной командой и описывает внутренние механизмы работы будущего продукта.
- **Завершение генерального плана.** Этот план представляет собой совокупность планов всех ролей в команде, а его объем и сложность зависят от размера проекта.
- **Завершение генерального календарного графика.** Этот график состоит из всех детализованных графиков проекта, а также в нем указывается дата выпуска продукта. Так же, как и в генеральном плане проекта, в генеральном графике объединена информация о каждой из ролей в команде, только в данном случае речь идет о календарных графиках.

- **Организация средразработки и тестирования.** Специальная рабочая среда позволяет надлежащим способом разрабатывать и тестировать решение, а также избегать отрицательного влияния на системы, в которых в конечном счете предполагается развернуть приложение. Это также среда, где определяются и конфигурируются компоненты инфраструктуры, такие, как конфигурации серверов, инструменты автоматизации развертывания и оборудование.

Основная контрольная точка этапа планирования — «Утверждение проектных планов». Достижение этой контрольной точки свидетельствует о том, что проектная команда и основные участники проекта достигли соглашения о том, что промежуточные контрольные точки достигнуты, запланированные даты реалистичны, роли и обязанности в проекте четко определены, перечень результатов проекта согласован, а также предусмотрены механизмы устранения и управления рисков.

Результаты этапа планирования

Результаты этого этапа служат основанием для принятия компромиссных решений в дальнейшем. К результатам этапа планирования относятся:

- функциональная спецификация;
- план управления рисками;
- генеральный план и календарный график проекта.

Этап разработки

На этапе разработки проектная команда создает решение, в том числе разрабатывает и документирует код продукта, а также создает инфраструктуру решения.

Процесс разработки

На этапе разработки команда выполняет несколько задач.

- **Начало цикла разработки.** Команда проверяет выполнение всех задач, характерных для этапов создания общей картины решения и планирования, и готовится к началу разработки продукта.
- **Создание прототипа приложения.** Проверка концепций, заложенных в проекте решения, в среде, которая аналогична окружению, где в конечном счете предполагается развернуть будущий продукт. Среда должна максимально точно воспроизводить промышленную среду. Эта задача выполняется до начала разработки.
- **Разработка компонентов решения.** Разработка основных компонентов решения и их адаптация в соответствии с потребностями решения.
- **Создание решения.** Последовательность ежедневных или более частых сборок, которая завершается выпуском базовых сборок, знаменующих реализацию основных функций продукта.
- **Закрытие этапа разработки.** Завершение работы над всеми функциями приложения и поставка кода и документации. Решение считается готовым, а команда переходит к процессу одобрения контрольной точки.

Контрольные точки этапа разработки

- **Приложение для проверки концепции готово.** Это приложение необходимо для подтверждения верности выбранной концепции решения. Оно испытывается в среде тестирования. Команда демонстрирует решение группе админист-

раторов и пользователей, чтобы проверить, правильно ли поняты и реализованы их требования.

- **Завершение работы над внутренними сборками.** Поскольку продукт разрабатывается по частям, настоятельно рекомендуется синхронизировать все части на уровне единого решения. Для этой цели служат внутренние сборки. Число и частота внутренних сборок зависят от размера и сложности проекта.

Кульминационный момент этапа разработки — контрольная точка «Окончательное утверждение области действия проекта», в которой все функции продукта готовы и прошли тестирование в рамках своего модуля. Продукт готов к внешнему тестированию и стабилизации. Кроме того, заказчики, пользователи, персонал поддержки и сопровождения и ключевые участники проекта могут оценить продукт и указать все недостатки, которые необходимо устранить до поставки продукта.

Результаты этапа разработки

К результатам этапа разработки относятся:

- исходный текст и исполняемые файлы;
- сценарии установки и конфигурации для развертывания;
- завершенная функциональная спецификация;
- элементы поддержки решения;
- спецификации тестирования и сценарии тестирования.

Этап стабилизации

На этапе стабилизации команда собирает, загружает и выполняет бета-тестирование продукта, а также проверяет сценарии развертывания. Основное внимание уделяется обнаружению, определению важности и разрешению неполадок — все это готовит решение к финальному выпуску. На этом этапе обеспечивается заданный уровень качества продукта. Кроме того, по завершении этапа решение готово к развертыванию в промышленной среде.

Процесс стабилизации

На этапе стабилизации команда выполняет следующие задачи.

- **Тестирование решения.** Реализация планов тестирования для проверки корректности работы решения. Когда решение становится достаточно устойчивым, выполняется пилотная эксплуатация в тестовой среде. Тщательное тестирование продукта подразумевает:
 - тестирование компонентов;
 - тестирование баз данных;
 - тестирование инфраструктуры;
 - тестирование защиты;
 - тестирование интеграции;
 - проверку продукта на предмет удобства работы с ним;
 - нагрузочное тестирование продукта, а также тестирование ресурсоемкости и производительности;
 - регрессивное тестирование;
 - регистрацию числа ошибок.

- **Пилотная эксплуатация** — развертывание решения в тестовой среде и тестирование с привлечением будущих пользователей и применением реальных СИС.

Контроль и отчетность по тестированию

Контроль и генерация отчетности по тестированию выполняются через короткие интервалы на этапах разработки, тестирования и стабилизации. На этапе стабилизации самая важная информация отчетов — число неполадок и ошибок. Регулярное информирование команды и других заинтересованных лиц обеспечивает высокий уровень осведомленности участников проекта.

Контрольные точки этапа стабилизации

- **Сходимость числа ошибок.** Эта контрольная точка свидетельствует об успехах команды в работе над ошибками и неполадками, а сходимость числа ошибок означает, что число устраненных ошибок превышает количество вновь обнаруживаемых. Далее количество обнаруживаемых ошибок может как повышаться, так и падать — даже после значительного общего снижения, поэтому сходимость числа ошибок — это скорее тенденция, а не фиксированный момент времени.
После начала сходимости число ошибок должно далее снижаться — вплоть до выпуска версии, в которой не будет обнаружено ни одной ошибки. Сходимость числа ошибок говорит о приближении завершения работы над продуктом.
- **Версия, в которой не обнаружено ни одной ошибки.** Достижение этой контрольной точки свидетельствует о том, что больше не удастся найти ни одной неполадки и число обнаруженных ошибок равно нулю.
- **Кандидаты на выпуск** — ряд контрольных точек, которые отражают четкое снижение числа неполадок и ошибок по сравнению с таковым в контрольной точке «Версия без обнаруженных ошибок».
- **«Золотая» версия** — контрольная точка, в которой удовлетворяется требование отсутствия ошибок (они не обнаружены) и критерии успешного завершения разработки.

Этап стабилизации завершается в контрольной точке «Подтверждение готовности продукта к выпуску». После анализа и одобрения решение готово к полноценному развертыванию в промышленной среде. Эта контрольная точка достигается, когда команда устранила все неполадки и предоставила продукт заказчику. В ней ответственность за сопровождение и поддержку продукта официально переходит к команде поддержки и сопровождения.

Результаты этапа стабилизации

К результатам этапа стабилизации относятся:

- финальная версия;
- информативные документы о версии;
- элементы поддержки производительности;
- результаты и инструменты тестирования;
- исходный текст и исполняемые файлы;
- проектные документы;
- обзор контрольных точек.

Этап развертывания

На этом этапе команда развертывает технологии и компоненты окружения, необходимые для работы созданного продукта, устанавливает и стабилизирует решение в развернутом состоянии, передает проект в руки команды сопровождения и поддержки и получает окончательное одобрение проекта заказчиком. После развертывания команда выполняет анализ проекта и проводит опрос, чтобы выяснить уровень удовлетворенности заказчика. Этап развертывания завершается контрольной точкой «Решение развернуто».

Процесс развертывания

На этапе развертывания проектная команда выполняет несколько задач.

- **Завершение развертывания и вспомогательных процедур** — создание формальной документации по развертыванию и административным процедурам, в которой описывается, как проектная команда планирует развертывать и переводить продукт в режим сопровождения.
- **Развертывание и стабилизация** — завершение развертывания окружения и самого решения.
- **Анализ проекта** — совместное выполнение послепроектного анализа заказчиком и проектной командой.

Примечание Операции по стабилизации могут продолжаться на протяжении всего этапа, так как компоненты проекта переводятся из тестовой в промышленную среду.

Контрольные точки этапа развертывания

Далее перечислены промежуточные контрольные точки этапа развертывания.

- **Развернуты основные компоненты.** Большинство инфраструктурных решений содержит компоненты, которые обеспечивают фундамент решения. Хотя с точки зрения заказчика эти компоненты и не являются решением, успех развертывания зависит от них. Базовые технологии могут развертываться до или одновременно с продуктом — все определяется самим решением. Во избежание задержек основные компоненты следует анализировать и одобрять до развертывания, одновременно со стабилизацией других частей решения.
- **Развертывание решения завершено.** По достижении этой контрольной точки все соответствующие пользователи получают доступ к решению. Ведущие разработчики отдельных мест развертывания должны отчитаться о запуске решения в подведомственных им областях. Эта контрольная точка часто отсутствует в проектах, не предполагающих развертывание на стороне клиента.
- **Развернутое решение стабилизировано.** В этой контрольной точке заказчик и команда достигают согласия об удовлетворительной работе продукта. Отдельные неполадки, возможные при установке решения, должны быть обнаружены и устранены.

Примечание Заказчик может утвердить достижение командой намеченных целей до запуска решения в промышленную эксплуатацию и закрыть проект. Для этого необходимо наличие устойчивого решения и четко определенных критериев успеха. Чтобы решение считалось устойчивым, следует реализовать соответствующую систему сопровождения и поддержки.

Время между контрольными точками «Развернутое решение стабилизировано» и «Развертывание завершено» иногда называют *периодом затишья* (quiet period). Активность команды снижается практически до нуля, но она оперативно реагирует на обнаружение ранее не замеченных неполадок и ошибок. Период затишья длится примерно 15–30 дней. В это время команда занимается измерением производительности и быстродействия решения и при необходимости оценивает трудоемкость сопровождения и поддержки, а также может приступить к работе над следующим выпуском продукта.

- **Решение развернуто.** Это завершающая контрольная точка этапа развертывания. По ее достижении развернутое решение должно выполнять все возложенные на него функции.

Иногда трудно определить момент завершения развертывания. Недавно развернутые системы часто проходят через процесс выявления неполадок и управления поддержкой при промышленной эксплуатации. Команде иногда трудно закрыть проект из-за вновь возникающих неполадок и проблем, проявляющихся после развертывания. Поэтому необходимо четко определить контрольную точку завершения развертывания и не пытаться отладить абсолютно все.

Результаты этапа развертывания

К результатам этапа развертывания относятся:

- системы сопровождения и поддержки:
 - процедуры и процессы;
 - база знаний, отчеты и журналы;
- хранилище документов, где размещаются все версии документов и кода, разработанных в течение проекта;
- план обучения;
- отчет о завершении проекта:
 - финальные версии всех проектных документов;
 - сведения об уровне удовлетворенности заказчика;
 - определение следующих стадий.

Занятие 3. Исходные данные для разработки проекта для Adventure Works Cycles

На этом занятии описывается конкретная компания, на примере которой мы будем иллюстрировать концепции и процедуры проектирования приложений в Microsoft .NET Framework. Исходные данные для этого примера будут использоваться в других главах.

Изучив материал этого занятия, вы сможете:

- ✓ определить основные бизнес-требования к приложению, которое планируется разработать для компании Adventure Works Cycles Application.

Продолжительность занятия — около 10 минут.

Пример — компания Adventure Works Cycles

Цель данного примера — помочь будущим архитекторам понять характер бизнес-проблем, которые им придется решать. Adventure Works Cycles — крупная международная компания, производящая и поставляющая велосипеды из металла и композитных материалов на рынки Северной Америки, Европы и Азии. Помимо расположенных в г. Ботшелл, штат Вашингтон, штаб-квартиры и основного завода, где работают 500 человек, компания имеет несколько региональных торговых подразделений, расположенных во всех странах, где Adventure Works Cycles продает свою продукцию.

В 2000 г. Adventure Works Cycles приобрела небольшой завод в Мехико, где производит важные детали для велосипедов, которые собирают в Ботшелле. В 2001 г. подразделение в Мехико — Wide World Importers — стало единственным производителем и распространителем туристических велосипедов.

После успешного в финансовом отношении года Adventure Works Cycles планирует увеличить свою долю на рынке за счет активизации сбыта среди лучших клиентов компании, повышения доступности товаров через внешний Web-сайт и сокращения сбытовых затрат за счет снижения производственных издержек.

Бизнес-задачи

Бизнес-задачи мы будем классифицировать по подразделениям компании — отдел продаж, управление персоналом, завод Wide World Importers, отдел снабжения, ИТ-отдел, производственный отдел, отдел системного администрирования и технический отдел.

Отдел продаж

Торговые представители региональных отделений отвечают за сбыт на территории Соединенных Штатов, Канады, Англии, Австралии, Германии и Франции. Штаб каждого регионального отделения состоит из нескольких торговых представителей и менеджера по продажам. В своей повседневной работе торговые представители используют ноутбуки и КПК под управлением Microsoft Windows CE.

Обычный рабочий день торгового представителя начинается с подключения к региональному отделению и загрузки свежих данных, в том числе информации

о наличии товаров на складе, сведений о товарах и маркетинговой информации. При посещении клиентов торговый представитель вносит сведения о заказах в свой ноутбук или КПК. В конце каждого рабочего дня он назначает встречи на следующий день или неделю, просматривает встречи, запланированные другими представителями, чтобы определить возможность совместных действий, и обновляет список контактов. Торговый представитель подключается по модему к региональному отделению, отправляет новую информацию и принимает свежие данные из главного или регионального отделения. В качестве почтового клиента в компании в настоящее время применяется Microsoft Outlook.

Команды сбытовиков определили, какие именно новшества позволят им лучше выполнять свою работу.

- **Сегментация и профилирование клиентов.** Команда отдела продаж должна на основании «сырых» данных получать полезную информацию, позволяющую ответить на следующие вопросы.
 - Каковы ранние признаки возможных проблем?
 - Кто лучшие клиенты в каждой из линеек товаров? С кем следует наладить долгосрочные отношения?
 - Каковы проблемы заказчиков, распределенных по категориям: географическое местоположение, уровень доходов и т.п.?
 - Сколько и какие товары приобретают клиенты?
- **Процесс продаж.**
 - Текущая политика скидок предполагает, что торговые представители имеют право предоставить скидки в размере до 15% на каждый заказ. Менеджеры по продажам вправе увеличить скидку до 20%. Приложение должно позволять сотрудникам компании предоставлять скидки в соответствии с их ролью.
 - Для координации сотрудничества отделений в различных странах отдел по продажам нуждается в поддержке международных операций, в том числе ему требуется возможность получать информацию о продукте, например дату и цены, на многих языках и в различных валютах.
- **Внутренняя связь.** Каждый торговый представитель должен получить только предназначенную ему информацию о клиенте и продажах. Каждый менеджер по продажам должен получать информацию о своих клиентах и назначенных встречах наряду с детальной информацией о каждом торговом представителе своего отделения. Менеджер должен иметь возможность выделять клиентов торговым представителям на основании их отношений с клиентами, хотя обычно клиенты распределяются по региональному признаку.
- **Управление потенциальными клиентами.** Торговым представителям необходим механизм сохранения и просмотра данных о потенциальных клиентах и возможность при заключении сделки переносить часть или всю необходимую информацию в заказ, чтобы не вводить ее повторно.
- **Система поддержки принятия решений** должна:
 - позволять сотрудникам отдела маркетинга и продаж запрашивать и использовать данные о клиентах при генерации стандартных отчетов, выполнять специальные (нестандартные) запросы, получать информацию о маркетинговых кампаниях, прогнозе продаж и сегментации клиентов, а также обращаться к сторонним источникам данных и инструментам финансовой оценки;

- представлять все операции **клиентам** в унифицированном виде, включая повторные контакты, встречи и сделки;
- позволить сотрудникам отдела маркетинга инициировать новые маркетинговые программы в масштабе **всей** компании. В **настоящее** время торговые представители не знают, как связать эти программы с **деятельностью** всех подразделений компании для достижения максимального эффекта;
- определять, анализировать и совместно использовать все виды информации о клиентах во многих отделах.

Отдел управления персоналом

Текучесть среди «почасовиков» составляет 10%, а среди штатных сотрудников — 25%. Руководство считает, что в следующем году компании придется увеличить штат на 35%, чтобы восполнить отток персонала и удовлетворить **растущие** потребности производства в **рабочей** силе. Кроме того, на заводе Wide World Importers много внимания требует план найма и удержания сотрудников из-за роста числа персонала и кандидатов на вакансии.

Руководство решило, что компания нуждается в решении, которое позволит отслеживать карьеру текущих сотрудников и поток резюме от кандидатов. Кроме того, руководство требует повысить точность прогнозирования требуемого числа **сотрудников** и планирования изменений в схемах вознаграждения. Отдел управления персоналом предъявляет к будущему решению следующие требования.

- **Управление банком резюме и анкет.** Все резюме и анкеты хранятся в документах различных форматов. Приложение должно обеспечить:
 - унифицированную систему хранения записей всех типов;
 - доступ к существующим данным о сотрудниках (данные хранятся в таблицах реляционной СУБД) со ссылками на резюме и анкеты;
 - инструментальные средства преобразования файлов любого типа в документы единого формата, что позволит совместно использовать их разными отделами;
 - возможность закрыть определенным пользователям доступ к некоторым областям документов, например к сведениям о зарплате;
 - возможность искать резюме или анкеты по ключевым словам или фразам.
- **Анализ и планирование.** Отдел управления персоналом нуждается в поддержке для выполнения следующих видов анализа:
 - компенсационного пакета и льгот с учетом изменения обменных курсов валют;
 - планирования и оценки требуемой рабочей силы;
 - моделирования и прогнозирования затрат на заработную плату.

Завод Wide World Importers

Завод расположен в г. Мехико и производит некоторые важные **компоненты**, которые используются при производстве велосипедов компанией Adventure Works Cycles. Компания недавно приобрела этот завод с целью расширения своей инфраструктуры в связи с ожидаемым ростом компании. Wide World Importers считается отдельным подразделением, однако руководство полагает, что некоторые приложения и данные завод и компания должны использовать совместно. Далее перечислены **бизнес-задачи**, связанные с **деятельностью** завода.

- **Обмен данными.** Adventure Works Cycles не в состоянии обеспечить регулярный перенос и обмен данными из-за того, что:
 - в Wide World Importers нет средств высокоскоростного обмена данными между локальной базой данных и тремя базами данных на основе Microsoft SQL Server Б Ботшелле;
 - вычислительная среда Adventure Works Cycles централизована. Требуется увеличить масштабируемость текущей базы данных путем перехода к распределенной среде;
 - в настоящее время не обеспечивается безопасность сетевого подключения, по которому осуществляется обмен информацией.
- **Администрирование и поддержка.** Ресурсы поддержки и администрирования на заводе Wide World Importers сильно ограничены. Отделы поддерживают собственные рабочие станции и серверы с установленными базами данных Oracle. Обслуживание и управление осуществляется двумя администраторами и менеджером по информационным системам. На заводе отсутствуют многие стандартизированные процессы, которые используются в повседневной работе компании Adventure Works Cycles.

Отдел снабжения

В настоящее время Adventure Works Cycles сотрудничает с несколькими поставщиками деталей и материалов, необходимых для производства. В отделе снабжения определили, что требуется организовать электронный обмен данными (Electronic Data Interchange, EDI) с ключевыми поставщиками, т. е. обеспечить обмен такими критически важными документами, как заказы на поставку, счета, платежные поручения и спецификации продукта. Снабженцы и сотрудники бухгалтерии больше половины своего рабочего времени тратят на взаимодействие с главными поставщиками. Руководство отдела ожидает, что после внедрения программы автоматизации EDI-обмена эффективность труда сотрудников отдела возрастет на 30%.

- Приложение для Adventure Works Cycles должно отвечать ряду требований.
- Необходимо обеспечить поддержку обмена самой разнообразной информацией между поставщиками и Adventure Works Cycles, в том числе структурированными и слабоструктурированными данными.
 - Поставщики должны иметь возможность загружать свои данные непосредственно в таблицы закупок базы данных Microsoft SQL Server.
 - Каждому поставщику необходимо обеспечить поддержку обмена данными через Web по защищенным каналам.
 - Возможность автоматически обнаруживать наличие у поставщиков новой информации для Adventure Works Cycles.

Отдел информационных технологий

Adventure Works Cycles собственными силами поддерживает и развивает Интернет-сайт. В настоящее время клиентам предоставляется основная информация о товарах, сведения о ближайшем отделении компании и возможность распечатать эту информацию. На сайте нельзя заказать товары в онлайн-режиме или просмотреть состояние существующего заказа. Возможность поиска на Web-сайте информации о продукте ограничена просмотром товаров по категориям. Для увеличения клиентской базы и предоставления дополнительных возмож-

стей существующим клиентам компания Adventure Works Cycles планирует вернуть Интернет-сайт со следующими функциями:

- онлайнный заказ товаров клиентами;
- онлайнная проверка состояния заказа;
- расширенные возможности поиска информации о товаре;
- возможность обращаться к отдельным разделам технических спецификаций товара;
- возможность просматривать информацию о товаре в разных валютах и на различных языках.

Кроме Интернет-сайта в Adventure Works Cycles создан небольшой интрасетевой Web-сайт, состоящий из домашней страницы и ссылок на сайты отделов компании. Каждый отдел поддерживает собственный сайт и уведомляет ИТ-отдел о любых изменениях на домашней странице отдела. Основная задача интрасетевого сайта — обеспечить связи внутри компании. Хозе Люго (Jose Lugo), финансовый менеджер Adventure Works Cycles, говорит, что очень сложно получать информацию от одного подразделения и предоставлять ее другим подразделениям. Заведующие отделами желают, чтобы больше информации и операций по доступу к данным выполнялись через внутренний Web-сайт. Интрасетевой Web-сайт должен обеспечивать следующие возможности:

- поиск информации в разных подразделениях;
- изменение и обновление внутренней информации (изменения цен, жалобы клиентов и др.);
- доступность данных о товаре всем отделам. Настройка доступа к информации для нужд различных подразделений (технический отдел, отделы продаж и маркетинга).

Производственный отдел

Ассортимент продукции Adventure Works Cycles состоит из девяти модельных рядов велосипедов. Каждый товар содержит один или больше подкомпонентов в зависимости от требований клиента. В настоящее время производственный отдел получает спецификации из технического отдела и использует их для сборки продукции. Всю информацию спецификации, необходимую для производства, вводит специально выделенный сотрудник производственного отдела. Часть базовой информации о товаре предоставляется клиентам посредством внешнего Web-сайта, однако им недоступна полная спецификация — она опубликована на внутреннем сайте и доступна только работникам отдела продаж.

По мере перемещения будущего товара по сборочному конвейеру он комплектуется соответствующими материалами и деталями — вплоть до завершения сборки. Вот связанные с производством задачи, которые следует решить.

- Планирование и производство. Для предотвращения задержек производства команде необходимы:
 - более простой доступ к такой производственной информации, как перечень производственных заказов и наличие запасов на складе для отдельных участков сборочного конвейера;
 - возможность анализировать данные производственного процесса.
- Ревизия складских запасов. Вместо ручной ревизии и обновления информации о складских запасах компания планирует организовать маркировку штрих-кодами и использовать КПК для управления информацией о товаре на складе.

Отдел системного администрирования

Полная доступность ключевых систем Adventure Works Cycles стала центром внимания отдела системного администрирования. Постоянно растет число жалоб на простой систем,

В отделе системного администрирования отсутствует централизованная система контроля и управления базами данных. В результате команда поддержки базы данных разрабатывает собственные операции и процедуры. Подобный децентрализованный механизм не позволяет оптимизировать использование ресурсов. Кроме того, сотрудники отдела системного администрирования требуют реализовать более совершенный способ управления системными ресурсами во всех операционных системах.

Чтобы решить бизнес-задачи, отдел системного администрирования должен обеспечить:

- постоянную доступность сервисов, что обеспечит поддержку ключевых систем Adventure Works Cycles;
- наличие резервных систем и политик восстановления всех баз данных компании Adventure Works Cycles, а также завода World Wide Importers;
- мониторинг всех ресурсов во всех операционных системах.

Технический отдел

Основная обязанность технического отдела — проектирование основных компонентов продукции Adventure Works Cycles. Команде необходима система управления потребностями отдела в документировании. Далее перечислены некоторые требования к этой системе.

- **Коллективное управление информационным наполнением.** Отделу необходима автоматизированная система управления рецензированием, одобрением и выпуском чертежей и спецификаций для нужд производства.
- **Доступ и хранение различных типов файлов.** Отдел нуждается в унифицированном методе хранения, поиска, выборки и архивирования файлов различных типов: чертежей, созданных в системах автоматизированного проектирования, и XML-спецификаций товаров. Эта система должна быть связана с табличными данными, которые применяются для управления версиями спецификаций и чертежей.

Требования к системе для компании Adventure Works Cycles

Система для Adventure Works Cycles должна быть реализована в виде Web-приложения; она предназначается как для клиентов, так и других компаний, в том числе розничных магазинов, продающих продукцию Adventure Works Cycles. Кроме того, в приложении следует предусмотреть несколько Web-страниц, на которых соискатели на должности в компании могли бы загружать и редактировать свои резюме.

Бизнес-задачи

Приложение для Adventure Works Cycles должно выполнять следующие задачи.

- Клиенты должны иметь возможность размещать заказы.
- Клиенты должны иметь возможность удалять заказы.
- Приложение должно «уметь» отклонять заказы.

- Клиенты должны иметь возможность изменять существующие заказы, находящиеся в обработке.
- Клиенты и агенты-посредники должны иметь возможность просматривать текущие заказы.
- Клиенты должны иметь возможность создавать новые клиентские записи,
- Клиенты должны иметь возможность редактировать информацию о себе.
- Клиенты должны иметь возможность размещать на сайте свои резюме для рассмотрения в качестве претендента на открывающиеся вакансии.
- Клиенты должны иметь возможность находить и редактировать ранее размещенные резюме,

Примечание Клиентам и посредникам предоставляется доступ только к собственной информации.

Требования к Web-сайту

- На каждой странице Web-сайта должна присутствовать функция поиска с соответствующими элементами управления, а в левой части страницы — навигационная панель.
- Заказчик должен иметь возможность искать товары по части описания товара или по цене.
- На домашней странице следует опубликовать список товаров, которые имеются сейчас в продаже, а также информацию о специальных предложениях. Обязательно наличие изображения и описания для каждого наименования товара.
- На странице групп товаров необходимо иерархическое дерево ссылок на соответствующие модели велосипедов или аксессуары к ним.
- Страница описания товара должна содержать информацию об одной модели — название и описание, большое фото и цены. Если доступны различные цвета и/или размеры товара, необходимо предусмотреть соответствующие раскрывающиеся списки. При выборе клиентом размера или цвета второй список должен обновляться, а диапазон цен — сужаться в соответствии с выбранным вариантом.
- На текущей странице заказа будут отображаться товары, заказанные клиентом, с указанием количества и суммарной стоимости. Клиент должен иметь возможность изменить количество или удалить отдельные товары. Кнопка «Продолжить покупку» возвращает заказчика на последнюю посещенную страницу описания товара.
- Сайт должен предоставлять зарегистрированным клиентам возможность войти в предназначенную для них область сайта по реквизитам — адресу электронной почты и паролю. Чтобы получить доступ к этой области, новые клиенты должны зарегистрироваться. (Посредникам в эту область доступ не предоставляется.)
- В области для зарегистрированных посетителей следует предусмотреть для них возможность вводить или изменять свои адреса электронной почты, пароли и другую персональную информацию.
- На странице определения адреса клиенты должны иметь возможность создавать, просматривать, изменять и удалять адреса предоставления счетов и отгрузки.

- На странице выбора адреса клиентам должна предоставляться возможность выбирать или удалять в заказе адреса предоставления счета и отгрузки.
 - На странице информации о заказе следует отображать сведения о заказе, в том числе налог с продаж, стоимость транспортировки и информацию о скидках посредника. Клиент должен иметь возможность изменять количество и удалять отдельные товары из заказа.
 - На странице оплаты необходимо предусмотреть возможность платить с помощью кредитных карточек — путем ввода всей информации о кредитной карточке или указания карточки, сведения о которой введены ранее и хранятся в базе данных. Посредникам необходимо предоставить возможность выбора типа платежа и ввода порядкового номера заказа.
 - На странице подтверждения заказа должна отображаться сводная информации о введенном заказе, в том числе дата и состояние заказа, а также номер подтверждения.
 - На странице контроля за состоянием заказов должен отображаться список всех заказов, введенных зарегистрированным клиентом. Клиенту следует предоставить возможность выбрать заказ, информация о котором будет отображаться на этой странице.
 - На странице открытых вакансий клиентам надо предоставить возможность размещать свои резюме.
- В дальнейших главах этой книге мы подробно расскажем о процессе проектирования решения, отвечающего требованиям Adventure Works Cycles.

Резюме

- Модель процессов MSF представляет собой сочетание водопадной и спиральной моделей.
- В модели процессов MSF объединены контрольные точки и итерационный метод.
- В модели команд MSF определены шесть ролей: менеджер решения, менеджер программы, разработчик, тестировщик, менеджер по выпуску и специалист по удобству использования продукта.
- Область действия проекта определяется в процессе управления компромиссами.
- В MSF рекомендован итерационный подход к проектированию решения и.
- Итерации в жизненном цикле разработки реализуются в виде регулярных версий или за счет поддержки обновляемых документов.
- В MSF документ общей картины и области действия решения создается на этапе создания общей картины решения MSF
- Функциональные спецификации создаются на этапе планирования модели процессов MSF.
- Решение разрабатывается на этапе разработки.
- Тестирование решения и пилотная эксплуатация выполняются на этапе стабилизации.
- Развертывание решения в промышленной среде и передача команде поддержки и сопровождения выполняются на этапе развертывания,

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Опишите различия между водопадной и спиральной моделями и расскажите, как они используются в модели процессов MSF.
2. Кто согласно модели команд MSF отвечает за процесс проектирования?
3. Треугольник компромиссов описывает три типа возможных компромиссов. Какой существует четвертый вид компромисса и почему к нему никогда не следует прибегать?
4. Заполните матрицу компромиссов на основании приведенного ниже утверждения.
«Учитывая, что зафиксирована функциональность, мы **определим** необходимые ресурсы и в случае необходимости скорректируем график».

	Зафиксировано	Определено	Корректируемо
Ресурсы			
График			
Набор функций			

5. Объясните, зачем в модели процессов MSF используются ежедневные сборки.
6. Какой этап завершается контрольной точкой «Подтверждение готовности продукта к выпуску»?
7. На каком этапе модели процессов MSF создается документ предварительной оценки рисков?
8. Когда в модели процессов MSF создаются сценарии тестирования?
9. Перечислите виды **тестирования**, выполняемого на этапе стабилизации.
10. Почему так важно создать документ общей картины решения на соответствующем этапе модели процессов MSF?
11. Перечислите ключевые задачи, выполняемые на этапе планирования?
12. Что такое «период затишья» и какие операции выполняются на этой стадии?

ГЛАВА 2

Сбор и анализ информации

Занятие 1. Сбор информации	32
Занятие 2. Анализ информации	42
Занятие 3. Использование нотаций моделирования	49
Занятие 4. Создание ВИС и СИС	60
Практикум. Сбор и анализ информации	68
Резюме	69
Закрепление материала	70

В этой главе

Сбор и анализ информации — этапы, предусмотренные набором моделей процессов Microsoft® Solutions Framework (MSF). В занятиях этой главы рассказано как собирать и анализировать информацию: информацию каких типов следует собрать, ее источники и описание некоторых методов сбора информации. Вы также научитесь анализировать всю собранную информацию и познакомитесь с некоторыми методами анализа информации.

Примечание В этой главе представлена информация, относящаяся ко всем этапам, а не характерная для какого-то определенного этапа процессов MSF. Очень важно освоить методы сбора и анализа информации до начала этапа создания общей картины решения.

Прежде всего

Для изучения материалов этой главы необходимо:

- знать модели процессов в MSF;
- установить на компьютере приложение Visio® 2000 Professional, которое необходимо для создания диаграмм.

Занятие 1. Сбор информации

До сбора информации следует разобраться в ее различных типах и характеристиках, чтобы быть уверенным, что именно эти сведения необходимы. Цель данного занятия — научиться «видеть» информацию о бизнесе с различных точек зрения. Чем шире обзор, тем с большей долей вероятности вы извлечете все необходимые входные данные для проведения эффективного анализа.

Изучив материал этого занятия, вы сможете:

- ✓ определить типы информации, которую следует собирать;
- ✓ определить методы сбора информации;
- ✓ определить источники информации, используемые для сбора требований;
- ✓ разработать стратегию сбора информации.

Продолжительность занятия - около 20 минут.

Категории информации

Архитектура предприятия — это моментальный снимок бизнеса как динамической системы в определенный момент времени. Благодаря корпоративной архитектуре обеспечивается организация ИТ-групп и процессов согласно целям бизнеса. Для сбора информации об этой архитектуре применяются четыре описательные категории архитектуры предприятия, необходимые для управления и классификации собираемой информации: бизнес, приложения, процедуры и технологии.

Бизнес

Эта категория описывает сам бизнес, в том числе функции и межфункциональные операции, осуществляемые в компании. К информации этого типа относятся обусловленные бизнесом высокоуровневые цели и задачи, товары и услуги, финансовые структуры, интегрированные бизнес-функции и процессы, основные организационные структуры и взаимодействие этих составляющих. Это сведения об общих бизнес-стратегиях и планах по преобразованию компании.

Приложения

Эта категория включает сервисы и функции, которые обычно выходят за рамки организации и связывают пользователей, обладающих различными навыками и выполняющих отличные функции, для достижения общих целей бизнеса. Здесь описываются автоматизированные и неавтоматизированные сервисы, поддерживающие бизнес-процессы. Это сведения о взаимодействии и взаимозависимостях между бизнес-системами.

К автоматизированным бизнес-сервисам обычно относят полнофункциональные приложения, утилиты, инструменты повышения производительности, компоненты и программные модули для анализа информации или функций заданий.

Нередко в организациях одни и те же задачи выполняются с применением различных инструментов. Собирая информацию о процессах организации, обязательно исследуйте все приложения, при помощи которых осуществляется та

или иная деятельность в компании. Существующие приложения или их элементы часто становятся основой для новых решений. Повторное использование существующих сервисов рентабельнее, чем создание их «с нуля». Собранный информация помогает оптимизировать бизнес-процессы за счет выявления неэффективных или повторяющихся действий.

Эта категория предоставляет информацию о текущем использовании систем и сервисов. Кроме того, информация от пользователей и бизнес-документов позволяет получить представление о направлении развития компании.

Процедуры

К ним относятся знания, необходимые организации для выполнения бизнес-процессов и процедур. Эта категория охватывает стандартные модели данных, политики управления данными и структуру потребления и создания информации, порождаемой в процессе ведения бизнеса.

Вы должны определить источники, владельцев и потребителей информации. Помимо определения стандартов и правил репликации, репозитория и хранилищ данных, отслеживание и анализ доступа к информации и способов ее использования необходимо для принятия решений, касающихся распределения, репликации и разбиения данных на разделы. Зачастую потребителей опрашивают недостаточно полно, поэтому очень сложно определить не только то, какая информация им нужна, но и то, что они с ней собираются делать.

Взаимосвязь между ключевыми бизнес-процессами и информацией, необходимой для их реализации, позволяет выработать стандарты и правила создания, выборки, обновления и удаления данных, совместного использования жизненно важных документов и сведений, а также уровни конфиденциальности и доступа. Следует понимать, что далеко не вся информация централизована или легко доступна всем сотрудникам или системам, которые в ней нуждаются. Обычно наиболее критично важная для бизнеса информация располагается на серверах баз данных, на рабочих станциях, из которых собственно и состоит активная рабочая бизнес-среда, а также зачастую — в головах сотрудников (т.е. какая-то часть вообще нигде не записана).

Технологии

Здесь определяются технические сервисы, необходимые для реализации и поддержки основной миссии бизнеса, в том числе топологии, среды разработки, интерфейсы прикладного программирования (application programming interface, API), безопасность, сетевые сервисы, сервисы СУБД, технические спецификации, аппаратные уровни, операционные системы и т.п.

Также к этой категории относится информация о корпоративных стандартах и правилах, определяющих порядок приобретения и развертывания инструментов для рабочих станций и серверов, основные приложения, инфраструктурные сервисы, сетевые компоненты и платформы.

Технологии обеспечивают связь между приложениями и информацией. Приложения создаются и основываются на различных технологиях, кроме того, применяются специализированные технологии доступа к информации, которая хранится с использованием различных технологий.

Эти сведения нужны для определения стандартных интерфейсов, сервисов и моделей приложений, которые следует использовать в процессе разработки. Данные такого рода часто существуют в виде ресурсов для разработки в проектных

командах, таких, как библиотеки компонентов и кода, стандартные документы и правила проектирования. На их основе часто определяются проектные цели и ограничения приложения.

Методы сбора информации

Существуют шесть основных методов сбора информации:

- наблюдение за действиями пользователя (shadowing);
- интервьюирование (interviewing);
- беседа с целевыми группами (focus groups);
- опросы (surveys);
- обучение, проводимое пользователями (user instruction);
- создание прототипов (prototyping).

Примечание Не обязательно использовать все перечисленные методы — достаточно выбрать те, что наилучшим образом подходят для конкретного источника информации. Более того, вам не обязательно быть экспертом во всех них. Если потребуется вы всегда можете пройти обучение или обратиться за консультацией к эксперту.

Наблюдение за действиями пользователя

Вы наблюдаете за тем, как пользователь выполняет свои задачи в реальной рабочей среде, а также по ходу задаете ему любые необходимые вопросы. Вы должны стать «тенью» пользователя на время, когда он выполняет свою повседневную работу. Таким образом вы получаете информацию из первых рук, «в контексте», и выясняете цели той или иной задачи. Чтобы получить максимум информации, необходимо «вытягивать» из пользователя как можно больше деталей и подробностей о том, почему задача выполняется так, а не иначе.

Наблюдение бывает пассивным или активным. В первом случае вы просто наблюдаете и прислушиваетесь к пояснениям, которые пользователь, возможно, даст. Во втором вы активно задаете вопросы, а иногда и самостоятельно выполняете некоторые задачи.

Совет по планированию Описываемый метод хорош для анализа часто выполняемых задач. А вот когда задача не столь типична, иногда непросто найти возможность понаблюдать, как она выполняется «в жизни».

Наблюдение за пользователями — хороший способ разобраться, чем сотрудник занимается изо дня в день. Но вам вряд ли удастся увидеть весь объем работы, так как не все задачи выполняются в одном сеансе. Например, бухгалтеры готовят отчеты только в конце месяца, разработчики пишут отчеты о ходе работ раз в неделю, а менеджеры собираются на планерки раз в две недели.

В придачу к полученной от отдельных сотрудников информации, вам, возможно, удастся получить сопутствующие артефакты, такие, как документы и ментальные снимки экранов имеющихся приложений.

Примечание С точки зрения сбора информации артефакт — это любая сущность, физически существующая в бизнес-среде и описывающая элемент или основной бизнес-процесс. Подробнее об артефактах мы поговорим далее на этом занятии.

Примеры вопросов, задаваемых при наблюдении за пользователями

Собирая информацию методом наблюдения, постарайтесь выяснить у пользователя некоторые вопросы.

- Как пользователь структурирует свою работу?
- Какие решения он принимает в начале и при завершении задачи?
- Как пользователи выполняют свою работу в текущей реализации?
- Как часто система мешает выполнять задачи?
- Как воздействуют на пользователей перерывы в работе? В состоянии ли они продолжить работу с того же места, на котором их прервали?
- Со сколькими сотрудниками взаимодействует пользователь при выполнении той или иной операции?
- Какие усовершенствования внес пользователь, чтобы упростить выполнение задачи?
- Возможна ли модификация тех или иных стадий выполнения задачи? Каковы возможности и от каких условий они зависят?

Кроме изучения порядка выполнения работы пользователями рекомендуется выявить те составляющие существующего решения, которые не удовлетворяют пользователей или даже усложняют им жизнь.

Во время наблюдения на некоторые вопросы постарайтесь найти ответы самостоятельно.

- Как в настоящее время пользователи выполняют свои задачи?
- Как повысить эффективность процессов? Обязательно ли все выполняемые вручную задачи должны возлагаться на автоматизированное решение?
- Какие из связанных задач могут повлиять на структуру приложения?
- Какие системные функции необходимы для поддержки этих задач?
- Каковы критерии производительности?
- Как должны быть структурированы функции решения?
- Как можно улучшить существующую систему?
- Какие функции используются часто, а какие реже?
- Что пользователям нравится в существующей системе, а что нет?
- Как снизить затраты на обучение и поддержку?
- Какая информация о пользователях не задокументирована?
- Что представляет из себя рабочее окружение пользователей?
- Каковы характеристики и предпочтения пользователей?
- Какие концепции и терминологию используют пользователи?
- Какое обучение проходили пользователи?
- Какие тренинги им еще нужны?
- Пользователи проходили обучение организованно или учились самостоятельно? Как обучение (или его отсутствие) сказалось на их работе с системой и выполнении их обязанностей?

Примечание Обязательно наблюдайте не только за пользователями, но и за руководством и задавайте им все необходимые вопросы. Если имеются внешние клиенты, не обойдите вниманием и их.

Интервьюирование

Наблюдение — эффективное средство для выявления того, что же происходит в компании, однако оно не всегда обеспечивает всю необходимую информацию. Например, оно не очень годится для сбора информации о таких задачах, как менеджерские операции, долговременные операции, растянутые на недели, месяцы или годы, а также процессы, почти или совсем не требующие заметного вмешательства человека. К последним относится работа службы автоматической оплаты счетов, поддерживаемая финансовыми организациями. Для сбора информации о подобной деятельности и процессах необходимы интервью.

Интервью — это встреча один на один члена проектной команды и пользователя или действующего субъекта (stakeholder). Качество информации, собранной командой зависит от навыков как интервьюирующего, так и интервьюируемого. Интервьюер, сумевший «втереться в доверие» узнает много нового о затруднениях и ограничениях существующего решения. Интервью дает возможность задать массу вопросов по темам, которые невозможно раскрыть средствами простого наблюдения за повседневной работой сотрудников.

Вот некоторые моменты, о которых следует помнить перед началом интервью:

- начинайте с общих (не специализированных) вопросов и стройте беседу так, чтоб интервьюируемый думал обо всех выполняемых им задачах и информации, которую он способен вам сообщить;
- на основании ответов на вопросы попросите интервьюируемого изложить наиболее общие задачи и разбить их на более мелкие;
- особопросите выделить информацию, которая обычно отсутствует, а также возможные альтернативные пути решения задач;
- пройдите по заданным вопросам несколько раз, повторно спрашивая интервьюируемого, не упустили ли вы что-то важное;
- попросите интервьюируемого изложить свои мысли, как можно улучшить ситуацию, но ни в коем случае не предполагайте, что предлагаемое решение правильное.

Примеры вопросов для интервью

Проводя интервью, тщательно структурируйте вопросы таким образом, чтобы не вводить человека в заблуждение, и избегайте вопросов, в ответ на которые можно получить информацию более чем одного типа. Вот примеры вопросов, задаваемых в процессе интервью.

- С какими проблемами вы сталкиваетесь при выполнении своих задач?
- В какой помощи вы нуждаетесь, работая удаленно?
- Есть ли у вас какие-либо потребности кроме задокументированных?
- Какие бизнес-политики помогают (или, наоборот, мешают) вам выполнять свои обязанности?
- Какие люди или документы снабжают вас информацией, необходимой для выполнения работы?
- Какие еще пользователи или системы влияют на вашу работу (например, сторонние поставщики или специалисты службы поддержки)?

Беседы с целевыми группами

Это встречи, на которой сотрудники, обсуждая определенную тему, предоставляют организатору встречи необходимую информацию. Применяйте этот ме-

тод, когда пользователей больше, чем вы в состоянии вовлечь в процесс сбора информации. Чтобы при работе с целевыми группами собрать полезную информацию, позаботьтесь, чтобы их участники были пользователями или действующими лицами связанных бизнес-процессов. Также необходимо убедиться в корректности определения темы для обсуждения и следить, чтобы группа от нее не отклонялась.

Подобные встречи позволяют собирать подробную информацию о том, как та или иная деятельность соотносится с бизнесом в целом. Члены целевой группы должны дополнять друг друга, чтобы получить полное описание бизнес-процесса.

Беседы с целевой группой не дадут ожидаемого результата, если ее участники разделены географически. Также, затея может закончиться неудачей, если отобранные для встречи пользователи не объединены одной цепочкой бизнес-операций или не обладают достаточными знаниями о ней. Наконец, разговор с целевой группой бесполезен, если организатор не способен направить дискуссию в нужное русло и следить, чтобы участники не отвлекались от основного предмета разговора.

Опросы

Этот метод предполагает подготовку набора вопросов, специально разработанных для сбора информации. Примеры опросников: регистрационные формы, формы обратной связи или формы для оценки удовлетворенности респондентов.

Разработка вопросов подчас весьма трудоемка, а для их составления и анализа результатов требуется профессионал. Опросы позволяют собрать информацию и определить, какие еще действия для уточнения информации необходимо предпринять.

Одно из преимуществ опросов — анонимность пользователей. Так удается добыть информацию, которую невозможно получить любым другим способом. Однако вы должны обеспечить конфиденциальность, чтобы защитить участников опроса. Возможно, придется изменить или сделать более нейтральным язык ответов, чтобы исключить возможность определения того, кто их дал. С помощью опросов извлекаются данные, которые легко поддаются группировке и интерпретации.

Примечание На результаты опроса сильно влияет позиция респондента, поэтому они весьма субъективны.

Вот примеры информации, которую можно собрать путем опросов:

- организационные структуры, политики или правила, которые содействуют или препятствуют выполнению задач;
- неудовлетворенность технической поддержкой или политикой;
- особые требования к аппаратному или программному обеспечению;
- проблемы обучения, такие, как эффективность существующих учебных программ, типы учебных программ, которые предпочитают пользователи, а также программы обучения, дающие наилучший результат в рабочем окружении пользователя.

Обучение, проводимое пользователями

Пользователи обучают вас выполнению своих задач. Вы принимаете участие в действиях пользователя и изучаете каждый этап процесса с его точки зрения. Вы также получаете знания, которые пользователь приобрел с течением времени и которые не представлены в артефактах или системах.

Инструктирование потребует значительных временных затрат, если исследуемый процесс довольно длительный. Этот метод может не дать результатов, если пользователь не умеет или не способен обучать других. Кроме того, различные сотрудники могут по-разному выполнять одни и те же задачи. Поэтому вы должны собрать информацию у многих пользователей.

Совет Определите экспертов по различным бизнес-операциям. Они знают, как решаются *возникающие* проблемы, и помогут вам разрабатывать новые процессы или совершенствовать уже существующие.

Для сбора информации о работе пользователей вы также вправе прибегнуть к *службе поддержки (help-desk) предприятия*. Это хороший источник, так как именно здесь удастся напрямую узнать о проблемах, с которыми сталкиваются пользователи. Сбор информации в службе поддержки — одна из разновидностей получения инструкций от пользователей.

Когда пользователи обучают вас выполнению задач, вы *получаете* информацию, которая позволяет определить:

- дизайн пользовательского интерфейса;
- необходимость в обучении для поддержки существующих и будущих процессов;
- критерии производительности системы;
- воздействие, которое оказывает на выполнение задачи физическая среда.

Создание прототипов

Сбор информации выполняется путем моделирования производственной среды. Предусмотрено несколько инструментов для сбора информации: видеочкамера для визуального наблюдения за действиями или программа, отслеживающая нажатия клавиш и щелчки мышью. Выбор инструмента моделирования зависит от типа информации, которую требуется собрать.

Используйте прототипы, когда невозможно понаблюдать за пользователем в обычной рабочей среде. Сведения, собранные методом прототипов, скорее эмпирические. Однако их легко проверить, хотя стоимость создания прототипов может оказаться высокой.

Прототипы помогают проверить или задокументировать информацию с точки зрения пользователя и бизнеса, а именно:

- требования клиента к качеству;
- требования к времени отклика;
- простоту использования;
- интеграцию существующих технологий и приложений;
- проблемы пользовательского интерфейса, например, что из того, что хотят видеть пользователи, следует добавить в приложение;
- проверку цепочек технологических процессов.

Источники информации

Вы уже знаете о различных типах информации о бизнесе. Она бывает самых разнообразных форм. Количество и многообразие источников информации определяется масштабом бизнеса. Вот некоторые из них:

- артефакты;
- системы;
- люди.

Артефакты

С точки зрения сбора информации *артефакт* — это любой объект, физически существующий в бизнес-среде, описывающий элемент основного бизнес-процесса. Как и в археологии, артефакты помогают понять существующую окружающую среду. Они предоставляют информацию о задачах, процессах, бизнес-требованиях и ограничениях. Это могут быть: учебные материалы, видеозаписи, требования закона, более ранние программные файлы или магнитные ленты, документация информационно-справочной службы и финансовая отчетность.

Некоторые используемые в бизнесе артефакты определяются легко. Например, сотрудники хранят часто используемые артефакты на своих рабочих местах. Другие артефакты хранятся в картотеках или на компьютерных носителях и поэтому не столь заметны. Для *тщательного* изучения всей информации вам может потребоваться доступ к конфиденциальным артефактам. Проявляйте *осмотрительность* при работе с артефактами, прямо или косвенно являющимися интеллектуальной собственностью, от которой зависит конкурентоспособность предприятия.

Сотрудники предприятия, как правило, *разрабатывают* свои собственные артефакты. Например, создают инструкции по работе с приложением или по выполнению какого-либо процесса или описывают выполнение задачи в электронном письме. Подобные неофициальные артефакты являются ценным источником информации. Они отражают не только повседневную работу пользователей, но обеспечивают дополнительную информацию, не представленную в существующем решении, а также цель выполнения тех или иных задач. Иногда эти артефакты оказываются единственной документацией, относящейся к отдельным процессам.

Во время работы над проектом команда создает такие артефакты, как заметки о заседаниях, сводки собранной информации о бизнесе и документ, представляющий *общую картину решения* (vision document). Некоторые из них, например протоколы совещаний, используются только проектной группой, другие, представляющие общую картину решения, необходимы как проектной команде, так и представителям компании.

Системы

Это часть *предприятия*, которая выполняет определенные действия. Каждая отдельная система представляет собой набор дискретных процессов, *ориентированных* на выполнение определенного действия. Системы часто состоят из подсистем. Система может быть как материальным активом, например система управления складом, так и нематериальным, например методы, используемые менеджером для выявления и решения проблем в рамках отдела.

Системы бывают сложными, так как подчас они объединяют многие подсистемы и работают с информацией самых разных категорий. Кроме того, не всегда сразу удастся «увидеть» все подсистемы. Поэтому, чтобы полностью понять систему, следует запланировать дополнительное время. Системы позволяют узнать, как на предприятии выполняются рутинные ежедневные операции.

Совет Неплохо заручиться поддержкой специалиста по исследуемой системе и добиться, чтобы он «провел экскурсию» по выполнению различных процессов, поддерживаемых системой.

Люди

Люди, непосредственные участники процессов, становятся ценным источником информации о порядке ведения бизнеса, часто предоставляя информацию, которая не отражена в документах. Это могут быть руководители, разработчики, менеджеры, клиенты и пользователи. Если документация не отличается полной (или вовсе отсутствует), люди становятся единственным источником необходимой информации.

Прежде чем обратиться к людям за информацией, следует идентифицировать роли, которые эти люди играют в бизнесе. Например, участники команды, отвечающей за информационную и техническую поддержку пользователей, помогая клиентам в выполнении ежедневных задач, способны пояснить, где пользователи испытывают наибольшие трудности. **Сотрудники-ветераны** с опытом выполнения различных бизнес-операций просветят вас насчет взаимосвязей между различными видами деятельности. Если компания пользуется услугами сторонних поставщиков, их уникальный взгляд на эффективность бизнеса и его процессов может оказаться весьма полезным.

Совет по планированию Перед беседой с пользователями необходимо поподробнее узнать о применяемых ими системах. Это знание поможет избежать общих вопросов типа «Как вы используете эту систему?», «Чем вы занимаетесь в течение дня?» или «Как построен ваш рабочий день?».

Определение стратегии сбора информации

До сбора информации необходимо определить его стратегию: определить круг пользователей, составить вопросы и выбрать подходящие методы сбора информации.

Определяя стратегию сбора информации, руководствуйтесь следующими вопросами:

- какие источники необходимо использовать для добывания информации;
- какую информацию необходимо собрать;
- какие методы сбора информации вы планируете использовать и к каким источникам тот или иной метод применим;
- как предполагается хранить собираемую информацию;
- сколько времени вы планируете собирать информацию?

Ниже перечислены основные принципы, которыми следует руководствоваться при определении стратегии сбора информации:

- **Используйте несколько методов сбора информации.** Если вы положитесь лишь на один-единственный метод сбора, полученная картина бизнеса может оказаться неполной. Комбинируя различные методы, удается преодолеть недостатки, присущие каждому в отдельности. Допустим, что бизнес-процесс, длящийся несколько дней, происходит раз в месяц в отделе бухгалтерии. Наблюдая за одним из сотрудников бухгалтерии лишь несколько дней, вы можете упустить важную информацию о таком процессе, если именно в это время сотрудник не будет задействован в нем. Дополняя метод наблюдения за пользователем беседами с целевыми группами и интервьюированием, вы сможете собрать информацию обо всех процессах, протекающих в бухгалтерии.
- **Определите наиболее эффективный метод.** Разрабатывая стратегию сбора информации, взвешивайте преимущества и недостатки каждого метода. Допустим, вам необходимо собрать информацию как можно быстрее, чтобы решить проблему, недавно возникшую на предприятии. Опросы требуют времени на планирование, администрирование и анализ, а вот наблюдение, интервью или обучение, проводимое пользователями может значительно ускорить сбор сведений.
- **Держите в уме все точки зрения, типы и источники информации.** Независимо от применяемого метода, помните, что следует собрать информацию со всех ракурсов — как пользователей, так и бизнеса. За время, которым вы располагаете, исследуйте как можно больше источников информации.
- **Получайте информацию от групп, участвующих в схожих бизнес-процессах.** Различные группы в рамках одного бизнеса могут участвовать в схожих процессах, решая одинаковые бизнес-задачи. Полученная от этих групп информация подчас позволяет проектной команде взглянуть на проблемы бизнеса с новой точки зрения.

Занятие 2. Анализ информации

На этом занятии речь пойдет об анализе собранной информации. Процессы сбора и анализа выполняются **итеративно**: исследователь получает данные и анализирует их, повторяя эту процедуру несколько раз. При изучении полученной информации у вас, без сомнения, возникнут новые вопросы. Вы их сформулируете и повторно обратитесь к соответствующим источникам для выяснения, затем продолжите анализ бизнеса. Такая последовательность действий продолжается на протяжении всего цикла проекта, хотя большая часть работы по сбору и анализу информации все-таки выполняется на начальном этапе.

Изучив материал этого занятия, вы сможете:

- ✓ анализировать информацию об архитектуре предприятия;
- ✓ описать варианты и сценарии использования системы;
- ✓ создать **внутреннюю** проектную документацию.

Продолжительность занятия — около 20 минут.

Информация об архитектуре предприятия

Собрав достаточно информации от клиента, следует ее обработать, чтобы определить, какая информация существенна для **решаемых бизнес-задач**. Вы должны выполнить синтез информации и создать детальное описание текущего состояния бизнеса.

В процессе анализа собранной информации следует убедиться, что ее достаточно для отражения текущего состояния бизнеса и требований к продукту, в том числе позволяет ли она определить:

- требования по безопасности;
- необходимость вспомогательных структур решения и их характеристики;
- запланированные изменения в бизнесе, которые способны повлиять на дизайн продукта;
- производительность, которую ожидают пользователи или требует бизнес, чтобы компания оставалась конкурентоспособной;
- **существующие** приложения, которым придется взаимодействовать с новым продуктом;
- влияние **существующих бизнес-процессов** на создаваемое решение.

В процессе синтеза и анализа вы сможете выявить «белые пятна» в собранной информации и при необходимости получить дополнительные сведения.

По мере создания конечного продукта проектной командой необходимо следить, чтобы он удовлетворял всем требованиям, определенным в процессе сбора и анализа информации. Также команда должна задокументировать влияние, которое новый продукт окажет на **существующую** среду в плане новых требований к бизнесу, таких, как **проблемы** поддержки, обслуживания и расширяемости. Новые требования должны также согласовываться с ограничениями, задокументированными во время проведения анализа.

Высокоуровневые варианты и сценарии использования системы

После синтеза информации переходят к разработке *вариантов* (use case) и *сценариев использования системы* (usage scenario), то есть более детального документирования бизнес-процессов и требований бизнеса и пользователей. Готовые варианты и сценарии использования системы будут положены в основу структуры, которая необходима команде разработчиков для проектирования решения. Кроме того, каждый вариант или сценарий использования системы соответствует одному или нескольким требованиям. Проверка этого соответствия позволяет проверить, все ли требования удовлетворены.

Варианты использования системы

Варианты использования системы (ВИС) отражают набор функций системы и то, как сотрудник взаимодействует с системой для поддержки работы бизнеса.

Основное назначение ВИС:

- определить бизнес-процесс и все действия от его начала и до конца;
- задокументировать проблемы контекста и среды;
- установить связь между потребностями бизнеса и требованиями пользователей;
- описать потребности и требования в контексте использования;
- сориентировать усилия пользователей и команды разработчиков.

ВИС обеспечивают следующие преимущества:

- обеспечивают контекст требований;
- облегчают понимание общих принципов функционирования бизнеса;
- служат базисом для сценариев использования;
- способствуют объективности и целостности при оценке предложений пользователей.

Сценарии использования системы

ВИС описывают высокоуровневое взаимодействие между пользователем и системой, а сценарии использования системы (СИС) обеспечивают *дополнительную* информацию о действиях и последовательности задач, из которых состоит процесс. В совокупности ВИС и СИС описывают последовательность операций.

Создавать ВИС и СИС вы научитесь на занятии 4.

Предварительный документ о требованиях

После получения информации от клиента одним из этапов анализа является создание *чернового документа о требованиях с предварительным списком требований*, созданным на основе собранных сведений. Основное назначение этого документа — зарегистрировать все возможные требования и по ходу удостовериться, что не утеряна никакая важная информация. Полученные из всех источников данные должны содержать требования и пожелания с точки зрения бизнеса и пользователей. Требования поясняют, что должно делать приложение, чтобы решать задачи бизнеса, а пожелания отражают то, что участники и пользователи хотели бы видеть в конечном продукте или решении.

Требования, перечисленные в предварительном документе, недостаточно проработаны и иногда даже представляют смесь требований и пожеланий. Они более глубоко прорабатываются на следующих этапах — создания общей картины реше-

ния и планирования, когда команда получит от клиентов больше информации. Подробнее об определении требований, пожеланий и ограничений — в главе 4.

Пример интервью

Далее приводится пример интервью, взятого у сотрудника компании Adventure Works Cycles, описанной в главе 1.

Основное из интервью регионального менеджера по продажам

Этот год был весьма удачен в плане продаж. Компания сильно упростила нам жизнь, снабдив портативными и настольными компьютерами. Мы делали прогнозы на следующий год и пришли к выводу, что столкнемся с небольшим падением объемов продаж, особенно если будем продолжать работать по старинке. Отдел продаж наметил несколько целей на следующий финансовый год. Основные три из них: (а) сосредоточиться на самых прибыльных клиентах, (б) эффективнее использовать рабочее время сотрудников отдела продаж и (в) лучше реагировать на конъюнктуру рынка.

У нас нет простых способов анализа клиентской базы. В настоящее время информация о клиентах хранится в наших системах, но ее сложно извлекать и очень трудно организовать ее просмотр в различных ракурсах. Для более точного определения, кто же (и почему) относится к нашим лучшим клиентам, нам необходим доступ к данным и способ разумного их анализа.

Другая проблема, с которой мы столкнулись в последние месяцы, — представление и продажи нашей продукции на зарубежных рынках. В настоящее время вся информация на наших компьютерах представлена только на английском языке. Необходимо, чтобы информация в базе данных существовала на разных языках и в различных форматах, принятых в других регионах, тогда нам не придется перекладывать перевод информации на отделы продаж.

Нашей команде необходимо получать самые свежие сведения о ценах ежедневно. В настоящее время наши торговые представители каждое утро подключаются к корпоративной сети и загружают прайс-лист. Однако в процессе загрузки не предусмотрена процедура выделения только изменившихся цен, поэтому торговому представителю приходится каждое утро загружать весь прайс-лист. Представляете, что чувствуют клиенты, которые сразу после заключения сделки узнают, что цены изменились.

Другая область, несомненно нуждающаяся в улучшении, — управление нашими возможностями по сбыту. Сотрудники должны использовать нашу систему управления клиентской информацией для планирования, выполнения и отслеживания продаж и маркетинговой стратегии. Кроме того, для нормальной работы торговые представители должны устанавливать на своих портативных компьютерах большие приложения и подключаться к корпоративной сети. Требуется большая мобильность: необходимо иметь возможность получать эту информацию с Web-сайта. Информация должна быть легко доступна, и ее должны однозначно понимать торговые представители и остальные сотрудники компании, в противном случае от нее мало толку. От новой системы мы ждем следующего: минимизировать объем технических знаний, необходимых сотрудникам отделов продаж и маркетинга для доступа к данным, и дать возможность сотрудникам получать стандартные отчеты, создавать индивидуальные запросы, отслежи-

вать акции по продвижению товаров и услуг и просматривать информацию о сегментации клиентов. Кроме того, система должна быть достаточно гибкой, позволяя нам добавлять источники данных и инструменты оценки финансового состояния сторонних компаний. Независимо от того, кто просматривает информацию о том или ином клиенте, сотрудник должен получить ясные, унифицированные данные о клиенте и отношениях компании с ним.

Пример ВИС

На рис. 2-1 показан предварительный ВИС, созданный на основании этого интервью.

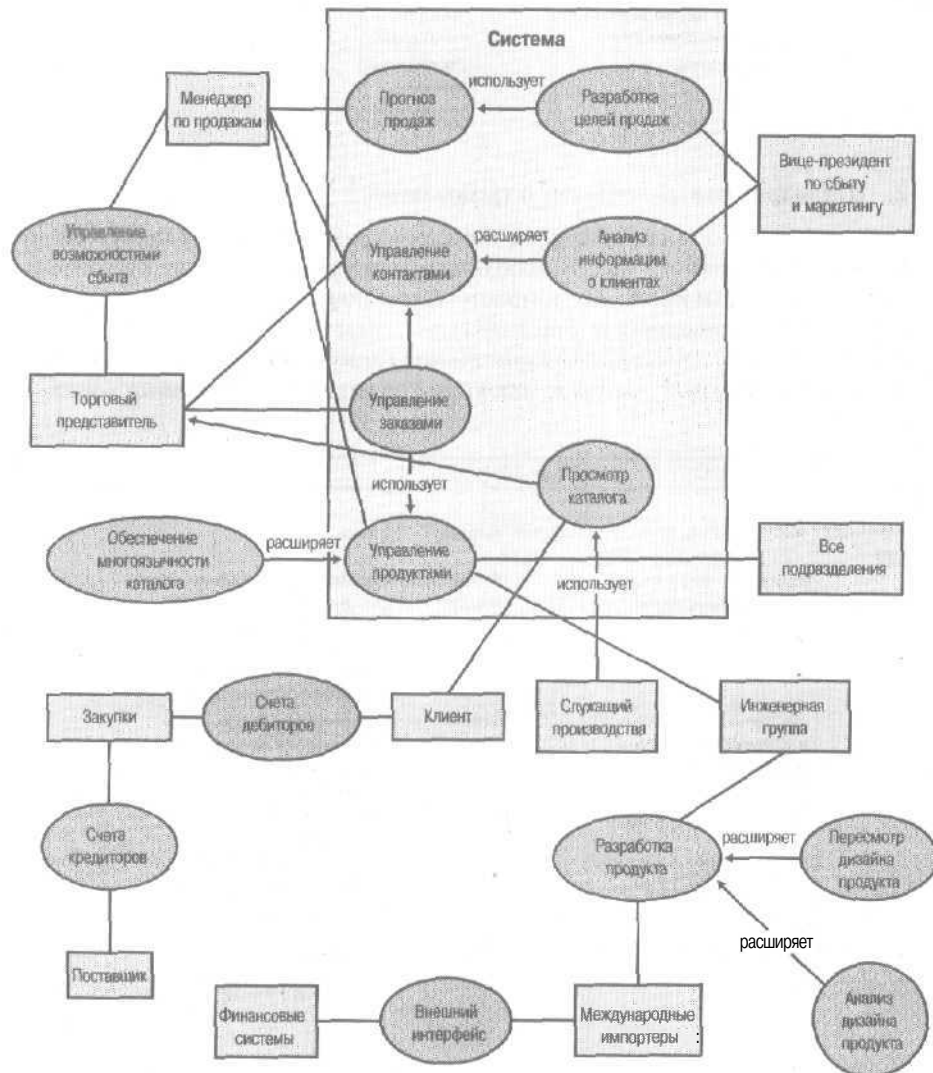


Рис. 2-1. Диаграмма варианта использования системы

Пример предварительного документа о требованиях

На рис. 2-2 показан список предварительных требований, который составлен на основе результатов интервью и схемы использования системы.

	A	B	C	D	E	F	G	H
1							1	
2		- Идентификатор запроса	Описание требования	Приоритет	Источник	Идентификатор ВИС	Текущая функция	Дополнительный столбец
3		И	Возможность синхронизации с онлайн-приложениями для записи накопленной за день информации		Региональный менеджер по продажам	x		
4		2	Извлечение данных о клиентах и возможность просматривать данные под различными углами зрения		Региональный менеджер по продажам	x		
5		3	Определение лучших клиентов		Региональный менеджер по продажам			

Рис. 2-2. Предварительный документ о требованиях

Анализируя информацию, вы скорее всего обнаружите, что вам требуется дополнительная информация. Число необходимых итераций зависит от требований бизнеса, сложности проблемы и прочих факторов, в числе которых временные ограничения, определенные участниками бизнеса.

На рис. 2-3 показан пример предварительного документа о требованиях с дополнительными вопросами, которые задаются клиентам для уточнения конкретных требований.

	A	B	C	D	E	F
1						
2		Идентификатор запроса	Описание требования	Приоритет	Источник	Вопросы
3		1	Возможность синхронизации с онлайн-приложениями для записи накопленной за день информации		Региональный менеджер по продажам	Какая конкретно информация собрана? Что значит «синхронизироваться» и с чем необходима синхронизация? Что изменяется после синхронизации? Какая особая информация действительно необходима и в какое время дня/недели?
4		2	Извлечение данных о клиентах и возможность просматривать данные под различными углами зрения		Региональный менеджер по продажам	Каким конкретно образом следует просматривать данные о клиентах? Что вкладывается в понятие «данные о клиентах»?
5		3	Определение лучших клиентов		Региональный менеджер по продажам	Каков критерий лучшего клиента? Доход? Доля дохода? Частота размещения заказов? Определенный набор заказываемых товаров?

Рис. 2-3. Дополнительные вопросы для сбора требований

Внутренняя документация проектной команды

После завершения сбора информация и в процессе анализа команда создает внутренние документы, которые, как правило, не попадают в руки заказчика, — каталоги действующих субъектов и бизнес-правил, а также словарь. Это «живые» документы, которые уточняются на протяжении жизненного цикла проекта.

Действующий субъект

Каталог действующих субъектов (actors catalog) содержит информацию обо всех участниках ВИС. (*Действующий субъект* — это сущность-объект, взаимодействующий с системой.) Каталог действующих субъектов содержит следующую информацию:

- имя субъекта;
- его обязанности;
- источник этой информации,

Например, в примере с компанией Adventure Works Cycles торговый представитель является действующим субъектом, в чьи обязанности входит установление контактов с клиентами в регионе и сбор сведений о продуктах и их спецификациях и информации обо всех клиентах. Источник этой информации отслеживается, чтобы в дальнейшем можно было вновь обратиться к нему за дополнительными сведениями или для уточнения полученных данных.

На рис. 2-4 показан пример каталога действующих субъектов.

	A	B	C	D	E
1					
2		Действующий субъект	Обязанности	Источник	Роль в бизнесе
3		Торговый представитель	Установление контактов с клиентами Знание продуктов и их...	Менеджер по продажам	

Рис. 2-4. Каталог действующих субъектов

Каталог бизнес-правил

Каталог бизнес-правил — это «живой», постоянно обновляемый документ, в котором перечислены бизнес-правила решения. Он предшествует документу, в котором излагаются требования к системе. Каталог бизнес-правил создается на ранней стадии процесса проектирования, когда проектная команда занимается сбором информации. Этот документ, как и каталог действующих субъектов, предназначен только для проектной команды и, как правило, не передается заказчику.

Каталог бизнес-правил содержит следующие поля с информацией о бизнес-правиле:

- номер, служащий для его идентификации и отслеживания;
- короткое наименование;
- описание;
- источник;
- ВИС или СИС, к которым может относиться данное бизнес-правило, или другое бизнес-правило, с которым связано данное;
- существующий набор функций, отражающий функции существующей системы, которые связаны с данным бизнес-правилом.

Например, бизнес-правило, определенное в ходе интервью и бесед с менеджером компании Adventure Works Cycles, гласит, что торговый представитель вправе дать своим клиентам скидку до 10%. Однако полномочия менеджера по продажам предусматривают право на предоставления скидки до 20%.

На рис. 2-5 приведен пример каталога бизнес-правил.

	B	C	O	F	F	G
1						
2	Идентификатор бизнес-правила	Наименование бизнес-правила	Бизнес-правило	Источник	ВИС/СИС	Существующий набор функций
3	100	Скидка	Торговый представитель вправе предоставить скидку не более 10%	Менеджер по продажам	UC5.01	
4						
5						

Рис. 2-5. Каталог бизнес-правил

Словарь

При проектировании бизнес-решения в команде используются строго определенные термины. *Словарь* содержит список этих терминов и их значение. Он необходим для того, чтобы все использовали одинаковые термины и каждый понимал, что они означают.

Занятие 3. Использование нотаций моделирования

Модели применяются для описания бизнес-процессов, понимания текущего состояния предприятия и для моделирования бизнес-процессов, которые пока не существуют, но которые планируется создать в будущем. Модели помогают наглядно представить бизнес-процессы и их связи с другими процессами. Они также отображают задачи, из которых состоит рассматриваемый процесс. На этом занятии вы узнаете о преимуществах, которые дают модели, и о различных типах применяемых нотаций моделирования.

Примечание Для создания диаграмм на этом занятии применяется Microsoft Visio® for Enterprise Architects.

Изучив материал этого занятия, вы сможете:

- ✓ перечислить преимущества моделирования;
- ✓ рассказать о роли языка моделирования UML в концептуальном проектировании;
- ✓ описать назначение различных UML-представлений;
- ✓ описать назначение различных UML-диаграмм;
- ✓ описать роль ORM в концептуальном проектировании;
- ✓ описать связь между UML-представлениями и различными этапам модели процессов MSF.

Продолжительность занятия — около 20 минут.

Преимущества моделирования

Модели применяются для описания как существующего, так и предлагаемого решения, призванного удовлетворить требования бизнеса. Вот лишь некоторые из преимуществ моделей:

- обеспечивают общую терминологию, с помощью которой описываются как существующие, так и предлагаемые решения;
- позволяют описывать сложные проблемы в виде простой структуры и значительно облегчают обмен информацией;
- модели помогают прийти к соглашению, облегчая проектной команде понимание потребностей бизнеса, требований пользователей, и того, какую информацию необходимо собрать.

Как только бизнес-процессы смоделированы и изменены в соответствии с требованиями, можно переходить к построению модели архитектуры, описывающей конечное бизнес-решение. Наиболее часто применяются две нотации моделирования:

- UML;
- ORM.

Что такое UML

UML (Unified Modeling Language) — стандартный язык, применяемый для моделирования информационных систем различной сложности — от крупных корпоративных ИТ-систем до распределенных систем, основанных на Web.

Создатели UML стремились предоставить пользователям стандартный визуальный язык, позволяющий разрабатывать понятные модели и обмениваться ими. UML не зависит от конкретных языков программирования и процессов разработки и применяется для:

- визуализации программной системы набором строго определенных символов. Разработчик приложения может однозначно интерпретировать UML-модель, созданную другим разработчиком;
- описания спецификаций информационной системы. UML помогает строить точные, однозначные и полные модели;
- конструирования моделей ИТ-системы, которые могут напрямую преобразовываться в текст на различных языках программирования;
- документирования моделей программной системы, выражая требования к системе на стадиях разработки и развертывания.

Примечание За дополнительной информацией об UML обратитесь к следующим материалам: «The Unified Modeling Language User Guide» Грейди Буча (Grady Booch), Айвара Джекобсона (Ivar Jacobson) и Джеймса Рамбо (James Rumbaugh) (Addison-Wesley, 1999) и «Use Case Driven Object Modeling with UML: A Practical Approach» Дуга Розенберга (Doug Rosenberg) и Кендалла Скотта (Kendall Scott) (Addison-Wesley, 1999).

Основные черты UML:

- простой, расширяемый и выразительный язык визуального моделирования;
- состоит из набора нотаций и правил моделирования программных систем различной степени сложности;
- дает возможность создавать простые, хорошо документированные и легкие для понимания модели ПО;
- не зависит как от языка программирования, так и от платформы.

Что такое UML-представления

UML позволяет разработчикам систем создавать стандартные планы любых систем и предоставляет огромное количество графических инструментов, которые применяются для визуализации и анализа системы с различных точек зрения. На основе диаграмм создают различные представления системы. В совокупности все представления системы составляют модель системы.

Модели или представления используются для наглядного изображения сложной информационной системы, причем различные аспекты информационной системы отображают в виде *UML-представлений (UML views)*. Обычно применяются следующие представления:

- **Пользовательское представление (user view)** выражает цели и задачи системы с точки зрения пользователей и их требований к системе. Это представление относится к части системы, с которой взаимодействует пользователь. Пользовательское представление также называют *представлением в виде набора ВИС (use case view)*.

- **Структурное представление** (structural view) отражает статическое или нерабочее состояние системы. Его также называют *представлением дизайна* (design view).
- **Представление поведения** (behavioral view) отражает динамическое или изменяющееся состояние системы. Его иногда называют *представление процессов* (process view).
- **Представление реализации** (implementation view) представляет структуру логических элементов системы.
- **Представление окружения** (environment view) отражает распределение физических элементов системы. Окружение системы определяет ее функции с точки зрения пользователей. Представление окружения также называют *представлением развертывания* (deployment view).

Что такое UML-диаграммы

Различные UML-представления содержат диаграммы, показывающие разрабатываемое решение с различных точек зрения. Не обязательно разрабатывать диаграммы для каждой создаваемой системы, но вы должны уметь разбираться в представлениях системы и соответствующих UML-диаграммах. Также, не обязательно использовать все диаграммы для моделирования системы. Следует выделить лишь те модели, которые позволят успешно смоделировать систему.

Применяйте перечисленные UML-диаграммы для изображения различных представлений системы:

- **диаграммы классов** (class diagrams) содержат классы и их связи. Связи (ассоциации) между классами изображаются двунаправленными соединительными линиями;
- **диаграммы объектов** (object diagrams) изображают различные объекты системы и их взаимосвязи;
- **диаграммы ВИС** (use case diagrams) показывают набор функций, который система предоставляет внешним объектам;
- **диаграммы компонентов** (component diagrams) отображают представление реализации системы. Она содержит различные компоненты системы и их взаимосвязи, такие, как исходный код, объектный код и исполняемый код;
- **диаграммы развертывания** (deployment diagram) показывают соответствие программных компонентов узлам физической реализации системы;
- **диаграммы коллективного взаимодействия** (collaboration diagrams) представляют собой набор классов и отправляемых и принимаемых ими сообщений;
- **диаграммы последовательностей** (sequence diagrams) описывают взаимодействие между классами — последовательность сообщений, которыми обмениваются классы;
- **диаграммы состояний** (state diagrams) описывают поведение класса в момент обращения к нему внешнего процесса или объекта. Она отображает состояния и ответные сигналы класса при выполнении действия.

Примечание Для генерации UML-диаграмм можно применять Visio. Это приложение позволяет проектировать и документировать решения, начиная от стадий начального анализа и проектирования и заканчивая развертыванием системы.

Взаимосвязь между UML-представлениями и этапами модели процессов MSF

На различных этапах модели процесса MSF создаются различные UML-диаграммы. Например, во время планирования используют набор диаграмм, изображающих предлагаемый дизайн решения, а на стадии разработки — диаграммы программных компонентов.

На рис. 2-6 показана взаимосвязь UML-представлений и соответствующих им диаграмм.



Рис. 2-6. Модель процесса MSF и UML-представления

В таблице 2-1 перечислены UML-представления и соответствующие им этапы модели процессов MSF.

Таблица 2-1. UML-представления и этапы модели процессов MSF

UML-представление	UML-диаграмма	Цель	Когда создается диаграмма	Когда используется диаграмма
Пользовательское представление	Диаграммы ВИС	Понять требования пользователей	Этап создания общей картины и этап планирования	Все этапы
Структурное представление	Диаграммы классов, диаграммы объектов	Определить базовые компоненты системы	Этап планирования	Этап разработки и этап стабилизации

Таблица 2-1. (окончание)

UML-представление	UML-диаграмма	Цель	Когда создается диаграмма	Когда используется диаграмма
Поведенческое представление	Диаграммы коллективного взаимодействия. диаграммы последовательностей, диаграммы состояний	Определить поведение системы в различных условиях и обстоятельствах	Этап планирования	Этап разработки и этап стабилизации
Представление реализации	Диаграммы компонентов	Узнать, как различные структурные блоки, * определенные в структурном представлении могут объединяться в группы	Этап планирования	Этап разработки и этап стабилизации
Представление окружения	Диаграммы развертывания	Определить физические особенности и особенности развертывания системы	Этап планирования	Этап разработки и этап стабилизации

Что такое ORM

Метод моделирования ролей объекта (Object Role Modeling, ORM) — это ориентированный на факты метод, применяемый для анализа информации в терминах объектов и ролей, которые они играют на концептуальном уровне. Эта методология позволяет документировать бизнес-правила и проектировать базы данных при моделировании сложных, связанных с обработкой данных бизнес-требований.

ORM применяется для моделирования бизнес-требований на стадии концептуального проектирования этапа планирования. Преимущество ORM заключается в концептуальном подходе к моделированию, который помогает обеспечить корректность, ясность, адаптируемость и эффективность решения. Это достигается благодаря использованию легко понимаемых человеком концепций и языка.

Качество решения зависит от его дизайна. Для создания дизайна решения строится формальная модель прикладной области приложения. В ORM эта область называется *предметной областью* (universe of discourse, UoD). В ORM используется естественный язык и интуитивно понятные диаграммы. Информация представляется в виде элементарных фактов. Наличие элементарного факта указывает, что у объекта есть свойство или что один или несколько объектов объединены связью.

Примечание Visio for Enterprise Architects поддерживает FORML (Formal Object-Role Modeling Language) — язык, связанный с ORM.

Дополнительная информация Дополнительную информацию о языках моделирования вы найдете в книге «Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design» (Реляционные базы данных: от концептуального анализа к логическому дизайну) Терри Халпина (Terry Halpin) (Morgan Kaufmann Publishers, 2001)

Процедура концептуального проектирования ORM-схемы

Процедура концептуального проектирования ORM-схемы (ORM conceptual schema design procedure, CSDP) предназначена для анализа и дизайна данных. Концептуальная схема определяет информационную структуру приложения: интересующие нас *типы фактов, ограничения* этих фактов и *правила вывода* одних фактов из других. CSDP служит для решения следующих задач:

- анализа внешней информации и представления ее в виде элементарных фактов;
- проверки популяции типов фактов. [Всеобъединенные экземпляры элемента в предметной области называются *популяцией* (population) элемента.];
- определения примитивных типов объектов в концептуальной модели;
- применения ограничений *уникальности* к концептуальной модели;
- применения к концептуальной модели ограничений *обязательной принадлежности к роли*;
- добавления к концептуальной модели ограничений на значения, на сравнение множеств и на подтипы;
- добавления к концептуальной модели круговых ограничений.

Анализ внешней информации и представление ее в виде элементарных фактов

Это наиболее важный этап CSDP. На этом этапе требуемая от системы информация представляется на естественном языке, например выходные отчеты и формы ввода в исследуемой системе. ORM-модели представляют все правомочные схемы использования данных в предметной области.

Вот некоторые важные термины, применяемые в моделировании.

- **Экземпляр** — элемент предметной области.
- **Популяция** (population) — группа всех объединенных экземпляров определенного типа элемента предметной области. В базах данных все строки таблицы составляют популяцию таблицы.
- **Множество** (set) — любая группа экземпляров. Но множество не обязательно то же, что и популяция, так как оно может быть частью популяции или комбинацией экземпляров нескольких популяций. Все популяции являются множествами, но не все множества являются популяциями.
- **Экземпляр факта** (fact instance) — одна отдельно взятая связь между двумя или более значениями данных.
- **Тип факта** (fact type) — набор экземпляров фактов с одинаковыми типами объектами и предикативными связями.
- **Тип объекта** (object type) — множество всех возможных экземпляров определенного объекта.
- **Предикат** (predicate) — глагольная группа, которая в определенной предметной области используется для связи типов объектов.

Вот пример экземпляра факта;

Писатель по фамилии Хемингуэй написал книгу «Прощай, оружие».

Здесь автор, *Хемингуэй*, и книга, *«Прощай, оружие»*, являются значениями данных, связанными глаголом действия *писать*.

В таблице 2-2 приведен другой пример — данные о преподавательском составе и учебных кафедрах университета.

Таблица 2-2. Пример таблицы с данными

Порядковый номер сотрудника	Имя сотрудника	Кафедра	Кабинет	Местный телефон	Разрешенные звонки
715	Дж. Адаме	Теория вычислительных машин и систем	69-301	2345	Местные
720	С. Бассли	Биохимия	62-406	9642	Местные
139	С. Кануто	Математика	67-301	1221	Международные
430	С. Кулп	Теория вычислительных машин и систем	69-507	2991	Международные
503	Т Д'Херс	Теория вычислительных машин и систем	69-507	2998	Местные
651	Б. Джонс	Биохимия	69-803	5003	Местные

Каждый отдельно взятый факт определяет связь между двумя объектами. Вот некоторые элементарные факты, которые можно вывести из таблицы:

- преподавателя с номером 715 зовут Дж. Адаме;
- преподаватель с номером 715 работает на кафедре «Теория вычислительных машин и систем»;
- преподаватель с номером 715 занимает кабинет 69-301;
- преподавателю с номером 715 выделен местный телефонный номер 2345.

Само название метода — «моделирование ролей объекта» — определяет способ использования в нем объектов и ролей. Объекты представляют собой либо значения, либо сущности-объекты. *Значение* (value) представляет собой символьную строку или число и определяется константой, например «Дж. Адамс» или «715». *Сущность-объект* (entity) представляет собой объект реального мира, имеющий описание, например «Преподаватель с порядковым номером 715».

Факты можно комбинировать. Например, два указанных ранее факта можно скомбинировать так;

Преподаватель с номером 715 по имени Дж. Адаме работает на кафедре «Теория вычислительных машин и систем».

Проверка популяции типов фактов

Чтобы провести проверку популяции, необходимо ввести в модель значимый пример популяции. *Значимый пример популяции* (meaningful sample population) представляет экземпляры информации предметной области и реальную задачу, которую пытается решить проектная команда. Пример популяции обычно берут из отчетов, диаграмм, графиков, экранов ввода и форм. В приведенных выше данных о преподавательском составе и учебных кафедрах каждая строка таблицы представляет один экземпляр факта. Каждая колонка таблицы представляет экземпляры роли типа объекта в типе факта.

Определение примитивных типов сущностей-объектов

Чтобы смоделировать бизнес-требования, необходимо определить типы объектов в предметной области. Типы объектов классифицируются как типы-сущности и типы-значения. **Примитивные типы-сущности** представляют базовые типы сущностей-объектов предметной области, они взаимоисключающие и исчерпывающие. Это наименьший общий знаменатель группы типов сущностей-объектов. Все примитивные типы сущностей-объектов являются атомарными и взаимоисключающим.

- **Атомарность.** Примитивный тип сущности-объекта является атомарным, поскольку не поддается разбиению на другие типы или структуры.
- **Взаимоисключаемость.** Популяции двух или более примитивных типов сущностей никогда не перекрываются. Объединение членов двух или более примитивных типов сущностей никогда не дает повторяющихся (избыточных) экземпляров.

В качестве примера примитивного типа сущности можно рассмотреть приведение типов объектов *StartTime* (время начала) и *StopTime* (время окончания) к новому примитивному типу *Time* (время)

Посмотрите на следующие типы фактов:

- *профессор* получил *степень* в *университете*;
- *старший преподаватель* получил *степень* в *университете*;
- *преподаватель* получил *степень* в *университете*.

Общий предикат этих типов фактов говорит о том, что типы сущностей — *профессор*, *старший преподаватель* и *преподаватель* можно объединить в один примитивный тип сущностей-объектов: *преподаватель*.

Применение ограничений уникальности

Необходимо явно проверить и **обеспечить** уникальность, чтобы удостовериться в правильном использовании элементарных фактов и избежать дублирования. **Ограничение уникальности** предотвращает дублирование экземпляров роли, подпадающих под это ограничение. Ограничение уникальности, распространенное на все роли в предикате, надежно **предотвращает** дублирование экземпляров типа факта. Ограничения уникальности гарантируют, что роли определены лишь *единожды*.

Преимущество ограничений уникальности заключается в том, что они:

- **предотвращают повторение фактов.** Ограничения уникальности обеспечивают неповторимость экземпляров фактов;
- **обеспечивают внутреннюю уникальность.** Ограничение уникальности на роль внутри одного предиката является внутренним ограничением, которое гарантирует, что роли (или комбинации ролей) внутри **одного** предиката встречаются в таблице только один раз. Чтобы тип факта был элементарным, все его экземпляры должны быть уникальными. Например, преподавательский состав университета состоит из профессоров, старших преподавателей и рядовых преподавателей и каждый из них специализируется в определенной области исследований. Внутреннее ограничение уникальности на типы фактов предусматривает, чтобы каждый преподаватель имел не более одного звания и не более одной специализации, работал не более, чем на одной кафедре и обладал не более, чем одним именем;
- **обеспечивают внешнюю уникальность.** Внешнее ограничение уникальности охватывает роли из одного или более предикатов и гарантирует, что экземпляры комбинаций ролей встречаются только один раз. Например, внешнее ог-

раничение уникальности требует, чтобы любая из комбинаций «название кафедры + имя преподавателя» имела отношение не более, чем к одному преподавателю.

Применение ограничений обязательной принадлежности к роли

В связях часто требуется обеспечить, чтобы все экземпляры типа объекта содержали какую-то информацию. Это достигается наложением ограничений на популяцию типа объекта. *Ограничение обязательности принадлежности к роли* гарантирует, что все экземпляры типа объекта привязаны к роли.

Ограничение обязательности роли имеет следующие характеристики:

- **глобальность.** По своей природе ограничения обязательности роли глобальны, в том смысле, что они требуют перечисления всей популяции типа объекта. Это очень важное ограничение, поскольку оно заставляет пересматривать все экземпляры на предмет включения во все прочие роли, а которых состоит тип объекта. Например, моделируя обязательное ограничение на имя и дату рождения сотрудника, необходимо позаботиться о наличии обоих элементов для всех сотрудников;
- **подразумеваемые функциональные зависимости.** Одна роль функционально зависит от другой, если есть связи «многие-к-одному» или «один-к-одному» между ролями и если на первую роль в связи обязательно наложено ограничение уникальности. Например, если каждый родитель непременно должен иметь потомка и потомки функционально зависят от родителей, то все потомки должны обязательно иметь родителей.

Ограничение обязательности роли в нотации FORML выражается включением слова *Each* (каждый) перед типом объекта, например:

Каждый человек имеет имя

В FORML-выражении слово *Each* показывает, что роль *Person* в данном типе фактов имеет ограничение обязательной принадлежности к роли и что имя каждого экземпляра типа объекта *Person* заведомо известно.

Ограничения на значение, сравнение множеств и подтипы

Ограничения используются для ограничения популяции. ORM обеспечивает различные методы ограничения области типа объекта. *Область* (domain) популяции представляет все возможные значения, существующие в популяции. Не обязательно использовать все члены области.

Например, на объект можно наложить ограничение, разрешающее хранить в нем только числа из диапазона 1—5 или только названия дней недели.

Ограничения на значение

Это ограничение предусматривает лимитирование популяции типа объекта определенной областью допустимых значений. Выражение ограничения на значение в FORML следует за шаблоном типа, затем идет фраза *The possible values of* (возможные значения), указывается тип объекта, а далее ставится двоеточие, слово *are* (есть) и перечисляются значения. Следующее FORML-выражение определяет ограничение на значения объекта *Person*, которое сужает область типов объекта до значений *Jeff*, *Maria* и *Pierre*.

Человек(имя) принадлежит к типу сущность-объект.

Каждый человек идентифицируется одним индивидуальным именем.

Возможные значения имени: «Jeff», «Maria», «Pierre».

Ограничения на множества

Часто существуют связи между популяциями различных типов фактов. Для отображения таких связей в ORM используются ограничения на множества. *Ограничения на множества* лимитируют множество экземпляров, которые могут принадлежать типу фактов. Ограничения множеств являются внешними и распространяются на роли в двух различных типах фактов. При ограничении роли в двух типах фактов лимитируются экземпляры фактов в каждом из типов фактов. Ограничения на множества ограничивают две популяции в их связи между собой. Вот некоторые типы ограничений множеств:

- **Ограничение на включение множеств** запрещает наличие экземпляров одного множества в другом множестве:
Ни один *сотрудник*, получающий *почасовую оплату*, не должен получать *зарплату*.
- **Ограничение на равенство множеств** требует, чтобы все экземпляры одного множества присутствовали в другом:
Сотрудник получает *зарплату* тогда и только тогда, когда он работает в качестве *менеджера по продажам*.
- **Ограничение на подмножество** лимитирует экземпляры множества, допуская только те, которые присутствуют в другом множестве:
Если *сотрудник* работает в качестве *менеджера по продажам*, то он одновременно является *продавцом*,

Подтипы сущностей-объектов

Иногда требуется особым образом управлять частью популяции типа объектов, отличным от того, который применяется ко всей популяции. Это подмножество популяции может присутствовать в различных типах фактов. Кроме того, вы должны разбить подмножество группы объектов на более мелкие группы, чтобы лучше понимать их или собирать о них дополнительную *полезную* информацию.

Например, подобное разбиение популяции типа объектов *Vehicle* (Автомобиль) может вылиться в два типа объектов — *GasVehicle* (Автомобиль на бензине) и *DieselVehicle* (Дизельный автомобиль). Возможно дальнейшая детализация: *ConsumerVehicle* (Легковой автомобиль) и *CommercialVehicle* (Грузовой автомобиль).

Подтип сущности-объекта — это тип объекта, содержащийся в другом типе объекта. Подтипы должны быть четко определены в терминах отношений их надтипов (supertypes). Подтипы сущностей-объектов одного надтипа могут перекрываться. Необходимо иметь возможность определять подтипы и надтипы сущностей-объектов.

Выражение типа объекта в FORML снабжается комментариями, чтобы показать все подтипы *сущности-объекта*, в котором участвует данный тип в качестве родителя или потомка. Следующее FORML-выражение представляет подтип, связанный с типом объекта *Salesperson* (продавец).

Продавец является типом сущности-объекта.

Продавец в первую очередь идентифицируется по схеме идентификации *сотрудников*,

Продавец является подтипом *сотрудника*. *Сотрудник* является надтипом *продавца*.

Сотрудник является основным надтипом *продавца*.

Определение подтипа: подмножество *сотрудников*, в которое входят те, кто является *продавцами*.

Каждый *менеджер* является *продавцом*, но не каждый *продавец* обязательно является *менеджером*.

Создание круговых ограничений в концептуальной модели

Тип объекта играет только одну роль в типе факта. Когда две роли в типе факта связаны с одним типом объекта, путь от типа объекта через пару ролей и обратно замыкается (образует круг). Например, в таком типе факта: «*Человек* голосует за *человека*», есть две роли: тот, кто голосует, и тот, за кого голосуют. Однако обе роли могут принадлежать одному объекту, например: *А голосует за Л*.

Круговые ограничения контролируют популяцию ролей типа объекта, когда у типа объекта две роли *соответствующих* одному типу факта. Далее приведены некоторые примеры круговых ограничений.

- **Нерефлексивные** (*irreflexive*, сокр. *ir*) круговые ограничения *предотвращают* связь экземпляра объекта с самим собой. FORML-выражение для нерефлексивного кругового ограничения выглядит так:

Ни один объект не ссылается на себя самого.

- **Симметричные** (*symmetric*, сокр. *sym*) круговые ограничения гарантируют, что для каждого кортежа в типе факта существует зеркальное отражение. В функциональных языках программирования *кортеж* (*tuple*) — это объект данных, содержащий два или более компонента. Выражение FORML для симметричного кругового ограничения выглядит так:

Если объект *o1* связан с объектом *o2*,
то объект *o2* связан с объектом *o1*.

- **Асимметричные** (*asymmetric*, сокр. *as*) круговые ограничения гарантируют, что ни для одного кортежа в популяции типа факта не существует зеркального отражения. Выражение FORML для асимметричного кругового ограничения выглядит так:

Если объект *o1* связан с объектом *o2*,
то объект *o2* не может быть связан с объектом *o1*.

- **Антисимметричные** (*antisymmetric*, сокр. *as*) круговые ограничения гарантируют, что ни для одного кортежа в популяции типа факта не существует зеркального отражения и ни один экземпляр не связан сам с собой. Выражение FORML для антисимметричного кругового ограничения выглядит так:

Если объект *o1* связан с объектом *o2* и *o1* не тот же самый объект, что *o2*,
то объект *o2* не может быть связан с объектом *o1*.

- **Интранзитивные** (*intransitive*, сокр. *it*) круговые ограничения *устанавливают* иерархические связи между экземплярами в популяции. Выражение FORML для интранзитивного кругового ограничения выглядит так:

Если объект *o1* связан с объектом *o2* и объект *o2* связан с объектом *o3*,
то объект *o1* не может быть связан с объектом *o3*.

- **Ациклические** (*acyclic*, сокр. *ac*) круговые ограничения не позволяют, чтобы путь через цепь связей циклически возвращался к началу. Выражение FORML для ациклического кругового ограничения выглядит так:

Объект не может циклически возвратиться сам к себе через одну или несколько связей: Объект связан с объектом.

Занятие 4. Создание ВИС и СИС

Варианты использования системы (ВИС) и сценарии использования системы (СИС) фиксируют функциональные требования к системе. На этом занятии вы подробнее узнаете о ВИС и СИС и научитесь их создавать.

Изучив материал этого занятия, вы сможете:

- ✓ создавать ВИС;
- ✓ описывать СИС;
- ✓ создавать СИС;
- ✓ уточнять требования на основании ВИС и СИС.

Продолжительность занятия — около 10 минут.

Как создается ВИС

ВИС представляют собой функциональные описания осуществляемых системой операций при инициировании пользователем события или действия. Создаваемые ВИС должны представлять системные процессы, в том числе все события, которые могут возникнуть в любых возможных ситуациях.

ВИС состоят из элементов, которые расположены внутри системы и отвечают за выполнение функций и поведение системы. Это действия, выполняемые системой для получения ожидаемого пользователями результата. Такая модель позволяет участникам проекта прийти к согласию относительно возможностей и границ применения системы.

Диаграмма ВИС документирует следующие связанные с проектированием действия:

- определение системы;
- определение действующих субъектов;
- определение взаимодействий между действующими субъектами и системой;
- определение границ применения системы.

Определение системы

Система ~ это набор подсистем, решающих задачи реального мира. Например, в сценарии продаж и маркетинга система обработки заказов может иметь подсистему, предоставляющую (при наличии соответствующих условий) скидки при выписке счета-фактуры. При разработке ВИС определяют одну систему или подсистему. Набор ВИС определяет связи между составляющими систему подсистемами в дополнение к связям между взаимодействующими между собой системами,

Определение действующих субъектов

Действующий *субъект* — это составляющая часть схемы использования системы, ВИС показывает взаимодействия между действующими субъектами и системой. *Действующий субъект* — это сущность-объект, взаимодействующий с системой и созданный для конструирования определенного события. Действующий субъект может быть:

- пользователем системы;

- **сущностью-объектом**, таким, как другая система или база данных, расположенная вне системы.

Например, в компании Adventure Works Cycles действующими субъектами являются торговые представители, менеджеры по продажам, клиенты и сотрудники, занятые в производстве.

Роли действующих субъектов описывают результат ВИС и то, зачем нужен тот или иной ВИС. Ориентация на действующих субъектов помогает команде проектирования уделить основное внимание тому, как будет использоваться система, а не тому, как ее разрабатывать и реализовывать. Кроме того, этот способ позволяет уточнить и затем определить **границы** применения системы, а также **выявить** потенциальных бизнес-пользователей системы, которых следует учесть в процессе моделирования ВИС.

Определяя действующих субъектов, руководствуйтесь следующими вопросами:

- кто использует систему;
- кто запускает систему;
- кто поддерживает систему;
- какие другие системы используют данную систему;
- кто получает информацию из данной системы;
- кто предоставляет информацию для системы;
- какие операции выполняются автоматически в определенное время;
- кто (или что) инициирует события в системе;
- кто (или что) должен взаимодействовать с системой, чтобы она отреагировала на определенное событие;
- есть ли интерфейсы для получения отчетов;
- есть ли интерфейсы администрирования системы;
- необходимо ли системе взаимодействие с любыми другими имеющимися системами;
- определены ли уже какие-либо действующие субъекты системы;
- есть ли какие-либо аппаратные или программные устройства, взаимодействующие с системой, которые следует смоделировать в процессе анализа;
- должны ли какие-либо внешние сущности-объекты оповещаться о том или ином событии системы? должна ли система обращаться к внешней сущности-объекту за помощью в выполнении задачи?

Определение взаимодействий между действующими субъектами и системой

После определения системы и действующего субъекта необходимо описать взаимодействие между ними, то есть создать ВИС для каждого взаимодействия. Описывайте только те взаимодействия, которые важны с точки зрения требований бизнеса и описания **общей** картины решения.

Определение границ применения системы

Одна из наиболее трудных сторон моделирования с помощью ВИС — определение точной границы создаваемой системы. Новичкам в моделировании на основе ВИС поначалу нелегко определить, какие именно действующие субъекты должны быть частью системы.

При определении границ системы участникам команды придется **ответить** на несколько **вопросов**.

- Что произойдет с ВИС-схемами, в которых участвует данный действующий субъект?
- Кто (или что) сейчас взаимодействует с этими ВИС?
- Что будет, если обнаружатся новые требования? Станут ли они частью системы?
- Обязательны ли эти требования для системы?
- Относятся ли эти требования к тому, что система должна делать исходя из логики вещей?
- Может ли эти требования удовлетворить один из существующих действующих субъектов?
- Относятся ли эти требования к тому, что клиенты или пользователи ожидают от системы?

На рис. 2-7 показан пример диаграммы набора ВИС.

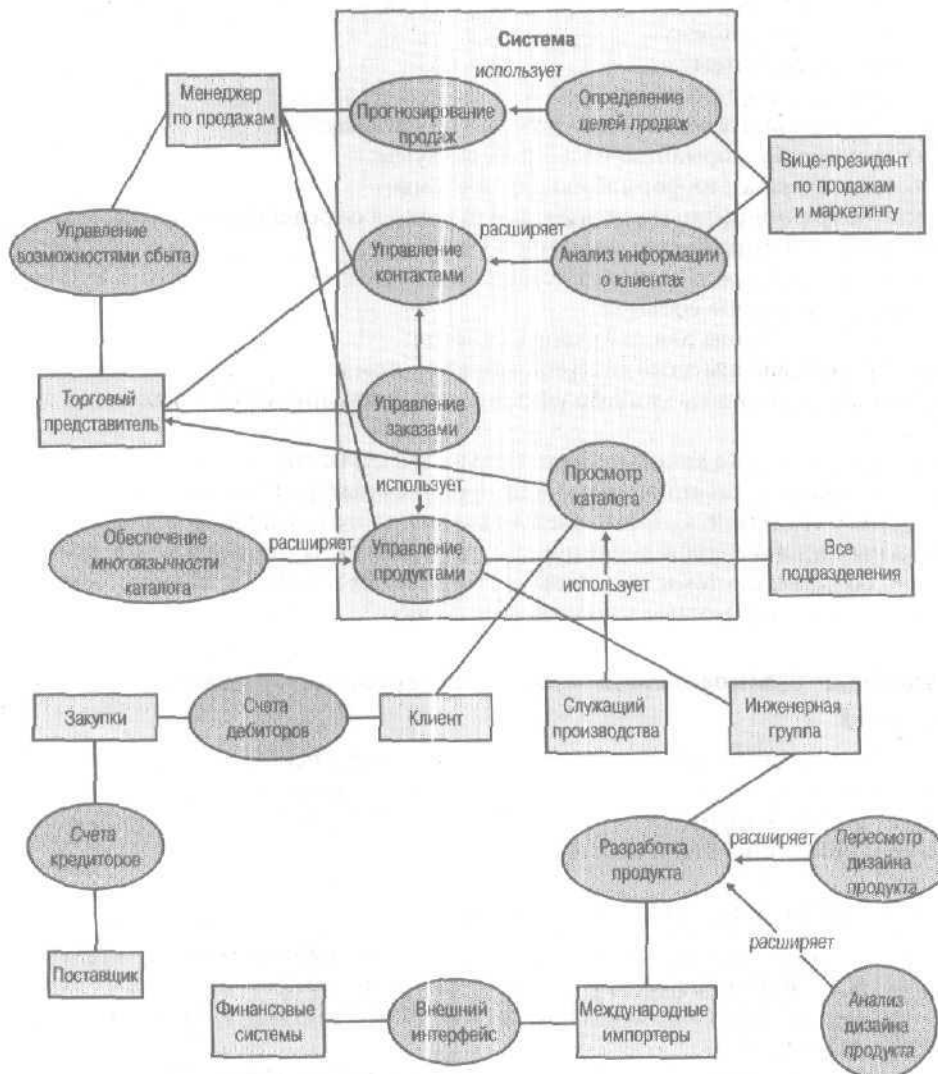


Рис. 2-7. Пример диаграммы набора вариантов использования системы

Как правило, действующий субъект изображается в виде прямоугольника, одиночный ВИС — в виде эллипса, а набор ВИС заключается в рамку, обозначающую систему.

Что такое ВИС

Варианты использования системы описывают высокоуровневое взаимодействие между действующим субъектом и системой. Группы ВИС детально описывают технологический процесс. ВИС предоставляют дополнительную информацию о действиях и последовательностях задач, составляющих процесс. СИС документируют последовательность задач.

Сценарии использования системы (СИС) детально описывают конкретный экземпляр ВИС. Полное документирование варианта использования системы требует создания большого количества сценариев использования. Если варианты использования системы являются диаграммами, то СИС представляют собой текстовый документ-рассказ.

СИС изображают объекты в технологической цепочке. Объекты изменяются системой и сами влияют на систему, кроме того их следует принимать во внимание для обеспечения нормальной работы системы. Например, в учебном центре объектами являются клиент, учебный курс и торговый представитель. Объекты обеспечивают представление характеристик и поведения элементов предметной области бизнеса. На этапе концептуального проектирования создаются СИС, отображающие объекты предметной области.

Кроме объектов, сценарии использования описывают исключения. *Исключение* (exception) — это нетипичное событие или нестандартный порядок выполнения задач в ВИС. Пример возникновения исключения: при вводе информации о новом клиенте в систему учета контактов учебного центра система оказывается нерабочей и торговый представитель должен прибегнуть к другим способам получения информации о клиенте.

Обработывая исключения в системе, придерживайтесь следующих рекомендаций:

- задавайте вопросы типа «что, если», чтобы выявить исключения, возможные при выполнении задачи или шага;
- определяйте относительную вероятность возникновения каждого исключения;
- обсудите имеющиеся на текущий момент стандартные и все возможные альтернативные методы обработки исключений;
- включите обработку исключений, отличающихся высокой вероятностью, в дизайн решения.

При разработке СИС вы наверняка столкнетесь с задачей, которую можно трактовать как ВИС. Например, в варианте использования системы «Запись клиента на курс» задачу «Обработка платежа клиента» можно выделить в ВИС высокого уровня, который является частью последовательности операций и имеет несколько сценариев использования.

Следовательно, каждый СИС «Запись клиента на курс» заканчивается операцией «Ввод номера курса». Затем создаются все значимые сценарии использования для нового ВИС «Обработка платежа клиента».

Определяются варианты использования системы, которые соответствуют цепочкам операций, а затем разрабатываются сценарии использования и описываются последовательности задач для каждого ВИС. По сценариям использования системы можно определить требования текущего состояния.

Почему следует создавать СИС для текущего состояния

После определения ВИС переходят к определению СИС, которые возможны для конкретных вариантов. Полученная от пользователей информация необходима для описания других возможных в данном ВИС сценариев использования системы. Создание СИС помогает определить, достаточно ли информации собрано. Сбор требуемого объема сведений для описания текущего состояния является частью итеративного процесса сбора и анализа информации.

Вы вправе создать сценарии как для текущего, так и для будущего состояния бизнес-окружения. *Сценарий текущего состояния* показывает, как выполняются бизнес-операции сейчас, а *сценарий будущего состояния* представляет действия так, как того требует бизнес заказчика. В обоих видах сценариев делается акцент на бизнес-процессы, информацию, пользователей и задачи.

В некоторых ситуациях **полные** сценарии необходимо создавать только для ВИС, имеющих много зависимостей или иницирующих большое число исключений. Это позволяет проектной команде **добиться** баланса между **стоимостью** и потенциальной выгодой. ВИС и СИС следует разрабатывать итеративно, причем их создают или **расширяют** во время разработки начального набора ВИС.

Преимущества

Хотя известно множество способов анализа существующих рабочих процессов, варианты и сценарии использования системы особенно эффективны в процессе моделирования. Их преимущества таковы:

- измеряя производительность **существующей** системы, проектная команда может определить, удовлетворяет ли новая система требованиям *удобства использования* (usability);
- команда может определить, что нормально работает в системе и где возможны проблемы;
- команда подчас **обнаруживает**, что на самом деле проблемы совсем не в том месте, где указывают пользователи, и вызваны они иными **причинами**. Это позволяет команде сконцентрировать свои усилия на реальных (а не мнимых) недостатках.

Создание сценария использования системы

Для создания СИС необходимо выполнить следующие задачи:

- определить предварительные условия СИС, а также информацию или условия для выполнения сценария;
- определить постусловия СИС, то есть выполненную работу или достигнутую в процессе выполнения цепочки операций цель;
- разбить действие на дискретные шаги;
- определить исключения, которые могут возникнуть на каждом шаге. Вам может понадобиться разработать СИС для этих исключений;
- определить требование, к которому относится данный СИС, для последующего контроля и возможности трассировки;
- определить источник СИС для последующего обсуждения и уточнения.

Ниже приведена иллюстрация создания СИС на основании ВИС на примере компании Adventure Works Cycles.

Название ВИС: Клиент запрашивает литературу по продукту

Сокращенное название: Клиент запрашивает литературу по продукту,

Идентификатор ВИС: UC05.1

Идентификатор требования: 14.1

Описание: Клиент желает получить информацию по продукту. Клиент имеет возможность просматривать его копию в онлайн-режиме, напечатать копию или запросить доставку по почте печатной копии.

Действующие субъекты: Клиент

Предварительные условия: Клиент имеет доступ к Интернету.

Клиент посетил Web-сайт Adventure Works Cycles.

Клиент щелкнул ссылку «Просмотр описаний продуктов».

Информация о продукте есть в базе данных, и сайт нормально функционирует.

Основной вариант развития событий:

Исключения

- | | |
|--|--|
| 1. Клиент переходит к описанию требуемого продукта. | В данном представлении продукт не присутствует. |
| 2. Клиент выбирает продукт, щелкнув его изображение или ссылку «Подробности». | |
| 3. Клиент просматривает общую информацию о продукте (наименование, описание, изображение, наличие на складе, цена, номер изделия). | Какие-либо сведения недоступны. |
| 4. Клиент щелкает «Требуется полное описание» | |
| 5. Клиент просматривает полное описание в режиме онлайн (UC05.1.3). | Клиент выбирает печатную версию (UC05.1.2). |
| 6. Клиент щелкнул «Получить по почте брошюру с полным описанием» (UC05.1.1). | Клиента нет в базе данных. Клиент завершает работу и сохраняет свой профиль. |
| 7. Клиент есть в базе данных. Клиент подтверждает свой адрес. | |
| 8. Клиент щелкает ссылку «Отправить запрос». | Процесс отправки запроса завершается с ошибкой. |
| 9. Клиент уходит со страницы описания продукта. | |

Постусловия: Запрос на получение полной спецификации по почте завершен и сохранен в базе данных.

Нерешенные задачи: Каким образом лучше всего поставить запрос в очередь на исполнение?
Если процесс отправки запроса трижды завершился с ошибкой, какой процесс следует вызвать?

Автор документа: Майк Дансеглио

История изменений:

Дата: 6 ноября 2002 г.

Автор: Хейди Стин

Описание: Исходная версия

Примечание Несмотря на то, что здесь показано, как создавать детальный ВИС, подобные подробные СИС создаются лишь на этапе концептуального дизайна. Этот пример демонстрирует, как создавать сценарии использования.

Как уточняются требования

Как уже говорилось, анализируя информацию при помощи вариантов и сценариев использования системы или любых других инструментов, уточняют список требований. При создании более детализированных ВИС из собранной информации выделяют требования, пожелания и ограничения, а также определяют любые скрытые требования. Требования и пожелания в конечном итоге определяют функции готового решения. На этой стадии команда только описывает, организует и расставляет требования и пожелания по приоритетам. Позднее, в процессе разработки продукта команда разработки определит его функции.

Примечание Требования уточняются, как правило, на этапах создания общей картины решения и в процессе создания концептуального и логического дизайна. Требования первого уровня создаются на этапе создания общей картины решения, а затем они детализируются, в конечном итоге вы получите набор функций и определите, какие из требований следует «закрыть» создаваемой версией продукта. Если вы добрались до физического дизайна, при любых изменениях в требованиях придется выполнить дополнительную итерацию. Вот почему следует договориться с заказчиком о требованиях на этапе концептуального и логического дизайна.

Требования и пожелания

Требования отражают характеристики процесса, жизненно важные для достижения бизнес-целей, а пожелания важны, но не критичны для таких целей или для решения бизнес-проблем. Пожелания основываются на реальной каждодневной работе людей. Однако они есть не что иное, как представление о возможном идеальном состоянии вещей с точки зрения пользователей. В процессе обсуждения требований и пожеланий с заказчиком некоторые пожелания превратятся в требования, и наоборот, часть требований окажутся не столь важными и попадут в группу пожеланий.

Ограничения и предположения

Ограничения определяют параметры, которым должно удовлетворять готовое бизнес-решение. Это особенности бизнес-среды, которые не могут (или не будут) изменяться. Часто ограничения становятся целями дизайна приложения. Если ограничения не определить должным образом, созданное решение может оказаться непригодным для внедрения и использования на предприятии.

Вот некоторые примеры ограничений, которые необходимо задокументировать:

- бюджетные ограничения;
- характеристики существующих или вспомогательных систем;
- архитектура сети;
- требования по безопасности;
- операционные системы;
- запланированная технологическая модернизация;
- ограничения полосы пропускания сети;
- соглашения и структуры поддержки и сопровождения;
- уровень знаний разработчиков и персонала службы поддержки;
- возможности обучения пользователей.

Предположения выясняются во время бесед с заказчиком и анализа собранных таким образом данных, причем они во многом схожи с ограничениями. Вот пример предположения: «В качестве базовой технологии нового решения мы будем использовать Microsoft .NET». В этом случае команда заранее предполагает, что решение будет создаваться на основе .NET.

Ограничение что-то запрещает, а предположение — это информация, которая задана до начала проекта, причем она может быть как «хорошей», так и «плохой».

Скрытые требования

Особенно важно определить скрытые требования. Представьте себе, вы завершили проект и вдруг узнаете, что компания-заказчик в ближайшем будущем планирует поглотить другую компанию, а вам придется обеспечить взаимодействие своей системы с системами приобретенной компании. Заказчик может посчитать нужным скрыть эту информацию, не представляя себе, как подобное решение скажется на проекте. Вот примеры скрытых требований:

- возможность взаимодействия с одноранговыми сетями и Интернетом;
- отсутствие брандмауэра для обеспечения беспрепятственного подключения;
- слияние, поглощение или иные изменения в структуре бизнеса;
- удовлетворение требований законов, принятых после начала работы над проектом;
- поддержка развернутой системы по истечении оговоренного периода;
- текучка кадров, способная затронуть проект или команду.

Практикум. Сбор и анализ информации



При выполнении упражнений по сбору информации используйте знания, полученные занятиях.

Упражнение 1. Подготовка к интервью

Вы собираетесь проинтервьюировать менеджера по персоналу компании Adventure Works Cycles, чтобы получить информацию о существующей корпоративной системе. В качестве подготовки к интервью сформулируете вопросы, которые считаете важными и уместными?

По завершении работы над списком вопросов посмотрите документ «Интервью с менеджером по персоналу» (*Interview with the Human Resources Manager.doc*), который хранится в папке `\Solution Documents\Chapter01` прилагаемого к книге компакт-диска и подумайте, какие еще вопросы потребуется задать менеджеру на этом и последующих дополнительных интервью.

Ниже перечислены некоторые стандартные вопросы.

- Сколько сотрудников ведет кадровая служба в настоящее время?
- Какие имеются категории сотрудников (постоянные сотрудники, совместители, контрактники)?
- Сколько новых сотрудников планируется принять в следующем финансовом году?
- От каких недостатков существующего решения вам бы хотелось избавиться в новом решении?
- Опишите, как в настоящее время происходит процесс найма.
- Расскажите, какая сейчас действует процедура наблюдения за новыми сотрудниками.
- Опишите, как в настоящее время анализируется распределение поощрений и как результаты анализа влияют на будущие решения о поощрениях.
- Сколько документов вы храните по каждому сотруднику? по кандидату?
- Какие категории вы используете для управления сотрудниками и кандидатами (навыки, уровни, отраслевые термины или специальности)?
- Все ли документы в настоящее время хранятся в формате Microsoft Word? Если нет, то какие еще форматы используются?
- Где вы храните свои документы? (Перечислите все сетевые диски и базы данных, включая имена серверов.)
- Кто наделен полномочиями и возможностью просматривать, изменять или добавлять записи сотрудников или кандидатов?

Упражнение 2. Создание вариантов использования системы для проекта автоматизации продаж и расширения Web-проекта

Ваша команда вчера завершила сбор информации о проекте Adventure Works Cycles. На основании интервью с региональным менеджером по продажам, ИТ-менеджером, вице-президентом по производству и клиентом, использующим Web, а также отчета о наблюдении за действиями партнера по продажам создайте как можно больше ВИС. На этом этапе требуется создавать только варианты использования системы. Полный сценарий вы создадите в одном из следующих упражнений. Разработайте варианты использования системы в формате «Действующий субъект — Действие — Объект», например, так: «Торговый представитель просматривает информацию о продукте».

Для создания UML-диаграмм вариантов использования системы вам пригодится Visio.

Одно из возможных решений этого упражнения вы найдете в файле *C02Ex2_Answer.vsd* в папке `\SolutionDocuments\Chapter02` на компакт-диске.

Упражнение 3. Разработка предварительных требований на основе первичного сбора информации

Вы начата сбор первичной информации с разговора с сотрудниками отдела продаж, администратором сети и инициатором проекта со стороны заказчика. У вас также есть наброски вариантов использования системы. Найдите фразы в интервью или вариантах использования системы, которые в будущем могут стать требованиями.

Воспользуйтесь документом *C02Ex3.xls*, расположенном в папке `\SolutionDocuments\Chapter02` на компакт-диске. На вкладке *Original* записывайте конкретные фразы из интервью. Чтобы упростить работу, на вкладке *Revised* приведены три примера. Одно из возможных решений задания вы найдете в файле *C02Ex3_Answer.xls* в той же папке. Вкладка *Original* содержит фразы из интервью, которые практически являются требованиями, на вкладке *Revised* показаны переформулированные фразы, начиная с комментариев менеджера по продажам. Они собраны по темам. В документе также перечислены вопросы, которые основаны на первых трех потенциальных требованиях, выраженных менеджером по продажам.

Упражнение 4. Разработка сценария использования системы

Создайте сценарий для варианта использования системы «Клиент запрашивает литературу по продукту». В первую очередь вам понадобится определить действующий субъект, объект и систему. Затем укажите предварительные и постусловия. Далее закончите последовательность задач и укажите возможные исключения. В процессе работы кроме привычных знаний о разработке ПО и опыта совершения покупок в режиме «онлайн» используйте интервью и документы о наблюдении за поведением пользователей.

Результаты работы записывайте в Excel-документе *C02Ex4.xls* из папки `\SolutionDocuments\Chapter02`. Одно из возможных решений этого упражнения приведено в документе *C02Ex4_Answer.xls* в той же папке.

Резюме

- Сбор и анализ информации — это первые этапы создания модели процесса в Microsoft Solution Framework.
- Сбор и анализ — итеративные процессы, в которых вы должны работать совместно со своими заказчиками.
- К методам сбора информации относятся интервьюирование, наблюдение за пользователем, обучение, проводимое пользователями, и создание прототипов.
- Анализ подразумевает создание вариантов и сценариев использования системы, предварительного списка требований, каталога действующих субъектов и каталога бизнес-правил.
- Моделирование — это один из методов описания бизнес-процессов. Модели показывают связи между бизнес-процессами и их поведением и служат дополнением к заданиям, из которых состоят процессы.
- ORM — это ориентированный на факты метод анализа информации на концептуальном уровне.
- UML — это стандартный язык моделирования, применяемый для моделирования программных систем самой различной степени сложности.

- **UML-представления** отображают различные аспекты программной системы на основании **UML-диаграмм**.
- **UML-диаграммы** отображают различные представления системы.
- Создание вариантов использования системы включает определение:
 - системы;
 - действующих субъектов;
 - взаимодействий между системой и действующими субъектами;
 - границ системы.
- ВИС описывают взаимодействие между действующими субъектами и системой и используются для описания последовательности технологических операций.
- ВИС предоставляют информацию о деятельности и последовательностях заданий, из которых состоит процесс.
- В процессе анализа необходимо устранить дублирование и определить наиболее важную для бизнеса и решения бизнес-проблем информацию.
- Выполняя синтез информации с точки зрения бизнеса и отдельных пользователей, необходимо четко различать требования и пожелания.
- Ограничения определяют параметры системы, которым должно удовлетворять конечное бизнес-решение и отображают особенности бизнес-среды, которые невозможно изменить.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. В чем отличие методов группового и индивидуального интервьюирования?
2. Когда при сборе информации следует предпочесть создание прототипов наблюдению за пользователями?
3. Как выбрать наиболее эффективный метод сбора информации в конкретном проекте?
4. Зачем создают варианты использования системы?
5. Что такое каталог действующих субъектов?
6. Что такое ORM?
7. Что такое UML?
8. Каково назначение различных UML-представлений?
9. Что представляет собой действующий субъект в ВИС?
10. Каково назначение СИС?
11. Какова последовательность создания сценария использования системы?

ГЛАВА 3

Создание общей картины решения

Занятие 1. Стадия создания общей картины решения	72
Занятие 2. Создание документа общей картины и области действия решения	77
Занятие 3. Создание документа структуры проекта	88
Занятие 4. Анализ рисков	93
Практикум. Разработка документа общей картины и области действия решения	96
Резюме	100
Закрепление материала	102

В этой главе

Успех проекта зависит от способности членов проектной команды и клиентов четко определять цели и задачи проекта. Общие концепции проекта формируются на этапе создания общей картины решения модели процессов MSF. В этой главе рассказывается об этапе создания общей картины решения, ролях и обязанностях членов команды на этом этапе. Вы также узнаете, как создается общая картина проекта и анализируются связанные с проектом риски.

Прежде всего

Для изучения материалов этой главы необходимо иметь общее представление о:

- моделях процессов в MSF;
- методах сбора и анализа информации.

Занятие 1. Стадия создания общей картины решения

На этой стадии команда, заказчик и спонсоры проекта определяют высокоуровневые бизнес-требования и общие цели проекта. Главная задача — согласовать то, как видят проект разные его участники, и выработать у членов команды единое мнение о полезности проекта для компании и его реализуемости. На этой стадии основное внимание уделяется четкости формулировки задач. Стадия создания общей картины решения считается завершенной по достижении контрольной точки «Утверждение документа общей картины и области действия проекта».

Изучив материал этого занятия, вы сможете:

- ✓ описать цель этапа создания общей картины решения;
- ✓ описать роли и обязанности членов команды на этом этапе;
- ✓ перечислить рекомендации по созданию проектной команды.
- ✓ описать результаты этапа создания общей картины решения.

Продолжительность занятия - около 5 минут.

Цель создания общей картины решения

На первом этапе модели процессов MSF команда создает общую концепцию бизнес-задачи, которую придется решать, а также описывает ее связи с бизнесом, клиентами и средой. Это позволяет команде получить четкое представление о решении, которое предполагается создать. Обязательное условие успеха проекта — ясное понимание потребностей и нужд клиента.

На рис. 3-1 показано место этапа создания общей картины решения в модели процессов MSF.

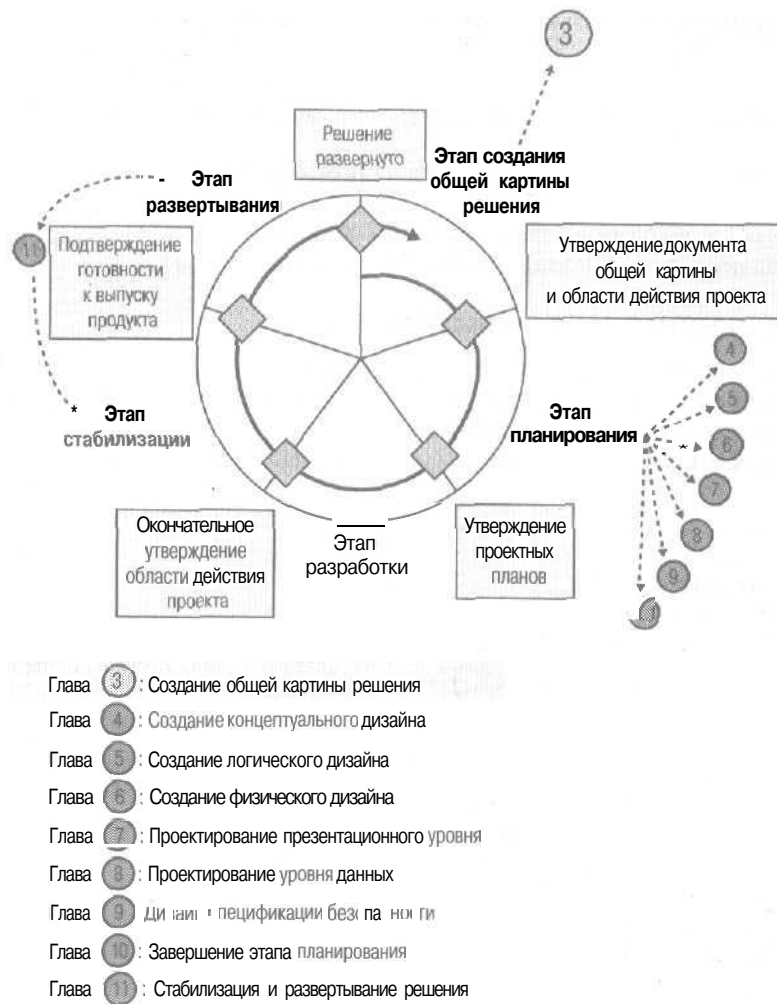


Рис. 3-1. Где мы находимся в процессе проектирования в MSF

Допустим, клиент заказал у вас Web-сайт. Фразы «нам нужен Web-сайт» достаточно, чтобы начать проект, но она не дает ничего для формулировки концепции проекта Web-сайта. Чтобы создать максимально эффективное решение, проектная команда должна определить цели, которые преследует клиент, заказывая Web-сайт. Кроме того, следует выяснить, необходимы ли сайту какие-то уникальные характеристики, которые позволят повысить конкурентоспособность компании. Что нужно компании: сложный сайт или такой, чтобы справился с растущим числом пользователей, а может, необходимо удовлетворить оба требования — по сложности и по масштабируемости? Проектная команда должна учесть целевой рынок и ожидаемую интенсивность обращений к сайту. Еще один параметр, который следует иметь в виду, — торговая марка и имидж компании на рынке.

Создание общей картины решает множество целей. Оно позволяет:

- определить цели и ограничения проекта;

- решить вопросы реализуемости решения, заручиться одобрением участников проекта и выработать единые ожидания заинтересованных лиц в отношении проекта;
- создать фундамент, на котором члены команды в дальнейшем построят решение;
- определить область действия проекта, которая поможет при детальном планировании следующего этапа;
- оценить ресурсы, необходимые для разработки решения;
- определить и планировать основные контрольные точки проекта.

Роли и обязанности членов команды

Хотя проектная команда работает над созданием общей картины и определением области действия проекта как единый организм, каждый ее член выполняет свою задачу на каждом этапе жизненного цикла проекта.

На рис. 3-2 показаны роли проектной команды на этапе создания общей картины решения.



Рис. 3-2. Модель команд MSF

В модели команд MSF предусмотрены шесть ролей.

- **Менеджер решения** (product management) отвечает за корректную реализацию требований заказчика. Менеджер решения также работает совместно с менеджерами над разработкой единого образа проекта. Для этого он изучает и анализирует бизнес-задачи, бизнес-требования, общую картину решения, бизнес-цели и профили пользователей.
- **Менеджер программы** (program management) определяет цели проекта, критерии и метрики успеха, формулирует концепцию решения и формирует инфраструктуру проекта.
- **Разработчики** (development) обеспечивают команде информацию о технических следствиях разработки продукта и реализуемости решения.
- **Специалисты по удобству использования** (user experience) анализируют потребности и проблемы, возникающие у пользователей, и оценивают продукт на предмет соответствия таким потребностям.

- **Тестировщики** (testing) предоставляют проектной команде информацию о достижении целей по качеству решения и определяют действия, которые необходимы для достижения желаемого уровня качества.
- **Менеджер по выпуску** (release management) определяет, что необходимо для развертывания продукта, как и когда он будет развертываться, а также требуется ли для развертывания дополнительная инфраструктура.

Организация проектных команд

Создание команды — одна из задач, выполняемых на этапе создания общей картины решения. Очень важно, чтобы члены проектной команды обладали требуемыми знаниями и навыками и могли решать задачи, необходимые для создания решения. Наличие экспертного опыта является залогом успешного выполнения проекта. Оценивая пригодность того или иного сотрудника для включения в команду, учитывают:

- **знания** — информацию, которой должен обладать специалист, чтобы компетентно выполнять работу, например знание основных принципов информационных технологий;
- **навыки**, которые представляют поведение или способности, определяющие компетентность в определенной области, например способность применять математическую логику или артистические качества;
- **исполнительность** — способность и прогнозируемость квалифицированной реализации задачи. Например, стоит предпочесть сотрудника, который выполняет поставленные задачи в срок, соблюдая требуемый уровень качества. Кроме способностей сотрудников **существуют** и другие параметры, которые следует учесть при выборе членов команды:
 - занятость сотрудника;
 - «стоимость» члена команды;
 - уровень допуска в системе безопасности.

Результаты этапа создания общей картины решения

Рассматриваемый этап содержит несколько промежуточных контрольных точек — организация костяка команды, создание предварительной версии документа общей картины и области действия решения и документа оценки рисков — однако этап считается завершенным при создании трех главных документов:

- **документа общей картины и области действия решения** — описывает цели и ограничения проекта. В нем перечислены параметры разрабатываемого продукта, требования, которым он должен удовлетворять, его функции и предварительный календарный график работ;
- **документа структуры проекта** — описывает организационную структуру проекта и процесс руководства проектом и определяет роли и обязанности каждого члена команды;
- **документа оценки риска** — содержит начальное **определение** и анализ рисков, связанных с проектом, а также планы мероприятий по обеспечению непрерывности бизнеса.

В зависимости от масштаба проект может **предусматривать** дополнительные результаты, в том числе:

- предварительный перечень функций будущего решения;
- предварительные требования и варианты использования системы (ВИС);

- предварительную архитектуру;
- схему графического пользовательского интерфейса.

В дополнение к документам, которые в обязательном порядке предъявляются заказчику, проектная команда разрабатывает несколько документов для внутреннего использования:

- каталог действующих субъектов;
- каталог бизнес-правил;
- словарь терминов.

Примечание Об этих документах рассказывалось в главе 2.

Занятие 2. Создание документа общей картины и области действия решения

Конечная контрольная точка этапа создания общей картины решения — утверждение документа общей картины и области действия решения. На этом занятии рассказывается, как создается такой документ.

Изучив материал этого занятия, вы сможете:

- ✓ описать содержание документа общей картины и области действия решения;
- ✓ создать формулировку задач;
- ✓ создать документ общей картины решения;
- ✓ создать профили пользователей;
- ✓ определить область действия проекта;
- ✓ создать концепцию решения;
- ✓ определить цели проекта;
- ✓ утвердить документ общей картины и области действия решения.

Продолжительность занятия — около 20 минут.

Документ общей картины и области действия решения

Этот документ представляет собой один из окончательных результатов этапа создания общей картины решения. В нем описываются цели и ограничения бизнес-решения; другими словами, это предварительное соглашение всех участников проекта относительно базовых концепций проекта. Основываясь на этом документе, проектная команда определяет высокоуровневые бизнес-цели. В самом начале работы команда принимает решение о целесообразности проекта именно на основе этого документа, а после одобрения этот документ применяется для планирования всего проекта.

В процессе создания документа общей картины и области действия решения команда проводит множество интервью с клиентами и заинтересованными лицами, анализирует высокоуровневые варианты использования системы на более низком уровне детализации и определяет корпоративные соглашения и ограничения. Как вы помните, сбор и анализ информации не прекращается на протяжении всех стадий процесса MSF.

В документе общей картины и области действия решения основное внимание следует уделять пониманию и определению проблемы. В этом документе необходимо указать:

- формулировку задач;
- общую картину решения (в виде отдельного документа);
- профили пользователей;
- область действия проекта;
- концепцию решения;
- цели проекта:
 - бизнес-цели;
 - проектные цели;

- критически важные факторы успеха;
- предварительный календарный график работ.

Создание формулировки задач

Формулировка задач (problem statement) — обычно краткое описание задач, которые заказчик планирует решить за счет реализации проекта. Этот документ создается практически исключительно на основе текущего состояния **бизнес-операций**. Чем точнее **сформулированы** задачи, тем лучше удастся оценить бизнес-требования организации.

Цель любого проекта состоит в решении какой-то проблемы, поэтому она в основном и определяет проект решения. Формулировка задач позволяет определить задачи, которые команда будет решать. Этот документ должен содержать достаточно информации о бизнес-задаче.

Вот несколько примеров формулировки задачи:

- операторы на телефоне не **в состоянии** обрабатывать большое число запросов из-за сложности и задержек в работе приложения;
- организация желает избавиться от регулярных затрат на поддержку работы старых версий оборудования и программного обеспечения;
- система должна **предоставлять** пользователям четкие указания по устранению возникших неполадок;
- необходимо увеличить число посетителей, регистрирующихся на нашем Web-сайте, за счет упрощения системы навигации.

Создание документа общей картины решения

Задача этого документа — обеспечить единое видение проекта и достичь согласия всех членов команды относительно оправданности и возможности реализации проекта. Этот документ также позволяет добиться единого мнения всей команды о будущем проекте.

Особенности документа общей картины решения

Он должен быть достаточно коротким, чтобы легко запоминался, четким и понятным, а также аргументированным, чтобы мотивировать сотрудников. Качественный документ общей картины решения отличаются следующие пять особенностей.

- **Конкретность** (Specific). Документ общей картины решения должен описывать идеализированное состояние бизнес-задачи, чтобы ожидаемый результат был понятен и аргументирован.
- **Измеримость** (Measurable) позволяет проектной команде четко определить, успешен ли проект и насколько он отвечает бизнес-целям.
- **Реалистичность** (Achievable). Проект должен быть реалистичным, то есть учитывать ограничения ресурсов, времени и опыта членов команды. Достижимые цели мотивируют команду.
- **Насущность** (Relevant). Документ должен предусматривать решение важной бизнес-задачи, в противном случае проектная команда будет решать несуществующую проблему, а сам проект лишится поддержки организации, которой он необходим.
- **Точный расчет времени** (Time-based). В документе должно быть точно указано планируемое время поставки решения.

Примечание Перечисленные особенности документа общей картины решения также называют SMART-характеристики — по первым буквам характеристик (Specific, Measurable, Achievable, Relevant, Time-based).

Примеры документов общей картины решения

Допустим, онлайнвые продажи через корпоративный Web-сайт компании значительно ниже, чем у конкурентов. При создании общей картины решения поставлена соответствующая бизнес-задача:

До конца года необходимо стать самой доходной компанией в отрасли за счет увеличения онлайнвых продаж.

Вот другой пример. В онлайнвой библиотеке доступен огромный каталог книг, журналов, статей и журналов. Руководство требует, чтобы абоненты могли пометить те книги из каталога, к которым им придется неоднократно возвращаться. Причем эта возможность должна предоставляться с любого компьютера. В этом случае документ общей картины решения выглядит так:

В течение текущего бюджетного года необходимо предоставить всем нашим абонентам возможность выделять закладками избранные страницы нашего Web-сайта, причем эта функция будет доступна с любого компьютера или устройства, на котором работает конечный пользователь.

Обратите внимание, что оба документа обладают SMART-характеристиками: они конкретны, измеримы, реалистичны, насущны, а временные рамки четко определены.

Создание пользовательских профилей

Бизнес-решение требуется разным клиентам. До начала разработки проекта важно понять, кому предназначается продукт, поэтому команде следует четко определить виды, или профили, будущих пользователей. Эти профили определяют потребности той или иной группы пользователей, что позволяет команде оценить ожидания, риски, цели и ограничения проекта. При создании пользовательских профилей следуйте нескольким рекомендациям.

- **Цели** конечного пользователя — это список особенностей поведения, которыми должен, по мнению пользователя, обладать продукт.
- **Ограничения.** Важно понять факторы, которые влияют на способность пользователей работать с решением, например необходимость особого оборудования и программного обеспечения. В некоторых проектах обязательным условием является наличие у пользователя вполне определенной ОС. Нет смысла проектировать решение для одной отдельно взятой операционной системы, если предполагается, что корпоративные пользователи применяют различные ОС.
- **Особенности поддержки.** В пользовательских профилях иногда указывают проблемы, с которыми пользователи сталкивались при работе с аналогичными решениями. Эта информация поможет спланировать функции поддержки, которые понадобятся при работе с новым решением.
- **Иностраные пользователи.** Выясните, будет ли решение поддерживать другие языки и необходима ли локализация.

- **Географическое распределение.** Опишите местоположение пользователей, в том числе географическое и физическое, а также число пользователей в каждом месте, пропускную способность и загрузку линий связи между отдельными сайтами.
- **Обмен информацией.** Опишите информационные потоки между пользователями, в том числе тип связи, важность и объем данных, которыми обмениваются различные группы пользователей.
- **Пользовательские функции.** Опишите задачи, выполняемые пользователями, например «создание профиля клиента» или «редактирование заказов и параметров клиентов». Эта информация пригодится при разработке вариантов использования системы.
- **Внутрикорпоративные связи.** В некоторых организациях существует жесткое иерархическое деление, четко определяющее кто, как, почему и когда имеет право обращаться в другие подразделения. Будущее решение должно удовлетворять эти ограничения. Задокументируйте состав и границы организационной иерархии.
- **Порядок принятия решений.** Опишите политики принятия решений, которые непосредственно влияют на эффективную работу создаваемого продукта.
- **Другие факторы.** Следует задокументировать доступность и использование ресурсов в каждом месте и указать все дополнительные факторы, например несовместимые протоколы, сетевые ОС и приложения, которые могут помешать реализации географически распределенного решения.

Определение области действия

Одно из критически важных условий успеха проекта — четкость определения *области действия проекта* (project scope), то есть того, что входит в рамки проекта. Этот параметр определяют на основе *общей картины решения* и ограничений, обусловленных конечностью проектных ресурсов, времени и других факторов. Область действия также зависит от функций, которые заказчик считает обязательными и которые команда должна реализовать в первой версии решения. При определении *границ* проекта команда вправе перенести в *будущие* версии функциональные возможности, *которые* напрямую не связаны с базовыми функциями решения. Функциональность, не входящая в *область* действия, документируется в следующей версии или *следующем* проекте.

Приступая к определению области действия проекта, вы продолжаете развивать *высокоуровневые* варианты использования системы, созданные в процессе анализа *бизнес-процессов* и описания *высокоуровневых бизнес-требований*. Для определения области действия проекта необходимо определить части ВИС, которые непосредственно влияют на бизнес-процессы и удовлетворяют бизнес-требованиям. Иногда требуется определить важность бизнес-задач и выбрать те, что будут реализовываться в будущих версиях продукта. Обычно соответствующие части ВИС выделяют и создают новую *ВИС-диаграмму*, которую впоследствии используют при разработке в проекте.

На рис. 3-3 показана *ВИС-диаграмма*, представляющая часть бизнес-операций. Область действия проекта выделена прямоугольником.

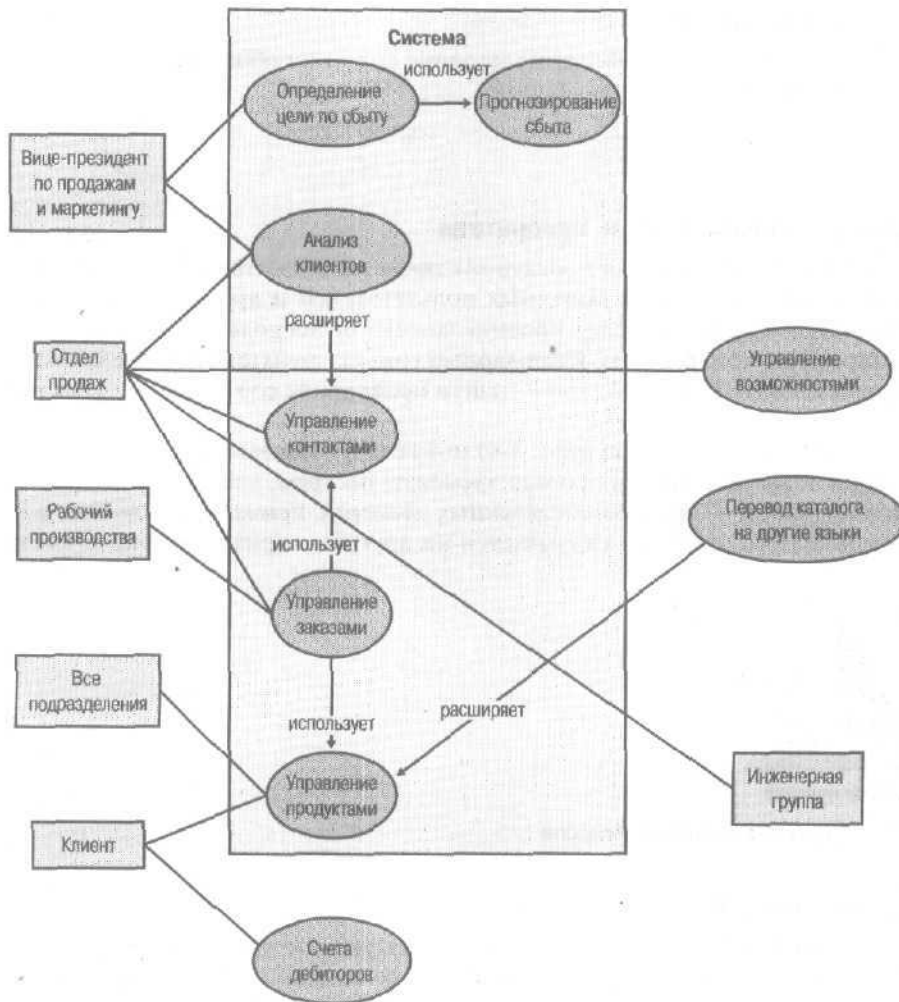


Рис. 3-3. ВИС-диаграмма с выделенной областью действия проекта

На рис. 3-3 показаны только те ВИС, которые попадают в область действия проекта. Чтобы понять, что не входит в эти рамки, обратитесь к документу *AWC Use Cases - Chapter 3.vsd* из папки `\Solution Documents\Chapter03` на прилагаемом к книге компакт-диске.

Уточнение требований

Это очень важная операция. Как вы помните, анализ и уточнение требований продолжается на этапах создания общей картины приложения и планирования. На первом этапе анализируют собранную информацию и определяют конкретный план, который в дальнейшем трансформируют в документ требований, где описаны все необходимые и желательные возможности для компании. Вы также вправе опрашивать клиентов, или собирать информацию из других источников. Вот пример высокоуровневого требования:

Система должна обеспечивать возможность локализации, что необходимо для поддержки

языка конечных пользователей.

После уточнения этого требования на этапе создания общей картины решения оно приобретает следующую форму:

В системе необходимо предусмотреть механизмы поддержки других языков, локализации и доступности.

Треугольник компромиссов и приоритеты

Определение области действия проекта заключается в достижении баланса потребностей самых различных конечных пользователей и других приоритетов, определенных заказчиком. Успех проекта зависит от нескольких параметров, в числе которых затраты, ресурсы, календарный график, функциональные возможности и надежность. Ключ к успеху — найти правильное соотношение этих параметров.

Треугольник компромиссов (рис. 3-4) показывает три наиболее важных для определения области действия проекта элемента; ресурсы, календарный график и функциональность — три взаимосвязанных элемента, причем ограничение или расширение одного элемента сказывается на других — приходится принимать компромиссы.



Рис. 3-4. Треугольник компромиссов

Роль выпуска версий

При определении области действия проекта выясняют требования, которым должны удовлетворять версии решения. На этапе создания общей картины определяют, какие варианты использования системы и требования следует реализовать в продукте.

Описанные функции соответствуют требованиям. Решение о том, включать ту или иную функцию, зависит от бизнес-задачи и от того, на скольких пользователей влияет функция. В первую очередь следует реализовать те из них, которые затрагивают наибольшее число пользователей.

После определения состава и важности бизнес-задач рассматривают варианты использования системы, относящиеся к самой важной задаче. В каждой последующей версии основное внимание уделяют подмножеству функций, которые решают самые важные задачи и оказывают влияние на самое большое число пользователей. Уберите функции, требуемые небольшим группам пользователей. Так определяется область действия каждой версии решения и всего проекта.

Примечание Как уже говорилось, документ требований преобразуется в список функций ближе к концу логического дизайна, о котором подробно рассказывается в главе 5.

Роль предположений и ограничений

Область действия проекта также зависит от предположений и ограничений, определяемых компанией и клиентами. Обычно предположения относят к ограничениям проекта. Вот примеры предположений.

При разработке будет использоваться Microsoft .NET Framework.

Разработкой будут заниматься две команды программистов.

Предполагается поддерживать обе системы - OLAP и OLTP.

Ограничения задают «прокрустово ложе» бизнес-решения. Это параметры бизнес-среды, которые не должны или не будут изменяться. Часто эти ограничения становятся целями проекта. Не определив должным образом ограничения, вы рискуете создать продукт, который не удастся развернуть в компании.

Вот перечень возможных ограничений, которые необходимо задокументировать:

- бюджетные ограничения;
- характеристики существующих систем поддержки;
- системная архитектура сети;
- требования по безопасности;
- операционные системы;
- запланированные обновления технологий;
- ограничения пропускной способности сети;
- соглашения и структуры обслуживания и поддержки;
- уровень знаний сотрудников отделов разработки и поддержки;
- пользовательские ограничения.

Перечислите ограничения, которые влияют на удовлетворение бизнес-требований и возможные решения. Проектная команда использует эти ограничения для создания продукта, который максимально отвечает требованиям и соответствует параметрам, определенным ограничениями.

Роль оценок

В зависимости от предположений и ограничений проект оценивают, т. е. определяют время и ресурсы, необходимые для построения решения, и стоимость ресурсов.

Кроме того, в разделе оценок указывают все оговорки и подробности договорных соглашений. Это помогает уточнить обязанности проектной команды.

Преимущества определения области действия

В этом разделе перечислены некоторые преимущества, появляющиеся при определении области действия проекта:

- команда может сосредоточиться на определении того, что необходимо сделать;
- крупные и нечетко определенные задачи удастся разбить на меньшие и более понятные подзадачи;
- удастся определить функции, реализуемые в каждой версии;
- удастся вычленить наборы функций будущего продукта, которые помогают распределить работу среди субподрядчиков или в команде;
- удастся определить, что команда должна делать и что не входит в ее обязанности.

Уточнение области действия

На этапе создания общей картины решения становятся понятными общие очертания проекта. Однако детальное планирование позволяет команде глубже разобраться в особенностях проекта. На этой стадии этапа создания общей картины решения вероятнее всего ясны лишь базовые параметры проекта. Определение и достижение баланса параметров проекта выполняется итеративно. В процессе анализа, создания прототипов и планирования команде иногда приходится пересматривать область действия, чтобы:

- реализовать уточненные требования пользователей;
- предусмотреть изменение в бизнес-требованиях;
- скорректировать решение в соответствии с техническими трудностями или рисками;
- сбалансировать такие параметры проекта, как ресурсы, время и функциональность.

Создание концепции решения

Концепция решения описывает подход команды к решению задач проекта и служит основанием для перехода на этап планирования. После определения бизнес-задачи и создания общей картины и области действия решения команда создает концепцию решения, где в общих словах описано, как команда планирует решать поставленную задачу.

На рис. 3-5 показана концепция решения для Adventure Works Cycles.



Рис. 3-5. Концепция решения

Создание предварительного плана

Концепция решения может описывать компанию. Поскольку она ориентирована только на концепции, а не на подробности реализации решения, концепция решения содержит не слишком много технических сведений. Ее должен одобрить спонсор (это обычно один из топ-менеджеров); далее на основании документа общей картины и области действия решения он будет добиваться выделения финансирования.

Концепция решения включает концептуальную модель программной и аппаратной архитектуры системы. Концепция решения — это предлагаемый метод решения задач, вошедших в область действия. Команда должна оценить различные варианты и выбрать максимально подходящий для конкретной ситуации.

Это позволяет команде сузить перечень вариантов концепции решения до нескольких альтернатив.

Например, в случае Web-сайта электронной коммерции возможны такие варианты: разработка сайта «с нуля», привлечение для выполнения этой работы внешней компании или покупка готового решения. Аналогичный выбор появляется при решении проблемы хостинга Web-сайта: размещать его на локальных серверах или на площадке Интернет-провайдера.

Элементы концепции решения

После оценки вариантов команда выбирает концепцию решения, которая лучше всего отвечает ее потребностям, ресурсам и запасу времени. Концепция решения содержит следующие элементы:

- факторы успеха, определяющие успех проекта, и критерии приемки — контрольный список требований, которые необходимо удовлетворить до передачи решения для промышленной эксплуатации;
- предварительно определенный подход к разработке и поставке продукта, который включает типовые сценарии работы сайта и методы реализации решения, число пользователей нового продукта и полный список предполагаемых результатов проекта;
- предварительное описание функциональных возможностей решения, которые реализуют основную бизнес-задачу.

Примечание Создание пользовательских профилей и концепции решения можно выполнять параллельно, а не последовательно.

Определение целей проекта

Для успеха проекта необходимо правильно определить его цели, которые бывают двух типов:

- бизнес-цели;
- проектные цели.

Бизнес-цели

Они определяют задачи, которые клиент планирует решить, внедрив новый продукт. Бизнес-цели становятся основой для определения критериев успешности решения. Определение бизнес-целей необходимо для четкой постановки задач проекта и гарантирует, что новый продукт удовлетворит бизнес-требования. Команда должна выбрать наиболее подходящий метод определения и согласования целей.

На каждом этапе проекта команда следит за балансом ресурсов, времени и функций решения. Необходимо распределить бизнес-цели по их важности, что позволит команде четко понимать, что, по мнению заказчика, действительно важно и какими целями можно при крайней необходимости пожертвовать.

Далее перечислены типичные бизнес-цели проекта решения для электронной коммерции:

- расширение географии сбыта компании за счет использования ресурсов, отличных от обычных магазинов;
- расширение рынка сбыта компании за счет «подключения» более молодых потребителей, обладающих достаточно высоким уровнем доходов и посещающих онлайн-магазины чаще, чем уже сложившаяся аудитория;

- сокращение времени на сбыт за счет развертывания более эффективных онлайн-сайтов;
- объединение всех поставщиков из разных стран в единую цепочку поставок и сокращение среднего времени размещения заказа и цикла поставки.

Проектные цели

Проектные цели во многом похожи на бизнес-цели. Различие в том, что первые больше сфокусированы на атрибутах решения и меньше — на ценности будущего решения для поддержки бизнеса. Проектные цели определяют не только то, что команда планирует выполнить, но и то, что она *не будет делать*. Как и с бизнес-целями, необходимо определить важность проектных целей — это поможет команде лучше ориентироваться в приоритетах заказчика и при необходимости проигнорировать наименее важные задачи.

Проектные цели того же продукта для электронной коммерции выглядят так:

- повысить удовлетворенность пользователей за счет сокращения среднего времени загрузки страницы до 5 секунд или меньше;
- свести к минимуму зависимость от наличия подключения к серверу;
- сократить время и усилия, затрачиваемые пользователем на онлайн-регистрацию.

Для проекта расположенной на сервере онлайн-библиотеки с поддержкой пользовательских закладок проектные цели таковы:

- локализация на многие языки самого сервиса и всех вспомогательных приложений;
- обеспечение доступности сервиса на уровне 99,99%;
- гарантия сохранности данных сервиса;
- предоставление доступа к сервису только уполномоченным пользователям.

При проектировании интерфейса мобильного приложения возможны такие проектные цели:

- простота и легкость поиска информации;
- настройка приложения в соответствии с особенностями конкретного мобильного устройства.

Проверка документа общей картины и области действия решения

После создания предварительной версии документа общей картины и области действия решения команда анализирует и обновляет документ. Стадия создания общей картины решения завершается контрольной точкой «Утверждение документа общей картины и области действия проекта». На этой стадии проектная команда и заказчик достигают единого мнения относительно общего направления проекта, а также области действия решения. Эта базовая версия документа является результатом данного этапа проекта и используется на последующих этапах проекта.

Документ общей картины и области действия решения формально одобряется на собрании, посвященном определению общего видения и границ проекта. Члены команды вместе с выбранными пользователями проверяют созданные ВИС, СИС и пользовательские профили. (Подобная проверка выполняется на всех этапах проекта.) Привлечение заказчика к утверждению завершённой работы гарантирует, что клиент понимает необходимость компромиссов при опреде-

лении области действия. Заказчик также получает представление о самом проекте и методах, применяемых командой для его реализации, — заказчик практически участвует в проекте, что значительно облегчает согласование и последующие обсуждения *функциональности* продукта.

Собрание, посвященное определению общего видения и границ проекта, позволяет команде и заказчику согласовать видение того, как предлагаемое решение должно решать бизнес-задачу и как оно соответствует текущему состоянию бизнес-процессов в рамках данной области действия.

Кроме того, заказчик принимает активное участие в проекте и *удовлетворен* тем, что команда прислушивается к его мнению. На этом собрании в проектной команде принимают окончательное решение, стоит ли продолжать проект. Цена имеющихся ресурсов и потенциальная прибыль могут не соответствовать общим затратам на разработку. Эта стадия позволяет команде пересмотреть проектное решение или ресурсы и ограничения.

Одобрив документ *общей* картины и области действия решения, члены проектной команды, заказчик и другие ключевые участники проекта *согласуют* следующее:

- определенные в общем виде потребности бизнеса, которые будет удовлетворять будущее решение;
- общую картину решения;
- цели проекта решения;
- риски, связанные с исполнением проекта;
- начальную концепцию бизнес-решения, как ее понимают менеджеры проекта;
- состав проектной команды;
- механизмы управления проектом.

Готовый документ *общей* картины и области действия решения для Adventure Works Cycles вы найдете в файле *AWC - Vision Scope.doc* в папке *\SolutionDocuments\Chapter03* на прилагаемом к книге компакт-диске. Проектная команда определила, что область действия версии 1 проекта охватывает *анализ* и ввод клиентских *заказов*, управление контактами и процессы заказа через Web. Таким образом, отслеживание товаров, документацию о взаимоотношениях сотрудников и данные о поставщиках предполагается реализовать в следующих версиях или *будущих* проектах.

При появлении каких-либо ранее неизвестных проблем, которые требуют изменения области действия или результатов проекта, команда должна организовать обсуждения и встречи для пересмотра *общей* картины и границ проекта.

Занятие 3. Создание документа структуры проекта

Документ структуры проекта — основной результат этапа создания общей картины решения. В этом занятии описывается цель создания такого документа и его главные разделы.

Изучив материал этого занятия, вы сможете:

- ✓ рассказать, для чего предназначен документ структуры проекта;
- ✓ описать компоненты документа структуры проекта.

Продолжительность занятия — около 5 минут.

Документ структуры проекта

Этот документ определяет подход команды к организации и управлению проектом. В нем описывается административная структура команды, стандарты и процессы, а также ресурсы и ограничения проекта. Это основной документ, где указан порядок взаимодействия членов проектной команды.

Он может выглядеть, как **формальный** документ с описанием порядка деятельности каждой из MSF-ролей команды в проекте. Кроме того, в нем документируется порядок управления изменениями и конфигурацией в проекте. Уровень детализации документа структуры зависит от проекта. Если на более поздних стадиях проекта ожидаются значительные изменения, документ должен подробно описывать, как команде следует реагировать на них.

Далее показан раздел управления изменениями из документа структуры проекта Scout, в котором участвуют две компании — Adventure Works Cycles и Contoso, Ltd.

Управление изменениями

Самое важное в проекте Scout - поставка решения с базовым набором функций до даты завершения проекта. (Конкретная дата будет определена после завершения работы над генеральным планом проекта. Примерные даты завершения: 1 сентября 2003 г. - для Web-сайта и 15 ноября 2003 г. - для проекта автоматизации продаж.) Поэтому процедуры управления изменениями предполагается реализовать указанным ниже образом.

Процесс управления изменениями и ответственные за документацию

Менеджер программы несет ответственность за процесс управления изменениями и соответствующие документы. Менеджер проекта компании Adventure Works Cycles отвечает за принятие решений в процессе управления изменениями, требуемыми заказчиком.

Распределение функций по версиям

В процессе планирования, сбора требований и разработки спецификации функциональности каждой функции необходимо присваивать параметр важности и версии, в которой ее предполагается реализовать, например **Critical V3.0** (то есть важность: критическая, версия: 3.0).

Совет по изменениям

В совет по изменениям проекта Scout входят сотрудники Contoso, Ltd. (команда разработчиков и менеджер программ) и Adventure Works Cycles (менеджер проекта и разработчики), причем все члены совета наделены правами вносить запросы на изменение функциональности. Другие члены команды могут обращаться со своими предложениями к любому члену совета, который в свою очередь выносит предложение на обсуждение на следующей встрече совета. Если член команды считает функцию критически важной, он вправе определять ее как таковую и добиваться оценки раньше, до запланированной встречи совета.

Оценка функций

Команда оценивает определение набора функций, его влияние на проект и решение, а также состав функций в версии 1.0 (или любой другой версии) и соответствующие риски.

Компромиссы при определении функциональности

Функции сравниваются друг с другом, и выявляются все возможные компромиссы.

Компромиссы при определении ресурсов

Изучается треугольник компромиссов и добавляется или удаляется соответствующий элемент (ресурс или функция). (Календарный график фиксирован.)

Распределение функций по версиям

После определения важности и распределения функций по версиям, а также добавления или удаления ресурсов или других функций назначается место каждой функции в проекте. Бюджет и договор корректируются. (Если Adventure Works Cycles посчитает, что график необходимо изменить, чтобы реализовать отдельную функцию, компания вправе запросить изменение календарного графика и внести соответствующие коррективы в договор, проект, бюджет и календарный график.)

Прекращение изменений

Обе компании соглашаются с тем, что контрольная точка определения области действия считается достигнутой при 70-процентной готовности, а любые новые функции могут вноситься только в версии 2.0 или 3.0.

Роли и их ответственность в команде

Менеджер программы отвечает за организацию создания документа структуры проекта. Основной вклад в создание документа вносят члены команды.

Компоненты проектной структуры

Есть три основных компонента документа структуры проекта:

- команда и структура;
- проектные оценки;
- графики проекта (предварительные версии).

Документ структуры проекта — это инструмент документирования решений, принятых в отношении выполнения и управления проектом, в том числе в нем указаны:

- роли и обязанности заказчика и проектной команды;
- решения о связях;

- логистические решения;
- решения по управлению изменениями;
- решения относительно оценки хода проекта.

Роли и обязанности заказчика и проектной команды

В разделе ролей и обязанностей перечисляются имена занятых в проекте специалистов и их контактная информация — номера телефонов и адреса электронной почты. Кроме того, здесь описываются обязанности различных ролей на следующих стадиях проекта.

Решения, принимаемые на этапе планирования

В процессе этапа планирования команда должна ответить на ряд вопросов.

- Как будут разрабатываться проектные планы?
- Как при создании проектного плана команда будет применять знания и опыт, полученные в результате анализа других проектов?
- Требуются ли регулярные совещания при разработке проекта?
- Как часто заказчику необходимо оценивать продукт?
- Как распределить функции по различным версиям решения?
- Кто определяет риски, связанные с проектом, и план обеспечения непрерывности бизнеса?
- Какие инструментальные средства следует применять при разработке и контроле проектных планов?
- Будет ли команда собираться на совещания, посвященные анализу продукта, в течение этапа планирования? Кого следует пригласить на такие совещания?

Решения, принимаемые на этапе разработки

В течение этапа разработки команда ответит на следующие вопросы.

- С какими командами, организациями и сторонними поставщиками следует взаимодействовать команде на протяжении проекта? Кто отвечает за создание и управление соглашениями со сторонними поставщиками?
- Каковы роль и ответственность лидеров команды при разработке?
- Если различные компоненты решения создают различные команды, то каковы роль и ответственность лидеров подкоманд проекта?
- Необходимы ли каждой подкоманде группы создания справочной системы, обеспечения удобства пользования и тестирования?
- Какова роль менеджеров в проекте?
- Какова роль субподрядчиков, если таковые требуются, в проекте? Кто отвечает за подбор субподрядчиков и контроль их работы?

Решения по связям

В этом разделе определяется порядок обмена информацией, который будет соблюдать команда на протяжении проекта. Он описывает связи как в самой команде, так и между командой и заказчиками. Этот раздел отвечает на ряд вопросов.

- Кого необходимо информировать о решениях, принимаемых при планировании?
- Как заказчиков, участников проекта и команду следует информировать о таких решениях?

- Какие собрания предполагается провести, где и кто должен присутствовать на них?
- Кто отвечает за формирование повестки собраний?
- Кто должен вести встречи?
- Кто будет отвечать за ведение и распределение протоколов совещаний?
- Кого и как следует информировать о ходе проекта?

Решения, касающиеся проектных записей

Как правило, при разработке каждого проекта ведут записи о типах контрактов, соблюдении графиков и планов. В документе структуры проекта описывается, какая информация будет содержаться в проектных файлах. Вот некоторые из решений, которые необходимо принять в отношении проектных записей.

- Какая информация должна регистрироваться в проектных записях, например проектные задания, графики, планы, нерешенные проблемы, соглашения и т.п.?
- Кто будет вести проектные записи?
- Кому следует предоставить доступ к проектным записям?
- Кто будет отвечать за хранение проектных файлов после завершения проекта и как долго их следует хранить?

Решения, касающиеся анализа решения после его внедрения

Такой анализ — важная часть проекта. Он помогает оценить успешность проекта и определить, какая поддержка необходима заказчику. На этой стадии также следует принять несколько решений.

- Когда после внедрения провести анализ и кто будет в нем участвовать?
- Какие части решения предполагается анализировать?
- Какую информацию и кто будет собирать на протяжении проекта для выполнения анализа после внедрения?

Логистические решения

Этот раздел документа структуры проекта перечисляет решения, касающиеся разработки решения, и отвечает на следующие вопросы.

- Какие методики разработки будут использоваться в проекте? В том числе предлагаются:
 - методы разработки, такие, как проверки спецификации продукта и контрольный список достижения бездефектной стадии;
 - методы тестирования;
 - методы документирования;
 - методы маркетинга.
- Кто определяет состав спецификаций продукта? Кто и как будет использовать спецификации продукта в процессе реализации проекта?
- Какие инструментальные средства потребуются для определения набора функций решения?
- Каковы критерии готовности спецификаций продукта?
- Кто отвечает за обновление спецификаций продукта?
- Какие части спецификации необходимо предоставить каждому стороннему поставщику, участвующему в проекте? Кто будет создавать эти спецификации?
- Как методы достижения контрольной точки отсутствия дефектов следует определить и реализовать в проекте?

- Кто оценит необходимый для проекта объем рабочей силы и времени? Что следует положить в основу этой оценки (личный опыт, анализ после внедрения и др.)?
- Кого следует обучать технологиям и навыкам, требуемым для работы над продуктом (например, менеджменту проектов)? Кто нуждается в обучении работе с решением на этапе планирования?

Решения по управлению изменениями

В этом разделе документа структуры проекта определяется порядок реагирования на любые изменения в проекте. Вот некоторые вопросы, на которые следует ответить при создании этого раздела.

- Как определяется изменение в проекте (например, редактирование графика или увеличение стоимости проекта)?
- Какова дата выпуска? Кто и как ее определяет? Что при этом учитывается?
- Кто устанавливает порядок управления изменениями?
- Как планируется определять и отслеживать изменения? Кто будет отслеживать эту информацию?
- Как оценивать влияние изменения? Каков порог числа отклонений, после которого необходима серьезная перепланировка?

Решения относительно оценки хода проекта

В этом разделе определяется порядок отслеживания и оценки хода проекта.

- Как оценивать прогресс проекта? Какая информация потребуется для оценки выполнения проекта?
- Как опрашивать членов команды, чтобы собрать информацию о решении той или иной задачи? Как часто планируется собирать такую информацию?
- Как часто обновлять календарные графики команды? Генеральный график проекта?
- Кто определяет и оценивает влияние каждого изменения?
- Кто будет заниматься адаптивными операциями, в том числе рекомендовать и одобрять их? Как оценивать эффективность этих операций?
- Как регистрировать, отслеживать и контролировать устранение нерешенных проблем?
- Каковы критерии определения исключения для целей отчетности? Кто займется оценкой и устранением исключений?

Занятие 4. Анализ рисков

Риск — это возможность потерь: от снижения качества решения до роста затрат, нарушения графика или провала проекта. MSF рекомендует оценивать риск постоянно, на протяжении всего проекта. На этом занятии рассказывается об анализе рисков и создании документа рисков проекта.

Изучив материал этого занятия, вы сможете:

- ✓ описать процесс управления риском в MSF;
- ✓ описать содержание документа рисков продукта;
- ✓ определить проектные риски.

Продолжительность занятия — около 10 минут.

Процесс управления риском

Риск — результат неопределенности результатов проектных решений. Важная сторона разработки успешного решения заключается в управлении и устранении рисков проекта. Большинство подразумевает под риском возможность потерь денег или нарушения контроля, функций, качества или своевременности завершения проекта. Однако к проектным рискам также относится недополучение максимальной выгоды или упущенная выгода в результате неопределенности при принятии решения.

В MSF процесс управления риском (рис. 3-6) подразумевает активную позицию команды на протяжении всего проекта. Команда постоянно оценивает, что может пойти «не так» и как предотвратить или свести к минимуму потери. MSF предполагает создавать формальный документ, где оцениваются риски и определяется их важность.

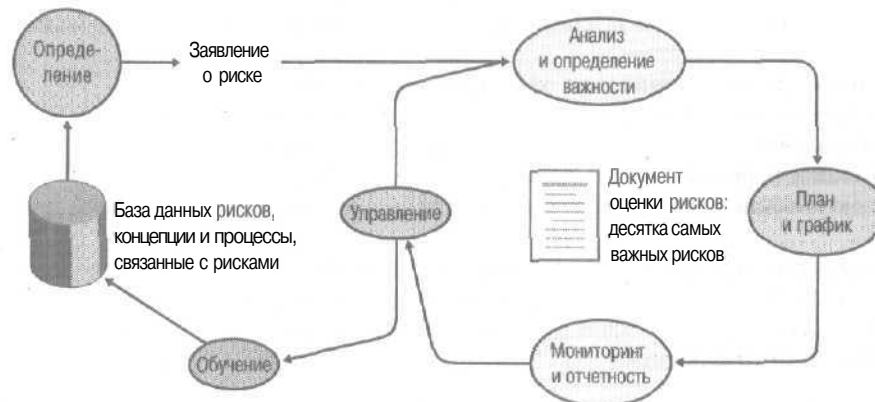


Рис. 3-6. Процесс управления рисками в MSF

В MSF определены шесть стадий управления рисками, на которых команда планирует и претворяет в жизнь стратегии управления рисками и документирует полученные знания для применения в будущих проектах.

1. Определение рисков позволяет выявлять их, благодаря чему команда получает информацию о возможных проблемах.

2. **Анализ и определение важности рисков** — преобразование оценок и информации о конкретных проектных рисках, обнаруженных на предыдущей стадии, в форму, пригодную для принятия решения об их важности, а также для принятия ответных мер.
3. **Планирование рисков** — использование результатов предыдущей стадии для формулировки стратегий, планов и мероприятий по управлению или устранению рисков. Обязательно предусмотрите в плане проекта время для планирования рисков.
4. **Отчетность и мониторинг рисков** — наблюдение за конкретными рисками и документирование планов мероприятий по управлению ими или устранению. Отчеты предоставляются команде, клиентам и ключевым участникам проекта.
5. **Управление рисками** — процесс выполнения планов мероприятий по управлению и устранению рисков и предоставление отчетов о состоянии дел в этой сфере команде и заказчику.
6. **Извлечение уроков** — формализация приобретенного опыта и соответствующих проектных документов и инструментальных средств, а также сохранение приобретенных знаний в форме, доступной для их повторного использования в компании.

Примечание Стадии управления риском — **логические**, не обязательно соблюдать последовательность их выполнения. В процессе выявления связанных с проектом рисков многие команды проводят несколько итераций определения, анализа и планирования, но не каждую итерацию завершают стадией извлечения уроков.

Содержание документа оценки рисков

На этапе создания общей картины решения команда создает документ оценки риска, в котором описываются и **документируются** все известные риски, а также оценивается их вероятности и влияние. Такой документ содержит следующие разделы:

- **определение рисков** — описание отдельных видов риска;
- **вероятность риска** — **вероятность** реализации рискового события;
- **серьезность риска** — разрушительность возможных последствий;
- **подверженность риску** — уровень подверженности риску;
- **профилактические мероприятия** — усилия по предотвращению или снижению риска;
- **мероприятия по обеспечению непрерывности бизнеса** — порядок и последовательность операций, выполняемых в случае наступления рискового события;
- **ответственный за риск** — член команды, который несет ответственность за регулярный контроль риска.

Создание документа оценки риска

Документ оценки риска создает менеджер программ. Перед документированием концепции решения рекомендуется сравнивать риски, связанные с различными концепциями решения.

Допустим, вы создаете Web-сайт электронной коммерции. Один из вариантов — создание сайта «с нуля». Однако для этого требуется организовать и обеспечить управление значительными ресурсами, которые к тому же придется от-

влечь от других дел. Также для выполнения проекта потребуется значительное время. Другой вариант — ускорение разработки за счет использования Microsoft Solution Accelerator for the Internet Storefront. Практически готовый исходный код, MSF-принципы планирования проектов и комплект ресурсов со справочной информацией и инструментальными средствами поможет команде решить поставленные бизнес-задачи. Кроме того, удастся снизить влияние рисков, связанных с созданием сайта «с нуля».

Количественная оценка рисков

Создавая документ оценки риска, определите в числовом выражении вероятность и влияние риска. Первая определяет вероятность, а второе — серьезность возможных потерь. Затем вычислите подверженность риску, перемножив эти два числа. Таким образом вам удастся сравнивать риски и определять их относительную серьезность и важность.

Внимание! Не обязательно использовать одинаковую шкалу для оценки вероятности и влияния. Однако необходимо применять одинаковый подход при оценке вероятности и влияния каждого риска. Если удастся точно оценить финансовые потери, влияние риска можно выражать в финансовых терминах. Однако если вы определите влияние одних рисков посредством безразмерного числа, а других — в денежном выражении, их не удастся сравнить.

Выявление десятки самых важных рисков

После создания документа оценки риска и определения важности отдельных его видов, создают список самых серьезных рисков — именно на них команда сосредоточит свои усилия в первую очередь. Такой список называется «десяткой», но он не обязательно состоит именно из десяти рисков. Менеджер программы должен достаточно часто просматривать список и обновлять его в соответствии с важностью рисков.

Некоторые риски в проекте электронной коммерции уникальны, поэтому команда не всегда знает о них. К ним относятся:

- недостаток опыта разработки, развертывания и поддержки Web-сайтов
- возможный ущерб в связи с использованием логотипа или товарной марки компании на сайте, который работает некорректно или на котором наблюдаются проблемы с доступностью;
- риски, связанные с безопасностью, в том числе вероятность вторжения в систему злонамеренных пользователей и воровство конфиденциальных данных клиентов, например номеров кредитных карточек

Практикум. Разработка документа общей картины и области действия решения



При выполнении упражнений используйте знания о создании общей картины приложения, полученные на занятиях. Все упражнения этого практикума основаны на описанных далее исходных условиях.

Исходные условия

Adventure Works Cycles — крупная международная компания, производящая и поставляющая велосипеды из металла и композитных материалов на рынки Северной Америки, Европы и Азии. Помимо расположенных в г. Ботшелл, штат Вашингтон, штаб-квартиры и основного завода, где работают 500 человек, компания имеет несколько региональных торговых подразделений, расположенных во всех странах мира, где Adventure Works Cycles продает свою продукцию.

Торговые представители региональных отделений отвечают за сбыт на территории **соответствующих** регионов. Штат каждого регионального отделения состоит из нескольких торговых представителей и менеджера по продажам. В своей повседневной работе торговые представители используют ноутбуки и КПК под управлением Microsoft Windows CE.

Обычный рабочий день торгового представителя начинается с подключения к региональному отделению и загрузки свежих данных, в том числе информации о наличии товаров на складе, сведений о товарах и маркетинговой информации. При посещении клиентов торговый представитель вносит сведения о заказах в свой ноутбук или КПК. В конце каждого рабочего дня он назначает встречи на следующий день или неделю, просматривает встречи, запланированные другими представителями, чтобы определить возможность совместных действий, и обновляет список контактов. Он подключается по модему к **региональному отделению**, отправляет новую информацию и принимает свежие данные из главного или **регионального** отделения.

Данные о клиентах хранятся в системах, но у региональных отделов нет единого и простого способа выборки и запроса нужной информации из них. Например, система не позволяет определить наилучших клиентов в общем зачете или лучших покупателей отдельных моделей. Компании требуется разумный механизм для определения лучших клиентов, а также параметров, по которым такие клиенты определяются.

В последние месяцы отделения продаж приступили к продвижению товаров компании на европейские и азиатские рынки. Пока вся информация хранится в системах только на английском языке. Торговый представитель из ганноверского офиса, говорит, что перед каждым посещением потенциального клиента ему приходится переводить все данные о продукте на немецкий. Он хорошо владеет языком, но есть технические термины, в правильности перевода которых он не уверен. Компания должна поддерживать информацию на многих языках, чтобы избавить торговых представителей от перевода спецификации.

Отдел продаж должен ежедневно получать самую свежую информацию о ценах. В настоящее время торговый представитель подключается к корпоративной сети и загружает данные о ценах каждое утро. Однако в процессе загрузки не выделяются обновленные цены у данного торгового представителя, поэтому ему каждое утро приходится выводить весь список. Как правило, выясняется, что

цены в интересующих торгового представителя секторах не изменились, и 20 минут на загрузку тратятся впустую.

Такое затруднение при загрузке вызвало неприятные последствия. Большинство торговых представителей отказалось от ежедневного обновления данных, а значит, они не в курсе действительно важных изменений. Зачастую им приходится переделывать заказ, повторно вычислять сумму и уведомлять клиентов о переоценке. Естественно, клиенты этим недовольны. В настоящее время отделением продаж предоставлено право в подобных ситуациях игнорировать цены и продавать товары по ценам, ранее оговоренным с клиентами. Но подобной практикой недовольна штаб-квартира, потому что из-за этого компания недавно потеряла значительные деньги.

Сейчас данными о потенциальных клиентах управляют сами торговые представители, причем эта информация не попадает в систему. Каждый торговый представитель создает документ Microsoft Word, куда заносит контактную информацию: имя, адрес, телефон, товары, интересующие клиента, дату первой встречи, дату последнего контакта и т.д. Кроме того, торговый представитель отслеживает «правила», которые помогают превращать возможность в реальную продажу. Например, «Клиент С купит товар Р, если (а) цена товара Р снизится на 5% или (б) если товар Р предложить с компонентами Х, Y и Z в качестве дополнительных возможностей».

Предполагается, что сотрудники региональных отделов продаж должны применять систему управления клиентами для планирования, реализации и отслеживания маркетинговой и сбытовой деятельности. Однако эта система сильно устарела, так как создана на языке четвертого поколения, и получить из нее какую-либо информацию крайне затруднительно. Кроме того, чтобы пользоваться этой системой, торговые представители вынуждены устанавливать на своих компьютерах тяжеловесное приложение и подключаться к корпоративной сети.

Сотрудники отделов сбыта определили, какие именно новшества позволят им лучше выполнять их работу.

- **Сегментация и профилирование клиентов.** Команда отдела продаж должна на основании «сырых» данных получать полезную информацию, позволяющую ответить на следующие вопросы.
 - Каковы ранние признаки возможных проблем?
 - Кто лучшие клиенты в каждой из линеек товаров? С кем следует наладить долгосрочные отношения?
 - Каковы проблемы заказчиков, распределенные по категориям: географическое местоположение, уровень доходов и т.п.?
 - Сколько и какие товары приобретают клиенты?
- **Процесс продаж.** Для координации сотрудничества отделений в различных странах отдел по продажам нуждается в поддержке международных операций, в том числе требуется поддержка формата Unicode, многих языков и различных форматов валют.
- **Внутренняя связь.** Каждый торговый представитель должен получить только предназначенную ему информацию о клиенте и продажах. Кроме того, каждый менеджер по продажам должен получать нужную только ему информацию о клиентах и назначенных встречах наряду с детальной информацией о каждом торговом представителе своего отделения.

- **Управление потенциальными клиентами.** Торговым представителям необходим механизм сохранения и просмотра данных о **потенциальных клиентах** и возможность при заключении сделки переносить часть или всю необходимую информацию в заказ, чтобы не вводить ее повторно.
- **Система поддержки принятия решений** должна:
 - позволять сотрудникам отдела маркетинга и продаж запрашивать и использовать данные о клиентах при генерации стандартных отчетов, выполнять специальные (нестандартные) запросы, получать информацию о маркетинговых кампаниях, прогнозе продаж и **сегментации** клиентов, а также обращаться к сторонним источникам данных и инструментам финансовой оценки;
 - представлять все операции клиентам в унифицированном виде, включая повторные контакты, встречи и сделки;
 - позволять сотрудникам отдела маркетинга инициировать новые маркетинговые программы в масштабе всей компании. В **настоящее** время торговые представители не знают, как связать эти программы с **деятельностью** всех подразделений компании для достижения максимального эффекта;
 - определять, анализировать и совместно использовать все виды информации о клиентах во многих отделах.

Упражнение 1. Формулировка задачи

Ознакомьтесь с ситуацией в отделе продаж компании Adventure Works Cycles и сформулируйте задачи **проекта**. Позаботьтесь, чтобы в этом документе описывались бизнес-задачи, и укажите проектной команде общее направление проекта.

Вот несколько примеров формулировки задач.

Возможности продаж не реализуются, потому что не удастся определить **оптимальных** потенциальных клиентов.

Информация о товаре **иногда** оказывается некорректной из-за "кустарного" перевода информации о товаре коммерческими представителями.

Коммерческие представители часто оперируют **неверными** сведениями о цене и данных товара из-за трудностей с загрузкой этой информации.

Коммерческие представители должны самостоятельно строить предположения относительно того, на каких клиентах следует сосредотачивать свои усилия. В результате много времени тратится непродуктивно.

Упражнение 2. Создание общей картины решения

ЗадOCUMENTИРУЙТЕ общую картину для приложения, предназначенного для решения описанной в практикуме ситуации.

Организовать получение корректной информации каждым сотрудником в том момент и в том месте, где она ему нужна, и в течение следующего года увеличить общий объем продаж каждым **торговым** представителем,

Упражнение 3. Разработка целей проекта

На основании представленных данных сформулируйте цели проекта, в том числе бизнес-цели и проектные цели.

Бизнес-цели таковы:

Увеличить продажи в течение следующего года за счет повышения эффективности работы торговых представителей.

Сосредоточить усилия на наиболее перспективных клиентах.

Улучшить методики отслеживания продаж и маркетинга.

Проектные цели таковы:

Усовершенствовать процессы продаж за рубежом за счет поддержки многих иностранных языков.

Предоставить каждому сотруднику отдела продаж только необходимую ему информацию.

Резюме

- Создание общей картины решения позволяет команде четко понять, чего же требуется заказчику.
- Члены команды должны обладать знаниями и опытом, необходимыми для выполнения задачи по разработке решения.
- Область действия определяет, что входит и что не входит в рамки проекта.
- К результатам этапа создания общей картины решения относятся документы: общей картины и области действия решения, оценки риска и структуры проекта.
- Проектная команда также разрабатывает такие документы, как список действующих субъектов и бизнес-правил, а также словарь терминов, используемый во внутренней документации.
- В документе общей картины и области действия решения описаны структура команды и проекта, перечень задач, общая картина решения, область действия проекта, концепции решения, пользовательские профили и цели проекта.
- В документе общей картины описывается продукт, позволяющий решить бизнес-задачу.
- В формулировке задачи определены проблемы, которые будет решать создаваемый продукт.
- Область действия проекта определяет границы применения проекта.
- Область действия проекта основана на предположениях, ограничениях и оценках проекта.
- Концепция решения описывает, как команда планирует достичь целей проекта.
- Пользовательские профили определяют всех вероятных пользователей решения с указанием их ожиданий, целей, рисков и ограничений.
- Бизнес-цели описывают, для чего заказчику необходимо решение.
- Проектные цели представляют параметры решения, которое проектная команда будет разрабатывать.
- Документ структуры проекта содержит информацию об административной структуре команды, стандартах и процессах, а также проектных ресурсах и ограничениях.
- К ключевым компонентам документа структуры проекта относятся:
 - команда и структура;
 - проектные оценки;
 - графики проекта (предварительные версии).
- Документ структуры проекта описывает организационную структуру проекта и процесс руководства проектом, в том числе:
 - роли и обязанности заказчика и проектной команды;
 - решения о связях;
 - логистические решения;
 - решения по управлению изменениями;
 - решения относительно оценки хода проекта.
- Стадия создания общей картины решения завершается контрольной точкой «Утверждение документа общей картины и области действия проекта».

- Команда, ключевые участники и заказчики подтверждают завершение этапа создания общей картины решения одобрением документа общей картины и области действия решения.
- MSF рекомендует оценивать риск постоянно, на протяжении всего проекта.
- В MSF определены *шесть* стадий управления рисками, в течение *которых* команда планирует и претворяет в жизнь стратегии управления рисками и документирует полученные знания для применения в будущих проектах:
 - определение рисков;
 - анализ и определение важности рисков;
 - планирование рисков;
 - отчетность и мониторинг рисков;
 - управление рисками;
 - извлечение уроков.
- На этапе создания общей картины решения команда создает документ оценки риска, в котором описываются и *документируются* все известные риски, а также оценивается их вероятность и влияние. Такой документ содержит следующие разделы:
 - определение рисков;
 - вероятность рисков;
 - серьезность рисков;
 - подверженность рискам;
 - профилактические мероприятия;
 - мероприятия по обеспечению непрерывности бизнеса;
 - ответственный за риск.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия, Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Какова цель создания общей картины решения?
2. Каковы обязанности различных ролей на этапе создания общей картины решения?
3. Каковы результаты этапа создания общей картины решения?
4. Зачем определяют область действия проекта?
5. Чем завершается этап создания общей картины решения?
6. Какие решения относительно управления изменениями регистрируются в документе структуры проекта?
7. В чем различие между бизнес-целями и проектными целями?
8. Перечислите основные рекомендации по созданию пользовательских профилей.
9. Перечислите обязательные компоненты документа оценки риска.
10. Как оценивать риски проекта?

ГЛАВА 4

Создание концептуального дизайна

Занятие 1. Основные сведения об этапе планирования	105
Занятие 2. Основные сведения о функциональных спецификациях	111
Занятие 3. Основные сведения о концептуальном дизайне	117
Занятие 4. Разработка концептуального дизайна	121
Занятие 5. Оптимизация концептуального дизайна	134
Практикум. Анализ требований	139
Резюме	141
Закрепление материала	143

В этой главе

Информации, собранной командой на этапе создания общей картины решения, обычно достаточно для начала работы над проектом. На этой стадии создается базовый документ общей картины и области действия решения. Ближе к концу этапа создания общей картины команда переходит к планированию модели процессов MSF. На этом этапе необходимо убедиться, что решаемая бизнес-проблема понимается полностью и команда в состоянии спроектировать адекватное решение. Кроме того, следует спланировать порядок разработки решения и оценить, достаточно ли для этого ресурсов.

На этапе планирования создается набор моделей и документов с перечнем требований — функциональная спецификация, или черновой план решения. Работа над ним начинается на этапе планирования.

В этой главе рассказывается о целях этапа планирования и трех видах дизайна на данном этапе — концептуальном, логическом и физическом. Вы также узнаете, зачем нужны функциональные спецификации и в чем их преимущества. Кроме того, здесь подробно описан концептуальный дизайн.

Примечание Существует масса методов моделирования бизнес-процессов и ключевых бизнес-операций. В этом учебном курсе речь в основном пойдет о вариантах и сценариях использования системы.

Прежде всего

Для изучения материалов этой главы необходимо:

- знать модели процессов в MSF;
- уметь создавать варианты использования системы (ВИС), сценарии использования системы (СИС) и диаграммы СИС;
- уметь оперировать полученным в результате выполнения этапа создания общей картины документом, описывающим общую картину и область действия решения, структурой проекта и документами по анализу рисков;
- различать разные виды информации: о бизнесе, пользователях, системе и процедурах.

Занятие 1. Основные сведения об этапе планирования

На этом этапе проектная команда описывает решение: что именно следует создавать, как и кто будет этим заниматься, готовятся функциональные спецификации, разрабатывается дизайн и создаются планы работ, смета и календарные планы представления отдельных компонентов решения.

Из этого занятия вы узнаете о целях этапа планирования и обязанностях различных MSF-ролей, а также о том, какие результаты следует обязательно получить на этом этапе.

Изучив материал этого занятия, вы сможете:

- ✓ объяснить цель этапа планирования;
- ✓ описать три стадии и промежуточные роли, актуальные на этом этапе;
- ✓ рассказать об ответственности, возлагаемой на различные MSF-роли на данном этапе;
- ✓ определить стандартные результаты, предъявляемые заказчику на этапе планирования.

Продолжительность занятия — около 5 минут.

Этап планирования

Прежде чем изучать этап планирования и его обязательные результаты, важно понять его место в общем процессе MSF.

Рис. 4-1 иллюстрирует место этапа планирования в модели процессов MSF.

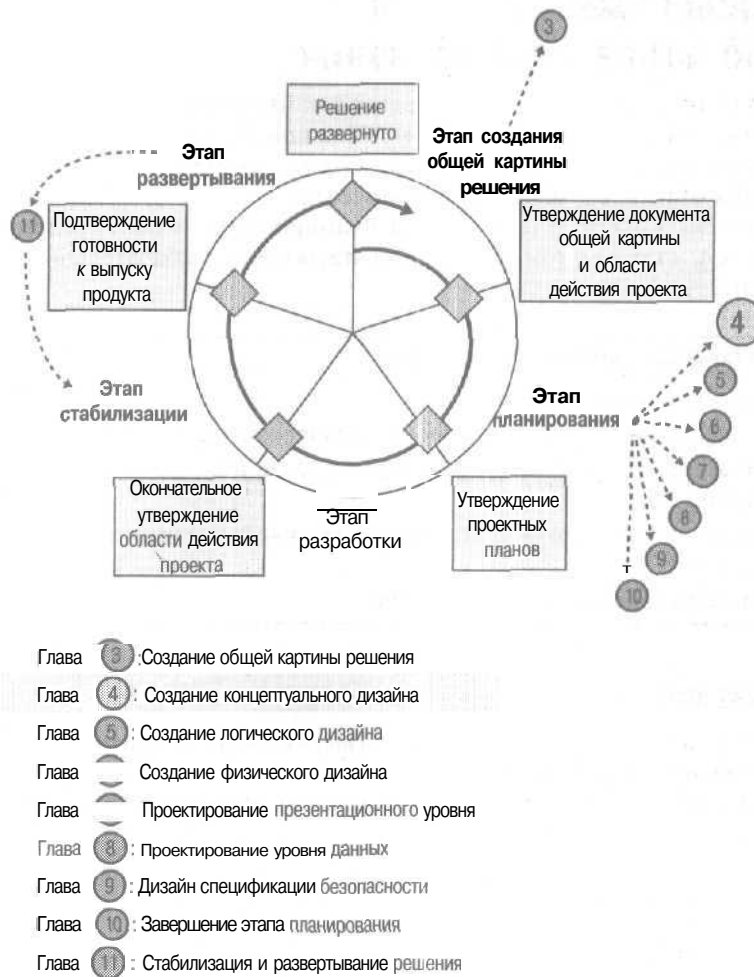


Рис. 4-1. Где мы сейчас находимся в процессе проектирования в MSF

Переход к этапу планирования

На этапе создания общей картины решения команда собирает достаточно информации, чтобы определить *область действия проекта* (project score). Этап планирования вполне можно начать уже на этапе создания общей картины решения, как только собрано достаточно информации для начала ее систематизации и анализа. В этом случае по завершении этапа создания общей картины команда продолжает работу над конкретизацией и анализом его результатов.

На рис. 4-2 показан переход между этими этапами.

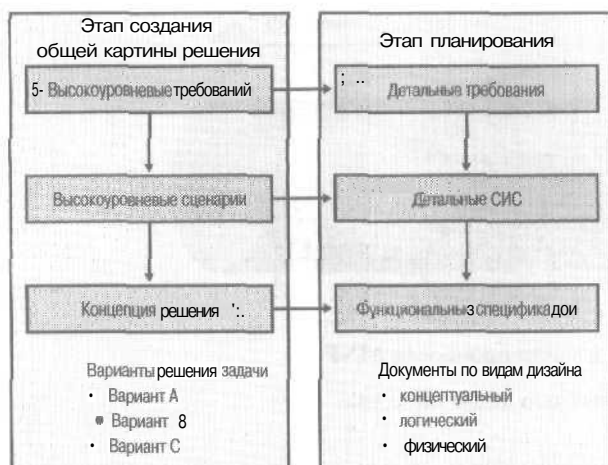


Рис. 4-2. Переход от этапа создания общей картины решения к этапу планирования

Цель этапа планирования

На этапе планирования команда проекта продолжает работу, начатую на этапе создания общей картины решения, а именно: работает над предварительными требованиями, задачами, их последовательностью и профилями пользователей. В результате вы должны выработать архитектуру и дизайн решения, планы по его разработке и развертыванию и календарные графики выполнения задач и загрузки ресурсов. На этом этапе команда составляет как можно более четкую и ясную картину решения. Процесс планирования должен двигать проект вперед, однако многие команды на нем спотыкаются, слишком много времени уделяя планированию. Ключ к успеху — уловить тот момент, когда информации уже достаточно для дальнейшего движения вперед. При недостатке информации рискованно переходить к следующему этапу, с другой же стороны, избыток информации может стать причиной стагнации проекта.

Три типа дизайна: концептуальный, логический и физический

На этапе планирования разрабатываются три вида дизайна: концептуальный, логический и физический (рис. 4-3), причем эти процессы выполняются не параллельно. Они имеют «плавающие» начало и конец и зависят друг от друга. Логический дизайн строится на основе концептуального, а физический — по результатам логического. Любые изменения концептуального дизайна отражаются на логическом дизайне и, в свою очередь, приводят к модификации физического дизайна.

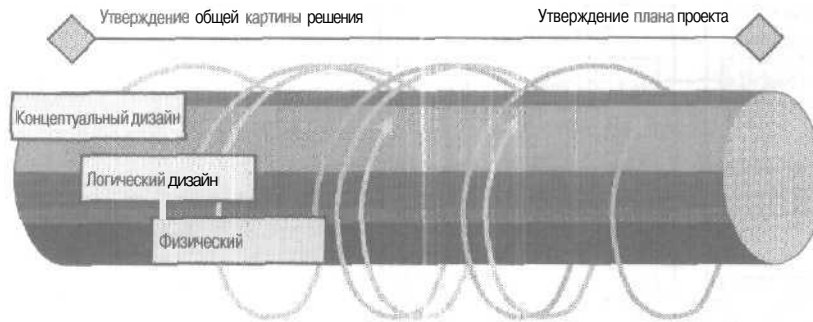


Рис. 4-3. Этап планирования в модели процессов MSF

В таблице 4-1 описаны различные виды дизайна.

Таблица 4-1. Три вида дизайна

Тип дизайна	Представление	Цель
Концептуальный	Представление проблемы с точки зрения пользователя и бизнеса	Описание проблемы и решения в терминах сценариев использования системы
Логический	Представление решения с точки зрения команды проекта	Описание решения как логического набора взаимодействующих сервисов
Физический	Представление решения с точки зрения разработчиков	Описание сервисов и технологий решения

Примечание Концептуальный дизайн можно разбить на более мелкие стадии: исследование, анализ и оптимизацию. О них мы и поговорим в этой главе.

Роли и обязанности членов команды на этапе планирования

На этапе планирования проектная команда работает как единое целое, тем не менее обязанности отдельных ролей отличаются.

- **Менеджеры решения** (product management) — группа сотрудников, отвечающих за то, чтобы план удовлетворял потребностям заказчика. Они заботятся об уточнении требований, анализе текущего состояния бизнеса, оптимизации концепции решения и создании концептуального дизайна.
- **Менеджеры программы** (program management) следят за тем, чтобы выполнить план проекта с применением только имеющихся ресурсов. Эта группа отвечает за весь дизайн, особенно за логический, и за функциональные спецификации, а также за создание плана проекта и календарных графиков и несет ответственность за выполнение этапа планирования.
- **Разработчики** (development) обеспечивают *техническую реализацию* плана, отвечают за логический и физический дизайн решения и включение их в функциональные спецификации. Кроме того, они определяют время и усилия, необходимые для разработки и стабилизации решения.

- **Тестировщики** (testing) отвечают за соответствие плана требованиям решению, оценку дизайна с точки зрения возможности тестирования функций и создание плана и календарного графика тестирования.
- **Менеджеры по выпуску** (release management) оценивают дизайн с точки зрения простоты развертывания, управления и поддержки. Кроме того, они планируют и создают календарные графики развертывания решения.
- **Специалисты по удобству использования решения** (user experience) заботятся о пригодности продукта к использованию. Они отвечают за анализ потребностей пользователей и создание стратегии сопровождения, а также за оценку дизайна с точки зрения удобства использования продукта. Они определяют время и усилия, необходимые для создания системы поддержки пользователей, и тестируют удобство применения всех возможностей, предоставляемых интерфейсом пользователя.

Контрольные точки и результаты этапа планирования

Кульминация этапа планирования — контрольная точка «утверждение плана проекта», в которой проектная команда, заказчик и другие заинтересованные стороны приходят к соглашению о предполагаемых *результатах проекта* (deliverables) и дают заключение, что план удовлетворяет требованиям и гарантирует успех проекта. Далее перечислены результаты, получаемые в этой контрольной точке.

- **Функциональные спецификации (базовые)** описывают, что именно будет представлять из себя продукт. Это итог усилий всей команды. Подробнее о функциональных спецификациях рассказывается на занятии 2.
- **Генеральный план проекта (базовый)** — это набор планов для каждой из шести ролей команды, они ориентированы на обеспечение максимального соответствия функций решения и функциональных спецификаций. Здесь описываются методы, которые команды предполагают применять для выполнения своих задач.
- **Генеральный календарный график проекта (базовый)** определяет временные рамки генерального плана проекта и синхронизирует календарные графики, разработанные разными командами. В нем указывается предполагаемое время выполнения работы отдельными командами. Объединяя отдельные календарные графики, проектная команда создает единый календарный график проекта. Это первый шаг на пути к определению конечной даты выпуска продукта.
- **Обновленный генеральный документ по оценке риска** уже создан на этапе построения общей картины решения и регулярно пересматривается и обновляется, особенно при прохождении контрольных точек. Здесь описываются различные виды риска, *связанного* с разработкой решения. Обычно все риски сортируются для выявления самых опасных из них и их оценки суммируются для получения общей оценки риска.

Все это «живые» документы, активно меняющиеся по мере развития проекта. Однако все вносимые в них изменения должны в обязательном порядке утверждать комиссия, в которую входят пользователи и другие заинтересованные стороны.

На рис. 4-4 показаны результаты этапа планирования.

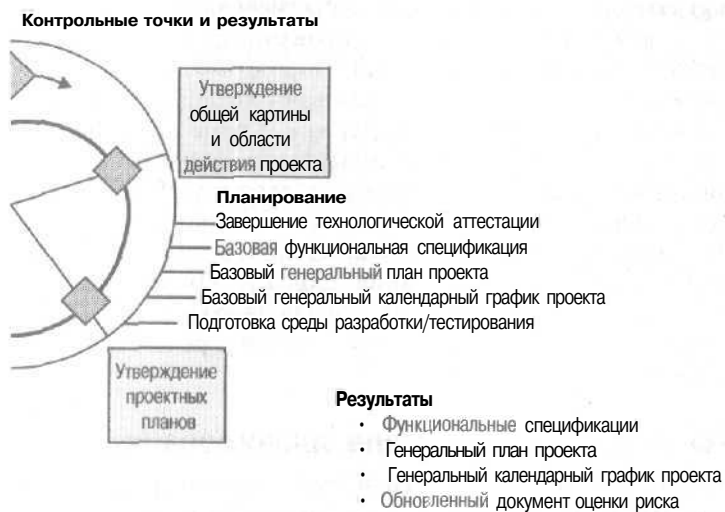


Рис. 4-4. Результаты этапа планирования

Занятие 2. Основные сведения о функциональных спецификациях

На этапе планирования команда проекта от описания задач переходит к проектированию решения. Этап планирования в модели процессов MSF считается логически завершенным по достижении контрольной точки «Утверждение плана проекта». Один из главных результатов, полученных к этому моменту, — функциональные спецификации. Именно они определяют, что, как и когда будет создано. План проекта и календарный график определяют порядок работы над проектом и его контрольные точки. На этом занятии рассказывается о целях и преимуществах функциональной спецификации. Вы также узнаете о рисках при отказе от создания функциональных спецификаций.

Создание спецификаций не имеет ни начальной, ни конечной точек. На этапе планирования в каждом виде дизайна — концептуальном, логическом и физическом вносится вклад в функциональные спецификации. По мере выполнения отдельных задач этапа планирования функциональные спецификации обретают завершенную форму. В соответствии с отраслевыми рекомендациями в конце процесса физического дизайна должна быть готова базовая версия функциональных спецификаций, которые затем постоянно обновляются в процессе разработки.

Изучив материал этого занятия, вы сможете:

- ✓ рассказать о назначении функциональной спецификации;
- ✓ описать цели и преимущества создания функциональных спецификаций;
- ✓ перечислить различные виды рисков, которые возникают в случае отказа от создания функциональных спецификаций;
- ✓ описать содержание функциональных спецификаций;

Продолжительность занятия — около 10 минут.

Что такое функциональные спецификации

Функциональные спецификации (functional specification) — это виртуальное хранилище проекта и связанных с проектом артефактов, которые создаются на этапе планирования в модели процессов MSF. Артефакты представляют собой результат процессов концептуального, логического и физического дизайна, выполняемых на этапе планирования. К артефактам относятся UML-модели, такие, как диаграммы ВИС и СИС, предварительные требования (уточняемые по ходу проекта), предварительный набор функций и различные информационные модели.

Примечание UML — стандартный язык моделирования, применяемый для документирования проектов, где выполняется объектно-ориентированное проектирование. Подробнее об UML рассказывается в книгах Мартина Фаулера (Martin Fowler) и Кендалла Скотта (Kendall Scott) «UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd edition» (Addison-Wesley, 2000) и Дуга Розенберга (Doug Rosenberg) и Кендалла Скотта (Kendall Scott) «Use Case Driven Object Modeling with UML: A Practical Approach» (Addison-Wesley, 1999).

Важно понимать, что функциональные спецификации виртуальны по своей природе. Многие артефакты концептуального, логического и физического дизайна — это документы в электронной форме или данные, хранимые в различных базах данных. Функциональные спецификации можно представлять в различных формах — электронных или бумажных; в текстовом или графическом формате, например в документах Microsoft Word или презентациях Microsoft PowerPoint. Поэтому не так-то легко соединить все артефакты в едином физическом документе или ином единообразном виде. Функциональные спецификации — не просто результат работы одного человека или роли, а плод совместных усилий нескольких ролей в команде.

Функциональные спецификации описывают область действия текущей версии решения перечислением того, какие функции оно должно поддерживать. Исключенные функции лучше оставить в документе об общей картине решения и области действия проекта; они, возможно, будут реализованы в будущих версиях решения или в качестве пожеланий или как элементов, выходящих за рамки решения. В функциональных спецификациях регистрируются принятые решения и соглашения, касающиеся набора функций, интерфейса, дизайна и приоритетов.

Функциональные спецификации — это форма, в которой группе разработчиков передаются результаты труда группы, ответственной за проектирование. Кроме того, функциональные спецификации необходимы группе тестирования для создания сценариев и планов тестирования, тестовых данных, а также для проверки требований к аппаратному обеспечению для решения.

Цели функциональных спецификаций

- **Гарантируют единое понимание требований бизнеса и пользователей.** Функции решения зависят от требований бизнеса и пользователей, для удовлетворения которых оно и создается. В небольших проектах требований немного, и их легко документировать. В больших и сложных проектах объем и сложность требований возрастают. Функциональные спецификации помогают заказчикам и проектной команде прийти к согласию относительно требований к решению.
- **Позволяют разбить задачу на более мелкие части и представить решение в виде логических модулей.** Чтобы добиться успеха в сложных проектах, необходимо четко определить все аспекты задачи, а также разбить решение на отдельные, четко определенные части. Функциональные спецификации упрощают решение за счет представления его в виде логических частей и тщательного документирования, а также помогают команде вносить изменения в дизайн на ранних стадиях процесса разработки. Внесение изменений в решение на этой стадии менее рискованно и обойдется в конечном итоге дешевле, чем при корректировке на более поздних стадиях процесса.
- **Обеспечивают среду для планирования, составления календарных графиков и построения решения.** Функциональные спецификации позволяют менеджеру программы определять задачи команды, составлять смету и бюджет всего проекта, а также корректно оценить ресурсы и время, необходимые для осуществления проекта и создать план и календарный график проекта. Тестирующим функциональные спецификации необходимы для создания тестовых данных и сценариев на начальных стадиях жизненного цикла проекта. Менеджеры по выпуску используют функциональную спецификацию для развертывания решения и поддержки сред разработки и тестирования.

- Служат соглашением между командой и заказчиком, в котором четко оговариваются все особенности продукта. В большинстве организаций и в большинстве проектов функциональные спецификации выполняют роль своего рода договора между командой и заказчиком — это задокументированное описание того, что именно следует разработать и передать заказчику. Функциональные спецификации не всегда представляют собой юридический документ, хотя это и не исключено. Если в проект вовлечены сторонние команды или организации, функциональные спецификации служат дополнением к наряду на работу по проекту.

Риски, возникающие в отсутствие функциональных спецификаций

Иногда ограничения (времени или бюджета) заставляют команду отказаться от создания и использования функциональных спецификаций. Команда может перейти к этапу разработки в модели процессов MSF и без них, но в этом случае риск провала проекта значительно возрастает. Далее перечислены некоторые виды возможных рисков, возникающие в связи с отказом от создания функциональных спецификаций:

- разработанное решение не полностью удовлетворит требования заказчика;
- члены команды не до конца уяснят, что ждет заказчик, и представления заказчика и подрядчика о продукте не совпадут. Вследствие этого теряется уверенность в том, что разрабатываемое решение — как раз то, что нужно заказчику;
- информации, которой располагает команда, недостаточно, чтобы проконтролировать качество решения и его соответствие ожиданиям заказчика;
- менеджеру проекта не удастся точно рассчитать бюджет и график проекта. Информация, содержащаяся в функциональных спецификациях, помогает команде оценить усилия и набор навыков, необходимых для разработки решения.

На рис. 4-5 показаны различные риски, возникающие при отказе от использования функциональных спецификаций для разработки решения.

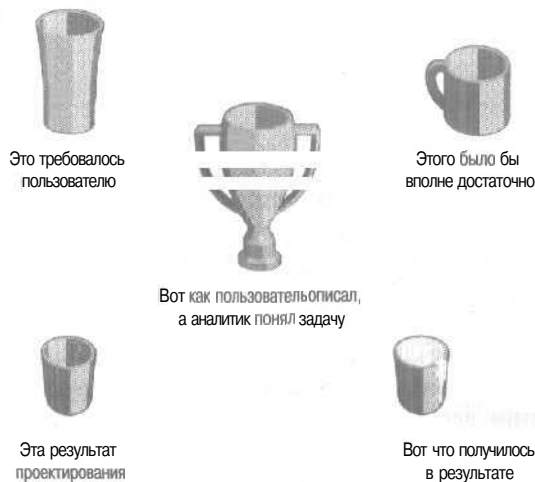


Рис. 4-5. Связь между функциональной спецификацией и результатом

Элементы функциональных спецификаций

Далее перечислены элементы функциональных спецификаций, каждый из которых представляет собой отдельный документ.

- **Краткое описание концептуального дизайна** сжато описывает общий вид решения и его архитектуру. В функциональных спецификациях используются следующие артефакты концептуального дизайна:
 - варианты использования системы;
 - сценарии использования системы;
 - контекстные модели, такие, как снимки экранов существующих систем и фотокопии руководств пользователя.

Эти артефакты могут существовать в разных формах. Например, контекстные модели представляют собой снимки экранов существующей системы или фотокопии руководств пользователя или отчетов; варианты использования системы хранятся в соответствующей базе данных, а концептуальные прототипы пользовательского интерфейса существуют в электронном виде.

- **Краткое описание логического дизайна** содержит информацию о пользователях, объектах и атрибутах. В функциональную спецификацию включаются следующие артефакты этапа логического дизайна:
 - модели задач и последовательностей задач;
 - модель логических объектов и сервисов;
 - концептуальные модели предлагаемого решения;
 - последовательности экранов пользовательского интерфейса;
 - логическая модель базы данных;
 - архитектура системы.
- **Краткое описание физического дизайна** содержит информацию из его ключевых разделов, например посвященных приложению и инфраструктуре. В функциональные спецификации включаются следующие артефакты этапа физического дизайна:
 - распределение сервисов по компонентам;
 - топология распределения компонентов;
 - правила применения технологий;
 - архитектура и строение инфраструктуры;
 - описание экранов пользовательского интерфейса;
 - физическая модель базы данных.
- **Стандарты и процессы** — этот раздел содержит информацию о стандартах и процессах, используемых командой при выполнении проектных задач, а также сведения о метриках качества и эксплуатационных характеристиках решения, которые необходимы для дальнейшей работы. Эти метрики определяются во время тестов и помогают достичь определенных в требованиях целей.

Перечисленные далее разделы включаются в функциональные спецификации при необходимости и в зависимости от размеров и области действия проекта. Чтобы избежать рассинхронизации версий, рекомендуется использовать ссылки на уже существующие документы, а не воспроизводить содержащуюся в них информацию. В частности, не следует включать в функциональные спецификации информацию, которая уже зафиксирована в документе об общей картине и области действия проекта или в документе оценки риска.

- **Краткое описание общей картины и области действия проекта** резюмирует возможности бизнеса, концепцию решения и область действия проекта в соответствии с документом общей картины и области действия проекта.
- **История проекта** содержит перечень важных событий и решений, которые были приняты к текущему моменту и способствовали продвижению проекта до его текущего состояния. Историческая информация позволяет гарантировать одинаковое понимание проекта командой проекта и заказчиком. Эта информация также помогает самым тщательным образом разобраться в деталях проекта.
- **Краткое описание функциональных спецификаций.** Оно полезно заказчикам и новым членам команды. Первая версия этого раздела создается по завершении сбора и анализа требований пользователей и создания дизайна пользовательского интерфейса. В дополнение к ссылкам на другие документы, из которых собственно и состоят функциональные спецификации, этот раздел обычно содержит снимки экранов, иллюстрирующие предполагаемый набор функций решения.
- **Краткое описание требований** — сюда включают требования, определенные пользователями, системой, процедурами и бизнесом. В изложении требований бизнеса описывается, чего заказчик, пользователи и действующие субъекты ждут от решения. В описании требований пользователей перечислены сценарии использования системы и указано, кто, когда и как будет работать с продуктом. В «выжимке» системных требований определяется зависимость от систем и сервисов. В резюме операционных требований описываются особенности реализации безопасности, управляемости и поддержки, а также сопровождения.
- **Краткое описание сценариев и вариантов использования системы.** В этом разделе представлены основные положения документа, описывающего СИС. Здесь содержится краткое изложение всех ключевых ВИС, указанных в документе.
- **Допущения и зависимости.** В этом разделе перечислены принятые в проекте допущения и зависимости. Пример зависимости — набор технических навыков, необходимых для разработки решения, а допущения — предположение о том, что платформой для развертывания решения станет семейство ОС Microsoft Windows. Необходимо также создать тесты для проверки и подтверждения этих допущений.
- **Краткое описание стратегии защиты.** В этом разделе описывается стратегия обеспечения безопасности, которая определяет архитектуру решения. В физическом дизайне указаны детали реализации безопасности для каждой функции и компонента в отдельности. Здесь приводится конспект стратегии безопасности вместе со ссылками на план обеспечения безопасности.
- **Краткое описание требований к установке.** Это резюме требований к среде, в которой предполагается устанавливать решение. Оно основано на подразделе плана развертывания решения, посвященного установке. Сведения о том, как эти требования предполагается учитывать, содержатся в физическом дизайне.
- **Краткое описание требований к удалению** содержит информацию о порядке удаления решения, то есть что надо учитывать перед и в процессе удаления решения. Сюда же помещается информация о том, резервное копирование каких данных следует выполнить до удаления решения, чтобы гарантировать возможность последующего безопасного восстановления и модернизации.
- **Краткое описание требований по интеграции.** Это раздел об интеграции и способности взаимодействия (interoperability) с другими системами, а также свя-

занные с этим цели проекта. Здесь размещается сжатый план перехода к использованию данного решения. Сведения о порядке интеграции, содержатся в физическом дизайне.

- **Краткое описание поддержки и сопровождения** — конспект требований по поддержке и сопровождению и связанных с этим целей проекта. Эта информация есть в плане процедур и плане поддержки, Как именно в решении будут обеспечиваться поддержка и сопровождение, описано в физическом дизайне.
- **Краткое описание правовых требований** — перечень всех требований, обусловленных местным законодательством. Правовые требования, как правило, определяются корпоративной политикой заказчика и государственными органами, регулирующими отрасль, в которой работает заказчик.
- **Краткое описание риска.** Здесь вкратце рассказывается о различных видах риска, которые могут повлиять на разработку и поставку решения. Для каждого вида указана вероятность его реализации и мероприятия для предотвращения опасностей.
- **Ссылки.** В этом разделе описываются все внутренние или внешние ресурсы, содержащие дополнительную информацию, которой нет в функциональных спецификациях.
- **Приложения** — это набор результатов дизайна, которые команда использует для создания функциональных спецификаций. Этот набор содержит дополнительную информацию, касающуюся концептуального дизайна, такую, как результаты опросов и профили пользователей, а также сведения о физическом дизайне, например существующие конфигурации клиента и сервера.

Занятие 3. Основные сведения о концептуальном дизайне

Этап планирования в модели процессов MSF включает разработку трех видов дизайна — концептуального, логического и физического. Концептуальный дизайн начинается на этапе создания общей картины решения и продолжается на протяжении всего этапа планирования. Поскольку процесс разработки дизайна в MSF эволюционный и итеративный, концептуальный дизайн ложится в основу как логического, так и физического дизайна.

На этом занятии рассказывается о концептуальном дизайне, а также о его целях и преимуществах.

Изучив материал этого занятия, вы сможете:

- ✓ дать определение концептуального дизайна;
- ✓ описать цели концептуального дизайна;
- ✓ описать стадии концептуального дизайна;
- ✓ описать «исследовательскую» стадию концептуального дизайна.

Продолжительность занятия — около 10 минут.

Что такое концептуальный дизайн

Концептуальный дизайн — это процесс сбора, анализа и определения приоритетов особенностей бизнеса и точек зрения пользователей на проблему и будущее решение, с последующим созданием высокоуровневого представления решения.

Разработка требований

Во время сбора информации собираются предварительные требования. Очень важно, чтобы команда понимала разницу между различными категориями требований: *пользовательскими, системными, процедурными* и *бизнес-требованиями*. Предварительные требования обычно формулируют на основе начальных интервью и другой информации, собранной на тот момент. По мере углубления понимания проблемы бизнеса предварительные требования расширяют и уточняют. Эти так называемые *требования-кандидаты* (candidate requirements) затем переводят в формат требований.

Отображение требований путем моделирования

Чтобы создать точный и пригодный к использованию концептуальный дизайн решения, необходим эффективный метод для представления и обсуждения решения с пользователями. Для этого создаются модели задач проекта. Один из способов моделирования таких задач и их последовательностей — построение вариантов и сценариев использования системы.

Дополнительная информация Очень подробно о создании вариантов использования системы рассказывается в книге Алистера Кокберна (Alistair Cockburn) «Writing Effective Use Cases» (Addison-Wesley, 2001).

В качестве примера рассмотрим действия команды, которой поручено спроектировать Web-сайт для компании, занимающейся электронной коммерцией. В процессе выяснения требований к Web-сайту члены команды заинтересовались, какие линейки товаров предполагается продавать через сайт и как посетители будут им пользоваться. Для ответа на эти вопросы необходимо изучить каталог товаров, порядок его поддержки, а также проанализировать все возможные действия посетителей на сайте. Команда формулирует набор задач и описывает их в виде вариантов использования системы. Для каждого описанного заказчиком действия архитектор создает детальный сценарий использования системы. Здесь же фиксируются все возможные исключения и альтернативные способы выполнения задач.

Концептуальный дизайн в модели процессов MSF

Согласно модели процессов MSF, концептуальный дизайн создается на этапе планирования. Но команда вправе начать работу над ним уже на этапе создания общей картины решения. Создание концептуального дизайна можно начинать, когда накоплено достаточно информации для дальнейшего сбора и уточнения требований.

Рис. 4-6 иллюстрирует место концептуального дизайна в общей модели процессов MSF.

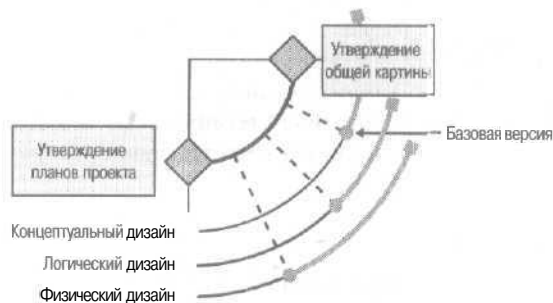


Рис. 4-6. Место концептуального дизайна в модели процессов MSF

Как вы помните, процесс дизайна итеративный, поэтому процессы создания логического и физического дизайна могут перекрываться с процессом концептуального дизайна. Эти три этапа не параллельны; у них различные начальные точки и базовые версии. Благодаря итеративному подходу создания концептуальный дизайн можно изменять по результатам логического и физического дизайна.

Цели концептуального дизайна

Без концептуального дизайна вы можете создать превосходный продукт, но... для решения не той задачи. Далее перечислены основные цели концептуального дизайна.

- **Понять бизнес-проблему, которую предстоит решить.** Концептуальный дизайн позволяет понять решаемую проблему и усовершенствовать бизнес-процессы. Цель заключается в уточнении, документировании и проверке того, что пользователи и заказчик ждут от решения.

- **Понять требования бизнеса, заказчика и конечных пользователей.** Концептуальный дизайн помогает проектной команде определить требования к проекту в определенном контексте и сформулировать «представление» решения, ориентированное как на процессы, так и на пользователей. Подобное представление не ограничивается списком желаемых функций, а охватывает более широкий контекст бизнес-процессов и операций.
- **Описать состояние предприятия, которое предполагается достичь в будущем.** Концептуальный дизайн также позволяет определить будущее состояние бизнес-операций. Проект будущего состояния бизнеса необходим для следующих этапов разработки дизайна.

В процессе создания концептуального дизайна проектная команда пытается понять контекст проблемы, зафиксировать на бумаге бизнес-операции и границы и взаимосвязи. На стадии концептуального дизайна не создается полный набор функциональных спецификаций, тем не менее этот дизайн необходим для начала работы над функциональными спецификациями.

В таблице 4-2 детально описывается область действия концептуального дизайна.

Таблица 4-2. Область действия концептуального дизайна

Чем не является концептуальный дизайн	В чем помогает концептуальный дизайн
Полным набором функциональных спецификаций	Начать работу над функциональными спецификациями
Определением компонентов, составляющих систему	Выделить отдельные задачи общей бизнес-проблемы, для решения которых будут в конечном итоге создаваться отдельные компоненты
Технологическим решением	Зафиксировать бизнес-операции, их границы и взаимосвязи

Стадии концептуального дизайна

Концептуальный дизайн состоит из трех шагов и соответствующих им *базовых версий* (baseline). Создание этого дизайна является итеративным процессом, и при необходимости цикл может повторяться.

1. Исследование. На этой стадии выполняются следующие задачи:

- получение ответов на ключевые вопросы;
- определение ключевых бизнес-процессов и операций;
- распределение процессов и операций по приоритетам;
- проверка, уточнение и расширение предварительных требований, вариантов и сценариев использования системы, созданных на этапе создания общей картины решения.

2. Анализ предусматривает:

- проверку результатов исследования бизнеса и пользователей;
- уточнение требований-кандидатов;
- документирование и моделирование контекста, рабочих потоков, последовательностей выполнения задач и связей со средой.

3. Оптимизация состоит из:

- оптимизации концепции решения, созданной на этапе создания общей картины решения;
- проверки и тестирования усовершенствованных бизнес-процессов.

На основе базовой версии результатов оптимизации создается базовая версия концептуального дизайна. Рис. 4-7 иллюстрирует стадии концептуального дизайна.

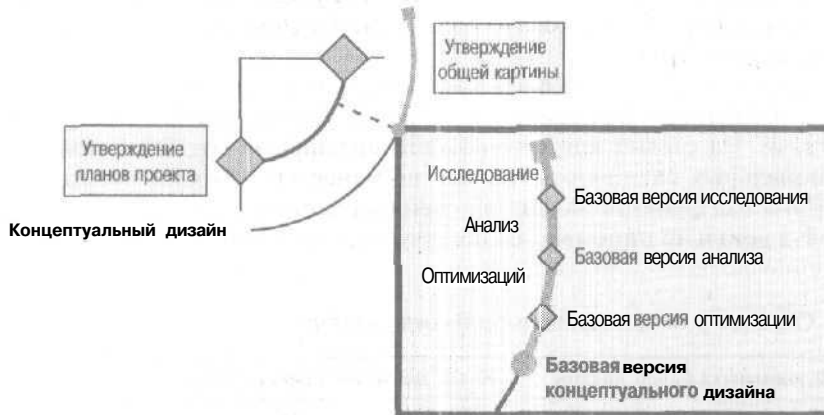


Рис. 4-7. Стадии концептуального дизайна

На стадии исследования команда собирает дополнительную информацию для уточнения и проверки данных, собранных на этапе создания общей картины решения. Эти данные в основном высокоуровневые, и в них недостает деталей. На первой стадии концептуального дизайна необходимо выявить детали. Например, команда сначала определяет вопросы, возникшие после первой итерации сбора информации; затем уточняет задачи, бизнес-процессы и потоки работ. По мере выявления важных деталей результаты объединяются в виде вариантов использования системы и предварительных требований.

Примечание Стадия анализа подробно описывается на занятии 4, а стадия оптимизации — на занятии 5.

Занятие 4. Разработка концептуального дизайна

После сбора детальной информации о требованиях бизнеса и пользователей и бизнес-процессах команда переходит к стадии анализа концептуального дизайна, на которой анализируются, детализируются и уточняются артефакты, созданные на этапе создания общей картины.

На этом занятии рассказывается, как изменяется формулировка требований по мере уточнения собранной информации. Вы научитесь классифицировать требования по различным категориям и уточнять ВИС и СИС, а также выбирать высокоуровневую архитектуру решения.

Изучив материал этого занятия, вы сможете:

- ✓ описать стадию анализа концептуального дизайна;
- ✓ модифицировать формулировку требований;
- ✓ классифицировать требования по категориям: пользовательские, системные, процедурные и бизнес-требования;
- ✓ уточнять диаграммы ВИС;
- ✓ выбирать подходящую прикладную архитектуру для создаваемого решения.

Продолжительность занятия — около 20 минут.

Стадия анализа в концептуальном дизайне

На этой стадии синтезируется информация, собранная на стадии исследования, и создаются детальные сценарии использования системы. Цели анализа таковы:

- изучение перечня пользователей, бизнес-процессов и операций;
- документирование и моделирование контекстов, потоков работ, последовательностей задач и бизнес-среды.

Задачи стадии анализа

На этой стадии выполняются следующие задачи:

- синтез информации;
- уточнение диаграмм ВИС;
- выбор подходящей прикладной архитектуры для решения;
- создание концептуальной модели решения.

Синтез информации — это процесс переработки собранных данных и интерпретация полученных результатов. Собранные данные трансформируются в значимую информацию. В процессе синтеза данных проектная команда выполняет следующие действия:

- выделяет из сказанного и выполняемого пользователями отдельные значимые фрагменты информации;
- записывает в детализированном виде потоки задач, выполняемых пользователями;
- определяет, какие инструментальные средства и фрагменты данных использованы;
- определяет исключения и альтернативы, возникающие в процессе выполнения пользователями своих задач;

- моделирует связи между бизнес-процессами, бизнес-системами и пользователями;
- моделирует существующее окружение пользователя и все его возможные изменения.

Результаты стадии анализа

В таблице 4-3 перечислены ключевые задачи и результаты стадии анализа.

Таблица 4-3. Задачи и результаты стадии анализа концептуального дизайна

Задача	Результат
Синтез собранной информации	Информационные модели: <ul style="list-style-type: none"> • связи между бизнес-процессами, бизнес-системами и пользователями; • потоки работ (workflow); • последовательности задач Обновленные профили пользователей Требования-кандидаты Детализированные ВИС
Создание сценариев использования системы	Действующие СИС

Обновление формулировки требований

В процессе обновления требований следует руководствоваться рядом критериев.

- Требования должны быть *четко определены* (well defined). Четко определенное требование — это законченное предложение, содержащее глагол «будет», «может», «должен» или «следовало бы».
- Требования должны быть *краткими*. Каждое требование должно относиться строго к одному элементу.
- Требования должны быть *проверяемыми*. Для определенных входных данных требование должно давать определенный результат.
- Требования должны *организовываться в иерархию зависимых требований*. Вам нужно сгруппировать связанные требования вместе под одним требованием более высокого уровня, чтобы получить набор функций.
- Требования должны создаваться на языке бизнеса без использования жаргона и аргю.

В таблице 4-4 показаны возможные требования, собранные и сформулированные проектной командой на основании текстов интервью на этапе создания общей картины решения.

Таблица 4-4. Предварительные формулировки требований на этапе создания общей картины решения

Номер требования	Требование
i	Определить лучших клиентов в разрезе продуктов и регионов (анализ прибылей и географический анализ). Именно на них должны концентрировать свои усилия менеджеры по продажам
2	Определить, когда конкретные клиенты покупают меньше
3	Определить лучших клиентов
4	Определить лучших покупателей

На стадии анализа команда модернизирует формулировку требований, добиваясь их краткости, законченности и проверяемости. При уточнении требований команда может обнаружить новые требования. Обновленные требования из таблицы 4-4 показаны в таблице 4-5. Теперь они краткие, законченные по форме, и их легко проверить. Несколько связанных требований организованы в логическую иерархию.

Таблица 4-5. Переформулированные требования

Номер требования	Требование
1.1	Возможность анализировать данные клиента
1.1.1	Возможность анализировать уровень прибыли по каждому продукту
1.1.2	Возможность анализировать уровень прибыли по каждому клиенту
1.1.3	Возможность анализировать уровень прибыли по каждому региону
1.2	Возможность сортировать клиентов (по убыванию/возрастанию)
1.3.1	Возможность сортировать клиентов по объему продаж (по убыванию/возрастанию)
1.3.2	Возможность сортировать клиентов по объему продаж и отдельным товарам (по убыванию и по возрастанию)
1.3.3	Возможность сортировать клиентов по объему продаж в регионах и объему продаж за определенный период времени (по убыванию/возрастанию)
1.3.4	Возможность сортировать клиентов по объему продаж за определенный период времени (по убыванию/возрастанию)
1.4	Возможность обнаружить тенденции продаж
1.4.1	Возможность обнаружить падение продаж
1.4.2	Возможность обнаружить падение продаж для каждого клиента

Примечание Помните, что проверять требования на каждом шаге следует вместе с заказчиком.

Классификация требований

После уточнения требования подразделяют по категориям: пользовательские, системные, процедурные и бизнес-требования.

Пользовательские требования

Описывают нефункциональные аспекты взаимодействия пользователя с системой и помогают определить вид **пользовательского** интерфейса, а также ожидания в плане надежности, готовности и доступности. Кроме того, они помогают выяснить, какое обучение понадобится пользователям, чтобы эффективно принимать решение. Успешное решение удовлетворяет как требованиям организации к технологической стороне вопроса, так и требованиям пользователей к работе с этими технологиями.

Примеры пользовательских требований

- Менеджеры по продажам не должны при каждой новой продаже вводить имена, которые уже применялись ранее.
- Кассир должен иметь **возможность** совершать более одной транзакции в минуту.
- Клиент должен иметь возможность совершить покупку через Web-сайт в течение пяти минут.

Системные требования

Они определяют атомарные транзакции и их последовательности в **системе**, а также помогают проектной команде определить, как новое решение будет взаимодействовать с существующими системами. Команда также устанавливает критические зависимости от внешних систем, которые необходимо учесть в процессе работы. Перед созданием нового решения команде следует разобраться в существующей в организации инфраструктуре. Это поможет спроектировать и разработать решение, развертывание которого вызовет минимальные нарушения рабочего цикла.

Примеры системных требований

- Все **бизнес-приложения** масштаба предприятия, **поддерживающие** уведомление пользователей в режиме, близком к режиму реального времени, должны содержать утвержденный руководством компонент уведомления, иначе они не будут одобрены во время финального обзора дизайна.
- Система не должна требовать от пользователей иных реквизитов, кроме предоставленных ими в процессе регистрации в корпоративной сети.

Процедурные требования

Процедурные требования описывают, что должно предоставлять решение для обеспечения максимальной производительности и улучшения обслуживания вместе с сокращением времени простоя и снижением риска. Это относится к **следующим** ключевым элементам:

- безопасности;
- **производительности** и надежности;
- управляемости;
- масштабируемости;
- удобству поддержки.

Примеры операционных требований

Пусть заказчик — музыкальный магазин, руководство которого хочет обзавестись **Web-сайтом** электронной коммерции. Далее перечислены некоторые из операционных требований к этому Web-сайту.

- **Производительность и надежность.** Клиент должен иметь доступ к сайту и возможность работы с его ресурсами в любое время в рамках оговоренного уровня сервиса.
- **Масштабируемость и гибкость.** Система должна обрабатывать постоянно изменяющееся количество запросов пользователей и транзакций. Кроме того, сайт следует спроектировать с возможностью модернизации и обновления в будущем без нарушения его доступности или производительности. Следует обеспечить масштабируемость и гибкость как инфраструктуры, так и бизнес-процессов.
- **Возможность управления производительностью.** Архитектура сайта должна предусматривать систему для управления общей пропускной способностью системы и временем отклика в рамках оговоренных уровней сервиса.
- **Высокий уровень безопасности.** Данные, сервисы и устройства в системе необходимо защитить от несанкционированного доступа. Система также должна поддерживать аутентификацию и безопасные транзакции.
- **Возможность административного управления.** Сайт должен поддерживать как локальное, так и удаленное администрирование.
- **Возможность восстановления после сбоя.** Следует предусмотреть возможность восстановления сайта после критического сбоя без серьезных последствий или в рамках оговоренного уровня сервиса.

Бизнес-требования

Бизнес-требования описывают потребности заказчика и ожидания, которые он связывает с решением, а также определяют, какие выгоды принесет заказчику использование решения и как оно поможет успешно вести бизнес. В процессе определения бизнес-требований следует рассматривать организацию как объект со своим собственным набором потребностей, которые создаваемое решение призвано удовлетворить. Эти требования существуют на уровне высшего руководства, принимающего решения, и обеспечивают контекст, в котором будет функционировать решение.

Примеры бизнес-требований

- Сотрудники центра обработки вызовов должны иметь доступ к таким сведениям, как длительность последнего звонка, текущего звонка, а также средняя длительность звонка для каждого телефонного оператора.
- При покупке значительного количества товара кассиры вправе изменять цену в определенных пределах без визы менеджера.
- Решение следует спроектировать, разработать и развернуть в максимального сжатые сроки.
- Система должна успешно взаимодействовать и обмениваться информацией с другими бизнес-процессами, приложениями и источниками данных.

В качестве примера рассмотрим музыкальный магазин, руководство которого желает увеличить продажи компакт- и DVD-дисков и расширить свой сектор рынка, для чего и собирается создать Интернет-магазин. Вот некоторые из предъявляемых в этом случае бизнес-требований:

- после отправки заказа пользователем соответствующий товар должен отмечаться как «принятый к исполнению» и удаляться из категории «доступно для заказа»;
- приложение должно поддерживать применение скидки к заказам, объем которых превышает определенный предел.

Уточнение диаграмм вариантов использования системы

На этапе создания общей картины проектная команда создает диаграммы для всех высокоуровневых ВИС в организации заказчика. Назначение этих диаграмм — перечислить ключевые ВИС, определить область действия решения и сформировать основу для создания концепции решения. Чтобы построить концептуальную модель дизайна, необходимо уточнить варианты использования системы, попадающие в область действия решения, на основе информации, полученной на стадии исследования процесса концептуального дизайна.

В процессе уточнения диаграмм ВИС выполняются следующие задачи:

- создаются подчиненные ВИС;
- создаются СИС для каждого подчиненного ВИС;
- выполняется проверка всех ВИС и СИС на основании исходных интервью, прочей документации, а также с привлечением пользователей;
- уточняются требования на основании информации проверенных и скорректированных ВИС и СИС.

Создание подчиненных ВИС

При создании подчиненных вариантов использования системы приходится вновь обращаться к каждому ВИС на диаграмме, который попадает в область действия проекта. Можно определить все задачи, связанные с ВИС, и моделировать их как подчиненные по отношению к высокоуровневому ВИС. Кроме того, определяются действующие субъекты, выполняющие конкретные задачи, и связи между задачами и действующими субъектами.

На уточненной диаграмме ВИС (рис. 4-8) показаны многие подчиненные ВИС для варианта использования «Управление заказами».

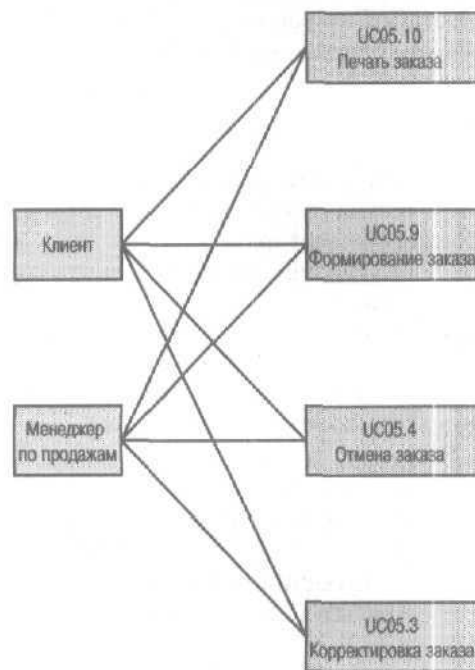


Рис. 4-8. Уточненная диаграмма варианта использования системы

Сценарии использования системы для подчиненных ВИС

Когда подчиненные ВИС готовы, переходят к созданию для них сценариев использования системы. Эта стадия предусматривает детализацию исходного СИС и добавление детализированного словесного сценария, указание основного направления хода событий, альтернативного хода событий (если таковой есть) и описания предварительных и постусловий.

Далее приведен пример сценария использования системы высокоуровневого ВИС для компании Adventure Work Cycles.

Название ВИС: Заказ спецификаций товара

Сокращенное название: Заказ спецификаций товара

Идентификатор ВИС; UC05.1

Идентификатор требования: 15.1

Цель: Предоставить клиенту полную (а не базовую) версию документа спецификации продукта.

Текстовое описание сценария: Просматривая описание товара в каталоге, клиент запрашивает более подробную информацию (спецификацию) о нем.

Действующие субъекты: Клиент

Предварительные условия: Клиент просматривает каталог товаров. Клиент просматривает товар в каталоге.

Основной вариант развития событий:

1. ВИС начинается, когда клиент щелкает ссылку "Заказать спецификацию этого товара".
2. Клиент выбирает формат спецификации.
3. Клиент выбирает способ доставки,
4. Выполняется подтверждение адреса.
5. ВИС завершается, когда заказ заполнен, отправлен на сервер и готов к исполнению.

Альтернативный вариант развития событий:

Постусловия: Заказ заполнен, отправлен на сервер и готов к исполнению, Клиент вернулся к предыдущей странице «Просмотр продуктов».

ВИС, которые используют или расширяют данный: «Клиент обновляет профиль», «Клиент создает профиль», «Отправка заказа на сервер».

Требования пользователя к реализации: Отсутствуют

Частота возникновения: Наблюдается в 17 из 100 сеансов работы клиентов, использующих Web.

Нерешенные задачи: Отсутствуют

Ответственный: Майк Дансельйо

История изменений:

Дата: 6 ноября 2002 г.

Автор: Хейди Стин

Описание: Начальная версия

Один из подчиненных **ВИС** для варианта «Заказ спецификаций товара» — «Заказ спецификаций товара по почте». Сценарий использования системы для этого **ВИС** выглядит так:

Название ВИС: Заказ спецификаций товара по почте

Сокращенное название: Заказ спецификаций товара по почте

Идентификатор ВИС: UC05.1.1

Идентификатор требования: 15.1.1

Цель: Предоставить клиенту возможность получить по почте спецификации выбранного из каталога товара.

Текстовое описание сценария: Клиент хочет получить спецификацию продукта по почте. Клиент просматривает каталог (UC05.1) и находит интересующий его товар. Если клиент *выбрал* почту в качестве способа доставки, то он должен указать адрес. Если клиент вошел в систему под *действующей* учетной записью, он сможет выбрать адрес из списка в своем профиле. Клиент также вправе указать адрес в форме. После *отправки* запроса клиент получает номер для *последующего* контроля заказа.

Действующие субъекты: Клиент

Предварительные условия:

- Клиент просматривает каталог продуктов.
- Клиент просматривает товар в каталоге.

Основной вариант развития событий:

1. ВИС начинается, когда клиент щелкает ссылку «Заказать спецификации этого товара».
2. Клиент выбирает формат спецификации, например лист спецификаций или брошюру.
3. Клиент выбирает способ доставки, например электронную почту, обычную почту со срочной доставкой в течение одного или двух дней.
4. Открывается список адресов, указанных в профиле пользователя.
5. Клиент выбирает адрес из списка.
6. Выполняется подтверждение адреса.
7. Клиент отправляет заказ.
8. ВИС завершается, когда клиент получает номер подтверждения.

Альтернативный вариант развития событий:

1. Альтернативный ход событий начинается с пункта 4.
2. 4а. Адрес отсутствует в профиле,
4б. Перейти к «Обновление профиля клиентом» или «Создание профиля клиентом».
3. ВИС продолжается с шага 5.

Постусловия: Заказ заполнен, отправлен и готов к исполнению. Открывается последняя страница «Просмотр продуктов».

ВИС, которые используют или расширяют данный: Расширяет UC05.1, «Заказ спецификаций товара».

Требования пользователя к реализации: Отсутствуют

Частота возникновения: Наблюдается в 17 из 100 сеансов работы клиентов, использующих Web.

Нерешенные задачи: Отсутствуют

Ответственный: Майк Данселио

История изменений:

Дата: 6 декабря 2002 г.

Автор: Хейди Стин

Описание: Начальная версия

Проверка корректности ВИС и СИС

Чрезвычайно важно, чтобы требования-кандидаты, ВИС и СИС проверялись совместно с пользователями и другими заинтересованными лицами. Это помогает выявить шаги процессов, которые, возможно, не задокументированы. На основании требований затем разрабатывается список функций. Коррективы списка функций выявляются в процессе уточнения ВИС, а дополнения вносятся только после согласования их заказчиком.

Как вы помните, проверка является итеративным процессом. Вместе с вариантами и сценариями использования системы она помогает обнаружить проблемы в требованиях.

Выбор прикладной архитектуры

Ключевой результат концептуального дизайна — концептуальная модель решения. Для ее создания необходимо понимать, какой набор сервисов должно предоставлять решение.

Сервисы в решении

Сервис (service) — это модуль прикладной логики, который включает методы выполнения операции, функции или преобразования.

Сервисам в соответствие ставятся действия, а сами они используются для реализации бизнес-правил, манипулирования данными и поддержки таких действий, как добавление, извлечение, просмотр и изменение данных. Сервисы доступны через сеть путем вызова опубликованного интерфейса, в котором описаны спецификации сервиса. Клиентов интересует не то, как реализован сервис, а какие функции он предоставляет.

Сервисы бывают простыми и сложными. Например, сервисы для создания, чтения, обновления и удаления информации — простые. Существуют также сервисы для выполнения сложных математических расчетов.

Вот сервисы, которые обычно реализуются в решении.

- **Пользовательские сервисы** представляют собой модули прикладной логики, обеспечивающие работу пользовательского интерфейса. **Пользовательские сервисы** приложения отвечают за взаимодействие приложения и пользователя. Чтобы разработать эффективные пользовательские сервисы, необходимо хорошо разобраться, кто будет работать с приложением, какие задачи они будут выполнять и каковы типичные схемы взаимодействия этих пользователей с приложением в процессе выполнения задач.
- **Бизнес-сервисы** — это модули прикладной логики, которые обеспечивают выполнение бизнес-правил в нужной последовательности. Бизнес-сервисы скрывают логику реализации бизнес-правил и преобразуют данные, полученные от пользовательских сервисов, других бизнес-сервисов и сервисов данных.
- **Сервисы данных** — это модули прикладной логики, обеспечивающие самый низкий уровень детализации при работе с данными. Сервисы данных применяются для реализации **бизнес-схемы** на источнике данных, на котором базируется приложение, и используются для управления данными всех типов — статическими, структурированными и динамическими. Эти сервисы задействуются всякий раз, когда пользователь или бизнес-сервис работает с данными или нуждается в доступе к ним.
- **Системные сервисы** — это модули логики приложения, обеспечивающие функции, лежащие вне бизнес-логики, например:
 - сервисы резервного копирования;
 - сервисы обработки ошибок;
 - сервисы безопасности;
 - сервисы обмена сообщениями.

Примеры сервисов

В таблице 4-6 перечислены сервисы приложения обработки заказов.

Таблица 4-6. Примеры сервисов

Сервисы	Примеры
Пользовательские сервисы	<ul style="list-style-type: none"> • Сервис отображения заказа • Сервис отображения данных учетной записи пользователя • Сервис отображения информации о товаре
Бизнес-сервисы	<ul style="list-style-type: none"> • Сервис размещения заказа • Сервис обновления данных учетной записи пользователя • Сервис извлечения информации о товаре • Сервис проверки наличия товара на складе
Сервисы данных	<ul style="list-style-type: none"> • Сервис информации о заказе • Сервис учетной записи пользователя • Сервис информации о товаре

Помните, что классифицировать сервисы следует по их функциям, а не местоположению. Классификация сервиса (пользовательский, бизнес-сервис или сервисы данных) зависит реально выполняемых функций, а не от его места в системе. Например, сервис данных, расположенный на клиентской рабочей станции, все равно остается сервисом данных.

Архитектура приложения

Сервисы в решении организуются в соответствии с архитектурой приложения, которая в свою очередь состоит из определений, правил и связей, формирующих структуру приложения. Архитектура показывает структуру приложения, но не содержит подробностей реализации, то есть она ориентирована на решение, а не на технологии, требуемые для его реализации. Чтобы построить концептуальную модель решения, необходимо выбрать модель приложения и вариант архитектуры приложения, который будет использоваться в решении.

Проектная команда выбирает архитектуру, ориентируясь как на сервисы, которые должно предоставлять решение, так и на ожидания пользователей, касающиеся рабочих характеристик приложения. Кроме того, учитываются связанные с решением допущения и ограничения. К допущениям (assumptions) также относятся используемые в решении операционные системы. Примерами ограничений служат бюджет, время, навыки и доступные проекту ресурсы.

Вот примеры некоторых прикладных архитектур:

- клиент-серверная;
- многоуровневая;
- основанная на объектах, не сохраняющих состояния;
- основанная на кэшировании.
- многоуровневая, состоящая из клиента с поддержкой кэширования и сервера, не поддерживающего состояния, но с кэшем.

Клиент-серверная архитектура

Клиент-серверная архитектура — это двухуровневый вариант архитектуры, основанный на механизме «запрос — ответ». Клиент инициирует сеанс работы с сервером и управляет ходом сеанса, по необходимости обращаясь к серверу. Клиент запрашивает у сервера один или более из предоставляемых им сервисов. Получив запрос, сервер выполняет требуемую операцию и возвращает результаты клиенту.

Преимущество такой архитектуры — возможность разделить вычислительные операции, необходимые для выполнения задачи, между двумя устройствами. Некоторые операции клиент может выполнить самостоятельно, не перегружая сервер запросами.

А основное ограничение клиент-серверной архитектуры — сильная зависимость клиента от сервера. Если клиенту не удастся подключиться к серверу, он не в состоянии выполнять даже базовые задачи. Это типичная ситуация в решениях, где выполняется обработка большого количества запросов. Такие приложения не очень хорошо масштабируются.

Многоуровневая архитектура

Многоуровневая архитектура — результат логического развития клиент-серверной архитектуры. Она состоит из иерархических уровней. Различные сервисы приложения четко разнесены по уровням, причем «общаются» только сервисы соседних уровней: переход через уровень невозможен. Уровни инкапсулируют сервисы и защищают их друг от друга, предоставляя упрощенный набор интерфейсов для общих ресурсов. Пользовательские сервисы, бизнес-сервисы и сервисы данных являются примерами многоуровневой архитектуры.

К преимуществам многоуровневой архитектуры следует отнести лучшую масштабируемость системы и высокий уровень безопасности. Сервисы можно рас-

Занятие 5. Оптимизация концептуального дизайна

Заключительная стадия концептуального дизайна — оптимизация. Здесь начинает формироваться образ конечного приложения в виде концепции решения.

В таблице 4-7 перечислены задачи и результаты стадии оптимизации.

Таблица 4-7. Задачи и результаты оптимизации концептуального дизайна

Задача	Результат
Совершенствование процесса	Описание задач, решаемых создаваемой системой
Проверка концептуального дизайна	Проверено соответствие будущего состояния системы и решаемых с ее помощью задач

На этом занятии вы узнаете, как оптимизировать описания процессов, полученные на этапе концептуального дизайна, а также как оценить оптимизированные процессы и проверить концептуальный дизайн.

Изучив материал этого занятия, вы сможете:

- ✓ оптимизировать процессы;
- ✓ оценить оптимизированные процессы;
- ✓ проверить оптимизированные процессы;
- ✓ проверить концептуальный дизайн.

Продолжительность занятия - около 15 минут.

Оптимизация процессов

В дополнение к исследованию и анализу существующих бизнес-процессов вы должны определить, какие процессы будут включены в решение и как пересекаются потребности этих процессов. Чтобы получить решение, ориентированное как на процессы, так и на **пользователей**, высшее руководство (то есть фактически владельцы процессов), пользователи и проектная команда должны работать сообща.

В процессе проектирования будущего состояния процесса изучаются сценарии, **описывающие** текущее состояние, устраняются неэффективные, узкие места и дублирующие операции. Лучше всего, если будущее решение проектная команда проектирует совместно с **пользователями**, с участием экспертов по реинжинирингу бизнес-процессов.

Примечание Проектировать будущее решение и проводить реорганизацию бизнес-процессов следует только в том случае, если эти действия укладываются в рамки проекта.

Хотя большинство команд разработчиков приложений в состоянии создать качественный проект будущей системы, все равно рекомендуется привлекать экспертов по реинжинирингу. Чтобы описать будущее состояние, команда выполняет следующие задачи:

- создает общую картину будущего состояния:
 - повышение производительности в сравнении с текущими процессами;
 - анализ компромисса между потребностями и пожеланиями;
 - соблюдение баланса между требованиями бизнеса и пользователей;
 - соблюдение баланса между широтой функционального наполнения решения и возможностью их технической реализации;
- пересматривает существующие процессы для обеспечения оптимальной поддержки ключевых бизнес-операций и процессов с привлечением при необходимости экспертов по реинжинирингу бизнес-процессов:
 - оптимизация процесса в целом;
 - интеграция процессов (по возможности);
 - устранение неэффективных, узких мест и дублирования;
- строит СИС предполагаемого решения, в которых будут отражены пересмотренные процессы;
- проверяет сценарии будущего решения совместно с другими заинтересованными лицами;
- при необходимости повторно выполняет процесс уточнения и совершенствования проекта.

Вместо того чтобы проектировать будущее решение с точки зрения аналитика, посмотрите на проект решения с точки зрения пользователя. Наверняка вы обнаружите возможности усовершенствовать рабочие процессы, устаревшие или непродуктивные корпоративные бизнес-политики, неоправданные бюрократические процедуры, потерю связи между процессами и не приносящие прибыли роли, которые затрудняют и фрагментируют бизнес-процесс, снижая его общую эффективность. Чтобы выяснить требования с точки зрения пользователя, придерживайтесь следующих рекомендаций:

- определите ненужные последовательные операции, узкие места и ненужные шаги в процессе;
- устраните дублирование в существующем рабочем окружении;
- интегрируйте независимые функционирующие информационные системы в единые системы, охватывающие все предприятие;
- найдите и избавьтесь от ненужной бумажной волокиты;
- определите минимальный обязательный уровень производительности процессов, попытайтесь перепроектировать процесс так, чтобы получить те же результаты при меньших усилиях;
- сократите время ожидания в системе, выделяя шаги, которые можно выполнять одновременно, а не последовательно;
- усовершенствуйте систему, предоставив пользователям дополнительную информацию, например о производительности системы, чтобы гарантировать немедленное обнаружение и устранение возникающих неполадок.

Пересмотр процессов, определенных в процессе проектирования

Ключевая цель проектной команды вместе с экспертами по реинжинирингу бизнес-процессов — создать продукт, который максимально облегчает пользователям выполнение их задач. Нет универсального метода эффективной перестройки процессов. Как правило, применяются самые разнообразные методики: мозговые штурмы с использованием различных творческих методов: создание метафор, варьирование слов и идей, исследование барьеров творческой фантазии,

генерация идей, открытые свободные дискуссии и дискуссии «без ограничений». Полученные идеи — это не готовое решение, но они позволят найти новые подходы к проблемам.

Эксперты, проводящие реинжиниринг бизнес-процессов, должны задаться следующими вопросами.

- Возможно ли расширить полномочия пользователей?
- Можно ли делегировать принятие определенных решений? Если да, то как это отразится на бизнес-операции?
- Что из того, что трудно добиться в физическом мире или на бумаге, можно реализовать программными методами?
- Каковы возможности автоматизации или применения новых технологий?

Во время пересмотра проекта рассматривайте различные альтернативы. Очень важен творческий подход. Нередко задача имеет несколько решений.

Ключевой принцип пересмотра проекта — каждое правило или допущение должно подвергаться критическому разбору. Также в процессе следует проанализировать все элементы процесса — входные и выходные данные, уровни производительности, ресурсы, процедуры управления и контроля, продуктивность и временные характеристики. Вот свод ключевых элементов, которые следует учитывать при оптимизации процессов.

- Критически относитесь к правилам, не относитесь к ним, как к данности. Например, можно поставить под сомнение правило: «В компании все медицинские запросы пользователей должны утверждаться на уровне подразделения, департамента и отдела».
- Постройте процессы в соответствии с требованиями производительности. Рассматривайте производительность с точки зрения удовлетворения потребностей пользователей, а не как один из способов извлечения краткосрочной выгоды.
- Выстраивайте действия вокруг товаров и услуг. Обязанности пользователя должны служить достижению целей процесса, а не отдельной задачи.
- Заменяйте бюрократические структуры и иерархии самоорганизующимися командами, работающими параллельно.
- Повышайте продуктивность, перемещая центр внимания с разделения работы и специализации задач на их сжатие и интеграцию.
- Определяйте, где можно применить технологии для обеспечения и поддержки реорганизованного процесса.
- Разбивайте процессы на подпроцессы и работайте в каждый момент времени только с одним из них.

Как только процесс оптимизирован и утвержден в качестве конечной цели, основное внимание смещается к реорганизации работы. Работа реорганизуется исходя из нужд бизнеса, а не того, что на первый взгляд кажется разумным и выполнимым. Команда должна изучить концептуальную модель решения и перестроить ее, сделав процессы более эффективными.

В результате исследований зачастую обнаруживаются новые ВИС, которые придется включить в концептуальную модель. На основании этих новых ВИС часто создаются альтернативные варианты организации процессов. Обсудите их с пользователями и экспертами в области бизнес-процессов, применяя особые методы, например ролевые игры. На основании результатов тестирования и проверки выбирают одну из альтернатив в качестве очередного детального проекта.

Вкратце процесс перестройки работ выглядит следующим образом,

- Устраивается мозговой штурм для генерации максимального количества идей:
 - используется визуализация;
 - участникам предлагается мыслить творчески и смело нарушать устоявшиеся правила;
 - участники генерируют, оценивают и анализируют варианты.
- Мозговые штурмы совмещаются с другими методами.
- Совместно с конечными пользователями и экспертами в области процессов проверяются и анализируются альтернативные варианты.

Оценка реорганизованных процессов

Она необходима, чтобы определить, нет ли неправильно понятой или упущенной из виду информации, касающейся требований пользователей. Эта оценка также позволяет убедиться в том, что у всех членов команды совпадает представление о решении.

Также оценивается стоимость решения и выгоды, которые оно дает в его текущем состоянии. Именно на этой стадии многие проекты отбрасываются, так как оказывается, что окупаемость инвестиций (return of investment, ROI) и них не удовлетворяет требованиям бизнеса.

Кроме того, необходимо определить первичные и вторичные выгоды. Следует обозначить конкретные преимущества и лиц, которые их получают (включая пользователей).

В таблице 4-8 сведены воедино критерии оценки большинства бизнес-приложений.

Таблица 4-8. Оценка преимуществ и недостатков реорганизации

Преимущества	Недостатки
Эффективность и результативность решения — как в целом, так и его отдельных частей	Ресурсы, затраченные усилия и время
Выгоды от внедрения решения	Затраты на приобретение технических средств и новейших технологий
Организационные и культурные последствия	Ежегодные и периодические издержки, обусловленные обслуживанием решения
Возможная прибыль и экономия	

Проверка модели концептуального дизайна

После создания концептуальный дизайн проверяется вместе с пользователями будущего решения и другими заинтересованными лицами. Можете совместить проверку с процессом оценки данных, полученных от пользователей. Проверка позволяет получить от пользователей подтверждение того, что концептуальная модель и требования представляют их видение бизнес-решения и что это решение охватывает все варианты и сценарии использования системы.

Концептуальный дизайн решения проверяется на основании всех ВИС и СИС, требований бизнеса, архитектуры, рисков, доступных ресурсов и времени и всех остальных созданных артефактов. Пользователям проще проверять сценарий, поскольку он содержит контекст требования.

Проверка сценариев также должна предусматривать проверку результатов определения приоритетов. Иначе говоря, приоритеты и все действия сценария необходимо проверить, так как это:

- снижает риск;
- позволяет обнаружить пропущенную информацию;
- помогает обнаружить различное видение и интерпретацию решения, особенно между руководством и пользователями;
- подтверждает объемы работ;
- помогает определить приоритеты;
- обеспечивает базу для создания логического дизайна;

Процесс проверки, тестирования и реорганизации обычно состоит из следующих шагов:

1. Реорганизации.
 2. Построения набора сценариев.
 3. Построения прототипа системы.
 4. Создания макета пользовательского интерфейса.
 5. Согласования с руководством и пользователями.
 6. Повторения последовательности операций до тех пор, пока нужды пользователей и бизнеса не будут полностью удовлетворены.
- Шаги 5 и 6 повторяются столько раз, сколько потребуется.

Методы проверки концептуального дизайна

Существуют несколько методов проверки сценариев.

- **Прогон сценария.** Человек, ответственный за сбор требований, проводит пользователей через весь сценарий и по ходу действия задает вопросы, стараясь определить, согласны ли пользователи с теми или иными действиями и событиями.
- **Ролевые игры.** Специально отобранные пользователи выполняют несколько сценариев, чтобы проверить процесс и выявить области, требующие доработки. Это помогает команде выбрать один из сценариев для разработки детального дизайна.
- **Создание прототипа.** Прототип предоставляет детальную информацию о процессе, его течении, его влиянии на организацию и его технологических возможностях. Создание прототипа — эффективный способ демонстрации собранных и синтезированных требований пользователям. Вы можете создать прототип в виде электронного приложения или в бумажной форме — как свод функциональных спецификаций. Команде следует пересмотреть и проверить прототипы. Последние помогают менеджерам принять решение о том, когда дизайн можно считать готовым. Команда также должна создать прототипы нескольких пользовательских интерфейсов. Они необходимы, чтобы проверить, вся ли необходима информация собрана, а не для того, чтобы увидеть, красиво ли выглядит то или иное диалоговое окно.

Практикум. Анализ требований



При выполнении упражнений используйте знания, полученные на занятиях.

Упражнение 1. Уточнение вариантов использования системы и требований

Откройте диаграмму Microsoft Visio *C04Ex1.vsd* из папки *\Solution Documents\Chapter04* на прилагаемом компакт-диске и перейдите на вторую вкладку; AWC Use Case — Ex1ToSolve.

1. Какие вопросы надо задать участникам проекта со стороны заказчика, чтобы определить варианты использования системы, содержащиеся в *Manage Orders* (управление заказами).

Ниже приведены примеры возможных вопросов.

- Что значит «управлять заказами»?
 - Каковы обязанности роли «Менеджер по продажам» в отношении управления заказами? Назовите как можно больше конкретных задач.
 - Каковы обязанности роли «Клиент» в процессе управления заказами? Назовите как можно больше конкретных задач.
 - Как оперируют с заказами клиенты и менеджеры по продажам?
2. Клиент предоставил вам следующую информацию:

- как «Клиент», так и «Менеджер по продажам» должны иметь возможность создавать, печатать, изменять и отменять заказы с использованием Web-сайта (менеджеры по продажам также используют специализированные устройства);
- занятые в проекте сотрудники компании *Adventure Works Cycles* согласны с тем, что необходима функция «Справка», чтобы пользователь мог получать ответы на наиболее типичные вопросы, получать справку в виде мастера, а также иметь доступ к сервису мгновенных сообщений;
- менеджер по продажам (но ни в коем случае не клиент) вправе предоставлять скидки на заказ.

Используйте эту информацию и ваши требования для построения соответствующих ВИС на третьей вкладке, *Ch4 Manage Orders* в файле Visio. В качестве отправной точки вам предоставлены варианты использования системы *Create Order* (Создать заказ) и *Revise Order* (Откорректировать заказ).

Одно из возможных решений вы найдете в документе *C04Ex1_Answer* document в папке *\Solution Documents\Chapter04* на прилагаемом компакт-диске.

Упражнение 2. Просмотр диаграммы концептуальной модели

Откройте Visio-диаграмму *C04Ex2.vsd* (папка *\SolutionDocuments\Chapter04*). Ознакомьтесь с концептуальной моделью. Откройте документ требований *C04Ex2.xls* в той же папке. Ознакомьтесь с требованиями и откорректируйте их формулировки, а также организуйте их в виде свободной иерархии, используя концептуальную модель в качестве основы. Например, объедините все требования, связанные с ВИС *Revise Order* (Откорректировать заказ). Охватывает ли данная концептуальная модель все текущие требования?

Одно из возможных решений вы найдете в документе *C04Ex2_Answer.xls* в папке *\Solution Documents\Chapter04*.


Резюме

- Этап планирования завершается построением архитектуры и дизайна решения, составлением планов на его разработку и развертывание и календарных графиков, соответствующих задачам и ресурсам.
- Этап планирования предусматривает разработку трех видов дизайна: концептуального, логического и физического.
- За этап планирования отвечает команда менеджеров программы.
- Основные результаты этапа планирования таковы:
 - функциональные спецификации, которые описывают будущее решение исходя из всех полученных данных;
 - генеральный план проекта — набор планов каждой из шести ролей команды. Основная цель — обеспечение соответствия функционального наполнения решения и функциональных спецификаций;
 - генеральный календарный план проекта, который определяет срок выполнения работы;
 - обновленный генеральный документ по оценке риска, где описываются виды связанного с созданием продукта риска.
- Функциональные спецификации — это виртуальное хранилище проекта и артефактов, созданных на этапе планирования модели процессов MSF.
- Цели функциональных спецификаций:
 - формирование единого понимания требований бизнеса и пользователей;
 - разбиение задачи на более мелкие части и представление решения в виде логических модулей;
 - создание среды для планирования, составления календарных планов и построения решения;
 - выполнение роли посредника и механизма согласования между проектной командой и заказчиком, определяющим, что, собственно, следует создать.
- Функциональные спецификации состоят из:
 - « сжатого описания концептуального дизайна;
 - сжатого описания логического дизайна;
 - сжатого описания физического дизайна;
 - используемых командой стандартов и процессов.
- Концептуальный дизайн — это процесс сбора, анализа и определения приоритетов особенностей бизнеса и точек зрения пользователей на проблему и будущее решение с последующим созданием высокоуровневого представления решения.
- Цели создания концептуального дизайна:
 - определение бизнес-проблемы, которую предстоит решить;
 - определение требований бизнеса, заказчика и конечного пользователя;
 - описание состояния предприятия, которое предполагается достичь в будущем.
- Процесс концептуального дизайна состоит из трех стадий: исследования, анализа и оптимизации.
- На стадии исследования команда собирает дополнительную информацию о требованиях бизнеса, ВИС и СИС.

- На стадии анализа выполняются следующие задачи:
 - синтез информации;
 - « уточнение диаграмм ВИС;
 - выбор подходящей прикладной архитектуры решения;
 - « создание концептуальной модели решения.
- Требования бизнеса должны быть:
 - четко определенными;
 - краткими;
 - проверяемыми;
 - сгруппированными в виде иерархии;
 - созданными на языке бизнеса без использования профессионального жаргона.
- Требования бизнеса делятся на следующие категории:
 - пользовательские;
 - системные;
 - « процедурные;
 - бизнес-требования.
- В процессе уточнения диаграмм ВИС проектная команда:
 - создает подчиненные ВИС;
 - создает СИС для каждого подчиненного варианта использования системы;
 - проверяет и корректирует все ВИС и СИС на основании уточненных требований.
- Чтобы создать концептуальный дизайн решения, определяют сервисы и архитектуру решения.
- Существует несколько видов сервисов, предоставляемых решением: пользовательские сервисы, сервисы данных и системные сервисы.
- Архитектура приложений состоит из определений, правил и связей, которые формируют структуру приложения.
- Приложение обычно основывается на одной из следующих архитектур:
 - клиент-серверной архитектуре;
 - многоуровневой архитектуре;
 - архитектуре, основанной на объектах без сохранения состояния;
 - архитектуре, основанной на кэшировании;
 - многоуровневой архитектуре с Web-браузером в качестве клиента.
- На шаге оптимизации команда оптимизирует решение и бизнес-процессы и проверяет их на предмет соответствия требованиям бизнеса и ВИС.
- Для описания будущего состояния системы проектная команда:
 - создает общую картину целевого решения;
 - реорганизует существующие процессы для оптимальной поддержки ключевых бизнес-операций и процессов;
 - строит сценарии будущей системы, отражающие реорганизованные процессы;
 - согласует проект системы со всеми участниками бизнес-процессов;
 - при необходимости процесс реорганизации повторяется.

- В процессе реорганизации проектная команда:
 - проводит мозговые штурмы для генерации максимального количества идей;
 - совмещает мозговые штурмы с другими методами;
 - проверяет и согласует альтернативы с конечными пользователями и экспертами в области реинжиниринга.
- Проектная команда проверяет **концептуальный** дизайн решения на предмет соответствия вариантам и **сценариям** использования системы и требованиям бизнеса.

Закрепление материала

 j Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Какова цель этапа планирования?
 2. Чем отличаются обязанности ролей «менеджер решения» и «менеджер программы» на этапе планирования?
 3. Каковы основные результаты этапа планирования?
 4. Каким образом функциональные спецификации становятся «черновым» планом для команды разработки?
 5. Каковы цели создания функциональных спецификаций?
 6. Какие виды риска возникают при отказе от создания функциональных спецификаций?
 7. Каковы различия между требованиями бизнеса и пользователей решения?
 8. Каковы преимущества концептуального дизайна?
 9. Чтобы увеличить продажи на 15% в следующем финансовом году, компания планирует создать Интернет-магазин, для чего предполагается разработать высокоскоростной сайт, обеспечивающий безопасную обработку кредитных карт пользователей и доступный 24 часа в сутки, 7 дней в неделю. Возможность покупать продукты через сайт будет предоставляться только зарегистрированным пользователям. Каковы пользовательские, системные, процедурные и бизнес-требования к этому сайту?
- К). Каковы цели стадии анализа в процессе концептуального дизайна?
11. Какие преимущества дает синтез информации?
 12. Какие задачи необходимо выполнить в процессе создания проекта будущего решения?

13. Какие преимущества дает проверка концептуального дизайна?
14. Перечислите четыре категории сервисов.
15. Изучите и распределите по категориям следующие сервисы:
 - отображения подробной информации о сотруднике;
 - обновления информации о сотруднике;
 - предоставления данных о сотруднике;
 - электронную почту.
16. Каковы особенности уточненных требований бизнеса?
17. Каким образом уточняются варианты использования системы на стадии анализа в процессе концептуального дизайна?
18. Каковы критерии оценки стоимости решения?

ГЛАВА 5

Создание логического дизайна

Занятие 1. Основные сведения о логическом дизайне	147
Занятие 2. Разработка логического дизайна	153
Занятие 3. Документирование результатов логического дизайна	166
Занятие 4. Оптимизация логического дизайна	173
Практикум. Определение объектов в логическом дизайне	178
Резюме	179
Закрепление материала	180

В этой главе

Этап планирования состоит из трех стадий, в течение которых создается дизайн трех типов — концептуальный, логический и физический. Стадии реализуются последовательно, однако их начальные и конечные точки перекрываются. Это означает, что процесс создания концептуального дизайна всегда начинается до логического дизайна, но все еще продолжается, когда стартуют работы над логическим дизайном. Точно так же логический дизайн всегда начинается до физического и продолжается с началом последнего. В таблице 5-1 описано создание различных видов дизайна,

Таблица 5-1. Три типа дизайна

Тип дизайна	Перспектива	Назначение
Концептуальный	Задача рассматривается с точки зрения пользователя и бизнеса	Задача и решение выражается в виде СИС
Логический	Решение рассматривается с точки зрения проектной команды	Решение определяется как логический набор взаимодействующих объектов и сервисов
Физический	Решение рассматривается с точки зрения разработчиков	Определяются сервисы решения и используемые технологии

В этой главе рассказывается о процессе создания логического дизайна на этапе планирования. Вы получите общее представление о назначении логического дизайна и его преимуществах, а также о составе команды, его создающей. Здесь описывается, как создать логический дизайн бизнес-решения, как применять инструменты и методы документирования результатов, а также оптимизировать логический дизайн.

Прежде **всего**

Для изучения материалов этой главы необходимо:

- понимать модель процессов Microsoft Solutions Framework (MSF);
- знать, каковы результаты этапа создания общей картины;
- иметь представление об этапе планирования;
- знать о трех стадиях концептуального дизайна — исследовании, анализе и оптимизации.

Занятие 1. Основные сведения о логическом дизайне

В процессе концептуального дизайна решение описывается с точки зрения бизнеса и пользователей. Далее следует продумать решение с точки зрения проектной команды. Именно это и выполняется на стадии логического дизайна.

На этом занятии вы узнаете о целях и назначении логического дизайна.

Изучив материал этого занятия, вы сможете:

- ✓ описать логический дизайн;
- ✓ описать преимущества логического дизайна;
- ✓ описать роли и обязанности членов команды в процессе создания логического дизайна.

Продолжительность занятия — около 10 минут.

Что такое логический дизайн

Логический дизайн определяется как процесс описания решения в терминах организации, структуры и взаимодействия различных его частей с точки зрения проектной команды. Логический дизайн:

- определяет составляющие части решения;
- описывает рабочую среду, в которой все части решения собираются воедино;
- иллюстрирует, из чего собирается решение и как оно взаимодействует с пользователями и другими решениями.

Создавая логический дизайн, команда принимает во внимание все требования бизнеса, пользователей, системные и процедурные, по безопасности, аудиту, поддержке журналов, масштабируемости, поддержке состояний, обработке ошибок, лицензированию, поддержке других языков, архитектуре приложения и интеграции с другими системами.

Совет по планированию Поскольку процесс дизайна в MSF эволюционный и итеративный, логический дизайн оказывает влияние на физический дизайн.

Когда выполняется логический дизайн

Процесс создания логического дизайна обычно начинается, когда проектная команда приступает к определению важных объектов и атрибутов объектов-сущностей, то есть еще до завершения концептуального дизайна. Решение о начале работы над логическим дизайном принимается в каждом случае индивидуально, в зависимости от проекта и команды. На рис. 5-1 показано место логического дизайна в модели процесса MSF.



Рис. 5-1. Где мы находимся в процессе проектирования в MSF

Когда команда проекта переходит от концептуального к логическому дизайну, угол зрения меняется. В процессе концептуального дизайна команда проекта определяет бизнес-задачу на основании данных, полученных в процессе исследования предприятия и групп пользователей. Во время создания логического дизайна команда проекта выбирает элементы решения с собственной точки зрения, то есть определяет, какие функции должно выполнять решение. На основании этой информации конструируют поведение и организацию решения,

Примечание Логический дизайн помогает команде уточнить требования, которые были сформулированы на этапе создания общей картины решения и уточнены в процессе концептуального дизайна.

Качество логического дизайна во многом зависит от того, насколько хорош концептуальный дизайн. Если проектная команда создала качественный логический дизайн, то, ознакомившись с ним, новый член команды легко определит

важные части решения и поймет, как совместными усилиями решить бизнес-задачу. Чаще всего этап логического дизайна перекрывается с физическим дизайном.

Примечание При создании логического дизайна проектная команда оптимизирует структуру решения, а при создании физического дизайна совершенствует его рабочие характеристики, в результате чего процессы логического и физического дизайна повторяются итеративно.

Задачи логического дизайна

На более низком уровне процесс логического дизайна можно разделить на следующие задачи (рис. 5-2):

- анализ логического дизайна, во время которого команда:
 - уточняет список инструментов и технологий, которые предполагается использовать;
 - определяет бизнес-объекты и сервисы;
 - определяет важные атрибуты и ключевые отношения;
- оптимизацию логического дизайна, в том числе:
 - совершенствование логического дизайна;
 - проверку логического дизайна.

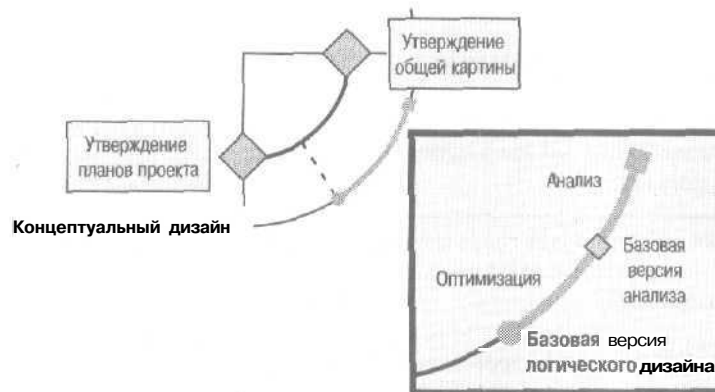


Рис. 5-2. Задачи в процессе логического дизайна

Во время создания логического дизайна команда иногда обнаруживает новые требования или функции, пропущенные ранее. В этом случае необходимо вновь обратиться к заказчику, чтобы выяснить, корректны ли эти требования или функции, и узнать его мнение по поводу этих «открытий». В зависимости от ответа заказчика область действия проекта может измениться. Если это случится, исправьте уже созданную документацию и модели и включите в них эти требования или функции.

Основное назначение логического дизайна

Главная задача логического дизайна — определить, что должна делать система, и описать это, используя определенный набор элементов. Логический дизайн позволяет:

- определить бизнес-задачи, решаемые с помощью существующих технологий. Логический дизайн — это не технологический проект, но он помогает команде определить бизнес-задачи, которые должна поддерживать система;
- определить возможности и ограничения технологии. Хотя логический дизайн не зависит от физической реализации, проектная команда может начать изучать ограничения и возможности, влияющие на выбор технологии, которая ляжет в основу решения;
- определить подходящие технологии для реализации. Команда должна полностью разобраться в тонкостях проекта, прежде чем выбрать технологию для его реализации. Логический дизайн не оптимизирован для физической модели, но он помогает команде проекта определить технологии, оптимальные для реализации решения. Технологии-кандидаты иногда **выявляются** во время создания логического дизайна, если, конечно, он прекрасно укладывается в рамки этих технологий;
- определить области логического дизайна, которые следует скорректировать в соответствии с особенностями инфраструктуры, эксплуатации и развертывания. Технологии разработки компонентов не влияют на логический дизайн. Тем не менее последний влияет на физический дизайн, а тот в свою очередь зависит от технологий. Следовательно, если заказчику требуется решение на основе Web, проектная команда в **процессе** логического дизайна должна учитывать ограничения, обусловленные особенностями развертывания.

Результаты логического дизайна

Результаты логического дизайна таковы (каждый из них подробно описан в занятии 3);

- логическая модель объектов;
- высокоуровневый дизайн пользовательского интерфейса;
- логическая модель данных.

Примечание Логическая модель объектов и логическая модель данных документируют практически одну и ту же информацию, но по-разному. В большинстве проектов для документирования логического дизайна вполне достаточно одной модели. Однако некоторые команды предпочитают создавать обе модели в преддверии физического дизайна, а затем проводить взаимное сравнение и проверку этих моделей.

Результаты логического дизайна становятся основой физического дизайна, Чтобы пояснить, что происходит в процессе логического дизайна, рассмотрим пример проектирования дома. На этапе концептуального дизайна заказчик и архитектор перечисляют требования к жилищу: возможности приема гостей, отдыха, трапезы, хранения вещей и необходимые коммунальные услуги. Во время логического дизайна архитектор создает план этажей, определяет структурные элементы — двери, окна, крышу, двор и комнаты. Архитектор также создает общий план дома. Точно так же в **процессе** работы с **пользователями** и заказчиком на этапе концептуального дизайна проектная команда определяет требования. Далее на их основе они утверждают компоненты решения и **полностью** конструируют общую инфраструктуру, в которую будут встраиваться эти компоненты.

Преимущества логического дизайна

Процесс создания логического дизайна решения весьма полезен. В частности, он помогает объединить «технарей» с другими членами команды. Вот основные преимущества логического дизайна.

- Логический дизайн помогает эффективно справляться со сложностью проекта: сначала определяется общая структура решения, а затем описываются его части и их взаимодействие друг с другом для решения поставленной задачи. Многие проекты терпят фиаско из-за их сложности и того, что реализация была начата до создания качественного дизайна. Сложность ведет к путанице, а значит, дизайн продукта наверняка окажется неудачным. Логический дизайн помогает команде осознать сложность бизнес-процессов и успешно справиться с ней,
- Логический дизайн отражает и поддерживает требования концептуального дизайна, позволяя лишний раз проверить, решает ли он бизнес-задачу. Логический дизайн помогает команде обнаружить ошибки и нестыковки в концептуальном дизайне, избавиться от дублирующихся требований и выявить возможности повторного использования в сценариях. На основании этих результатов пересматривают концептуальный дизайн и корректируют требования, чтобы гарантировать, что проект действительно решает поставленную задачу.
- Логический дизайн позволяет найти точки соприкосновения множества разнообразных подразделений и их систем и дает ясное представление о проекте в целом. Команда анализирует логический дизайн, пытаясь определить возможности повторного использования и сделать дизайн более эффективным, а приложение — более удобным для поддержки и сопровождения.
- Логический дизайн служит отправной точкой для физического дизайна. Поскольку логический дизайн — это, по сути, модель объектов, проектная команда может определить расположение и атрибуты объектов для их физического представления.

Роли команды в процессе логического дизайна

В процессе логического дизайна обязанности отдельных групп команда отличаются. Например, группа менеджеров программы отвечает за разработку функциональных спецификаций, она должна обеспечивать взаимодействие и общение с командой, следить, чтобы проект укладывался в график, и предоставлять отчеты о состоянии проекта. Эта группа несет основную ответственность за выполнение логического дизайна.

На рис. 5-3 показаны обязанности различных ролей в процессе логического дизайна.

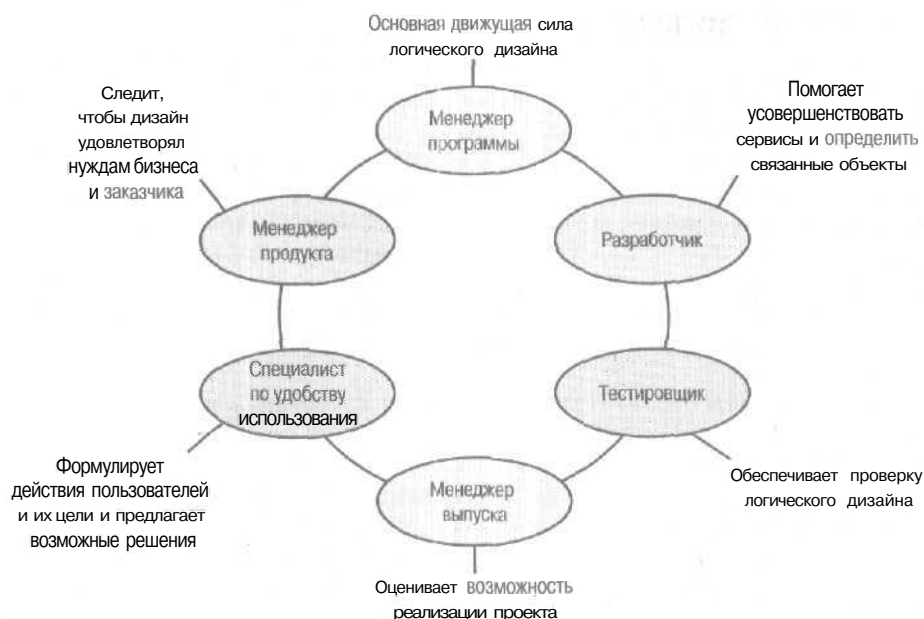


Рис. 5-3. Роли в команде и их обязанности

В таблице 5-2 собраны сведения о задачах, которые выполняет каждая роль в команде в процессе логического дизайна.

Таблица 5-2. Роли членов команды

Роль	Основные обязанности	Дополнительные обязанности
Менеджер решения	Следит за тем, чтобы дизайн удовлетворял нуждам бизнеса и заказчика	Управление ожиданиями пользователей
Менеджер программы	Несет основную ответственность за результаты логического дизайна	Определяет план проекта, включая ресурсы, график и оценку риска
Разработчик	Определяет сервисы и связанные объекты	Выступает технологическим консультантом в вопросах выбора и создания прототипов технологий
Тестировщик	Проверяет соответствие моделей логического дизайна концептуальному дизайну	Составляет высокоуровневый план тестирования
Специалист по удобству использования	Определяет действия пользователей и их цели и предлагает решения	Составляет высокоуровневый план обучения пользователей
Менеджер выпуска	Оценивает возможность реализации проекта	Определяет инфраструктуру и методы развертывания

Занятие 2. Разработка логического дизайна

На этапе анализа в процессе создания логического дизайна, команда разделяет проблему и ее решение на более мелкие части, или модули. *Модуль* — это логический элемент, который используется как абстракция при построении ВИС и СИС, конструируемых в процессе концептуального дизайна. Для каждого модуля команда определяет объекты, сервисы, атрибуты и отношения. Во время логического дизайна команда также определяет возможные технологии для проекта.

На этом занятии рассказывается, как определить технологии-кандидаты для применения в бизнес-решении на основании логического дизайна, бизнес-объектов, сервисов, атрибутов объектов и сервисов и отношений между объектами в вариантах использования системы.

Изучив материал этого занятия, вы сможете:

- ✓ выбрать технологии-кандидаты;
- ✓ определить предварительные бизнес-объекты на основании ВИС;
- ✓ определить сервисы на основании ВИС;
- ✓ определить атрибуты на основании ВИС;
- ✓ определить отношения между объектами на основании ВИС.

Продолжительность занятия — около 20 минут.

Уточнение возможных технологий на этапе логического дизайна

В процессе создания концептуального дизайна команда определяет доступные технологии, которые можно использовать в тех или иных частях решения. На этапе логического дизайна команда уточняет уже составленный список технологий-кандидатов, внося в список дополнительные технологии. При оценке технологий-кандидатов учитываются особенности бизнеса, архитектуры предприятия и технологий.

Характеристики, связанные с особенностями бизнеса

- **Целесообразность.** Следует определить, действительно ли выбранная технология удовлетворяет нуждам бизнеса и способствует выполнению требований.
- **Стоимость решения.** Необходимо оценить полную стоимость продукта, включая лицензии разработчика, сервера и поставщика, а также стоимость модернизации. Команда также должна принимать во внимание затраты на приобретение базового аппаратного и программного обеспечения, поддержку, инфраструктуру и обучение. Новый продукт может быть как краткосрочной, так и долгосрочной инвестицией. В первом случае временно решается самая насущная проблема, например удовлетворить потребность в небольшом и простом клиентском пользовательском интерфейсе, пока не готово основное Web-приложение для интрасети. Долгосрочные инвестиции — это то, что приносит пользу на протяжении многих лет, например оборудование рабочего места более скоростным доступом к сети или покупка самого современного оборудования для серверов и рабочих станций.

- **Опыт.** Важно понимать, что существующий опыт пользователей по работе с теми или иными технологиям имеет огромное значение. Постарайтесь собрать ответы на такие вопросы, как, например, «Каков опыт обучения и консультаций?».
- **Окупаемость инвестиций.** Каждая инвестиция должна окупаться и в конечном итоге приносить прибыль. Никогда не выбирайте технологию только потому, что она новая или «модная». Инвестиции в новые технологии должны представлять ценность для бизнеса, например повышать доходы или снижать расходы.
- **Зрелость.** Зрелый продукт принят рынком, хорошо изучен и стабилен. а найти специалистов по его поддержке не составляет труда.
- **Удобство поддержки и сопровождения.** Выбирая технологию, очень важно осознавать, что ее придется сопровождать вместе с созданным решением. Команда должна учесть требуемый заказчиком уровень поддержки проекта и предприятия.
Есть и другие факторы, которые нужно учитывать при выборе технологий-кандидатов:
 - простота развертывания;
 - конкурентные преимущества;
 - время, необходимое для вывода на рынок технологии-кандидата;
 - восприятие продукта на рынке;
 - возможность интеграции с существующими системами.

Характеристики, обусловленные особенностями архитектуры предприятия

Приложение должно отвечать «букве и духу» существующей корпоративной архитектуры. Кроме того, проектной команде следует учитывать планы по ее развитию в будущем. Например, если планируется использовать Microsoft Windows Server 2003, этот факт необходимо принять во внимание при выборе технологий-кандидатов.

Архитектура предприятия описывает текущее состояние и планы на будущее. Решение должно укладываться в ограничения на архитектуру предприятия. Например, новая технология потокового видео потребует дополнительной пропускной способности сети, а новая система работы с изображениями — мониторов с большой диагональю. На эти изменения потребуются не только деньги, но и время на изучение и реализацию новых технологий. Если две технологии имеют одинаковый набор функций, лучшей окажется та, которая больше соответствует существующей архитектуре предприятия.

Технология должна хорошо взаимодействовать с другими системами в организации. Кроме того, необходимо определить коммуникационный интерфейс, чтобы другие приложения легко могли взаимодействовать с новой технологией.

Характеристики, обусловленные особенностями технологий

- **Безопасность.** Определите требования по безопасности нового решения, в том числе поддержку аутентификации, управления доступом, шифрования и аудита. Аутентификация и авторизация — это два *различных* этапа в едином процессе предоставления пользователю доступа к ресурсам. В процессе *аутентификации* выясняется, действительно ли пользователь тот, за кого себя выдает, а *авторизация* необходима для определения действий в системе, разрешенных *аутентифицированному* пользователю. Сервисы безопасности так-

же обеспечивают возможности шифрования для защиты информации в процессе ее передачи или хранения. *Аудит* предусматривает регистрацию операций (или попыток их выполнения) пользователей в системе.

- **Стандарты взаимодействия сервисов.** Платформы и стандарты взаимодействия взаимосвязаны. В процессе выбора стандарта взаимодействия сервисов команда должна взвесить возможности межплатформенной интеграции и требования по производительности. Например, *Microsoft .NET Framework* сильно упрощает разработку приложений в распределенной среде. Кроме того, поддержка межязыковой интеграции позволяет добиться высокой производительности в средах, где активно используются сценарии и программы на интерпретируемых языках, а также обеспечивает упрощенную модель взаимодействия **компонентов**.
- **Доступ к данным.** Выбирая сервис доступа к данным, принимайте во внимание производительность, поддержку стандартов, общее направление развития технологий предприятия, управление доступом к данным и разнообразие поддерживаемых хранилищ данных. **Технологии-кандидаты** для реализации сервиса доступа к данным иногда могут требовать поддержки определенных отраслевых стандартов или возможности эффективного использования сервиса независимо от источника данных. Сервисы доступа к данным не должны ограничивать группу разработчиков **одним-единственным** языком программирования или особым клиентским интерфейсом. Определенная независимость позволяет команде разработки использовать наиболее эффективные инструменты для обработки данных, не задаваясь тем, как осуществляется доступ к данным.
- **Хранение данных.** Системы хранения данных отвечают за хранение всей бизнес-информации, в том числе информации о сотрудниках, **компаниях** и клиентах. Выбирая хранилище данных, оцените несколько продуктов. Решение следует основывать на единой структуре и одном месте хранения информации. Местонахождение информации очень важно. Данные могут располагаться на одном компьютере, на ферме серверов или на **клиентских** компьютерах. Местонахождение хранилища данных напрямую влияет на производительность приложения и способ его разработки.
- **Системные сервисы.** Проектная команда должна определить необходимые системные сервисы и технологии, которые будут эти сервисы поддерживать. Системные сервисы также часто обеспечивают сервисы высокой доступности, такие, как устойчивость к сбоям и балансировка нагрузки. Раньше многие системные сервисы программировались в самом приложении, но сейчас большинство систем предоставляют системные сервисы, не зависящие от конкретного приложения.

Вот некоторые примеры системных сервисов.

- **Транзакционные** сервисы обеспечивают механизмы и среду для приложений, основанных на транзакциях. В **транзакционной** системе последовательность шагов может группироваться и выполняться как единая транзакция. Если произошел сбой на любом из шагов, вся транзакция терпит сбой и отменяется (или, другими словами, выполняется откат).
- Асинхронные коммуникационные сервисы не требуют немедленной связи. Они базируются на механизме, основанном на **сообщениях**. Приложению, посылающему запрос, необходима гарантированная доставка сообщения в удаленную систему, но его работа никак не зависит от наличия (или отсутствия) ответа в течение определенного периода времени.

- Сервисы обмена сообщениями — это нечто большее, чем простая электронная почта. Они отвечают за маршрутизацию информации, срочность доставки которой не критична. Сервисы организации очередей сообщений позволяют размещать сообщения в очередях, если адресат не доступен. А значит, приложение может посылать сообщения получателю вне зависимости от его доступности.
- **Инструментальные средства разработки** применяются для программирования различных частей приложения, таких, как сервисы и компоненты. В инструментальных средствах обычно предусмотрены интегрированная среда разработки, средства по управлению исходным кодом, мастера и библиотеки исходного кода. Эти средства помогают значительно сократить время, затрачиваемое на создание приложений. Выбирая средства разработки, проектная команда должна учитывать несколько факторов.
 - **Навыки разработчиков.** Для освоения новых инструментов требуется дополнительное обучение и время на изучение, что влияет на календарный план и ограничения на ресурсы.
 - **Особые требования проекта** обычно диктуют выбор инструмента, для применения которого придется обучать команду или даже привлекать сторонние команды разработчиков (аутсорсинг). Одни инструментальные средства более просты в работе, другие — посложнее. Принимайте во внимание время, которое придется затратить на реализацию проекта при условии использования того или иного средства разработки. Часто необходимо искать компромисс между сложностью средства разработки и тем, насколько эффективным получится конечный продукт. Например, на языке ассемблера можно создавать исключительно быстрые приложения, но на подобную разработку потребуется куда больше времени и сил, чем при программировании того же приложения на высокоуровневом языке, например Microsoft Visual Basic .NET.
 - **Возможность применения различных языков для различных задач в проекте.** Выбирая язык программирования, убедитесь, что он поддерживает дизайн и реализацию *слабо связанных компонентов* (loosely coupled components), которые при необходимости заменяются или обновляются. Но прежде чем решиться на применение разных языков для разных задач, оцените имеющийся опыт, знания и навыки разработчиков и проблемы сопровождения, которые могут возникнуть после развертывания продукта. Например, .NET Framework поддерживает несколько языков программирования. «Многоязычие» также повышает продуктивность программистов, поскольку они вправе выбрать близкий и знакомый им язык программирования.
 - Помните, что **выбор средства разработки влияет на другие технологические решения.** Например, для инструмента (или для решения, созданного с его применением) может потребоваться особая ОС. Microsoft ASP.NET предоставляет независимую от языка платформу для построения Web-приложений, но сама эта среда не является платформенно-независимой. Приложения должны работать в ОС семейства Windows.
- **Операционные системы.** Выбирая операционную систему учитывайте, что наличие в ней нужных сервисов способно значительно упростить задачу и снизить объем работ по программированию приложения. Кроме того, выполнение требований по безопасности и масштабируемости подчас также прекрасно делегируются операционной системе. Выбор ОС часто зависит от типов

устройств: КПК, рабочих станций, серверов и специализированных вычислительных устройств.

Внимание! Все определенные на этом этапе технологии документируются в разделе технической спецификации, которая может быть частью любого проектного документа, будь то план проекта или функциональная спецификация.

Определение предварительных бизнес-объектов

Объекты — это люди или предметы, описанные в сценариях использования системы. На основе объектов создаются сервисы, атрибуты и отношения.

На рис. 5-4 показан общий процесс идентификации объектов, сервисов, атрибутов и отношений на стадии анализа в процессе логического дизайна.

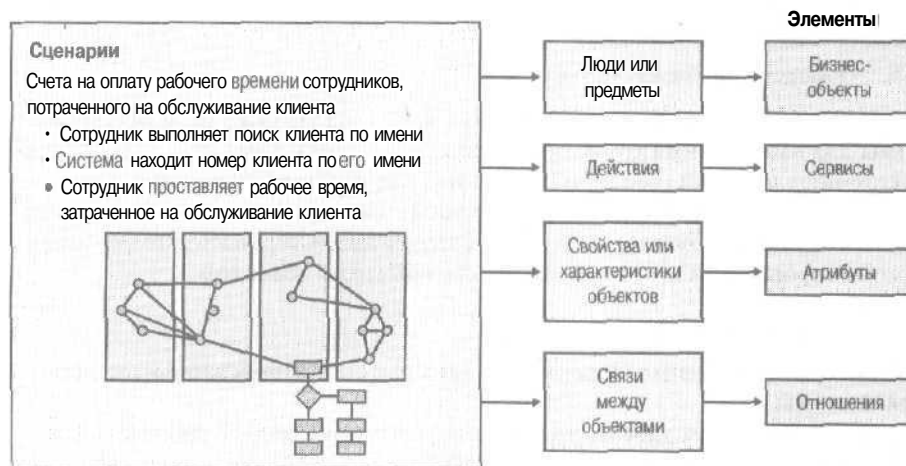


Рис. 5-4. Стадия анализа в процессе создания логического дизайна: определение объектов, сервисов, атрибутов и отношений

Определение объектов

Необходимо определить бизнес-объекты, или компоненты, которые будут поддерживать функции, выполняемые приложением. Посмотрите на сценарии использования системы, созданные в процессе концептуального дизайна. Они помогут определить эти объекты (рис. 5-5). Далее следует продумать поведение объектов и атрибуты, а также отношения с другими объектами.

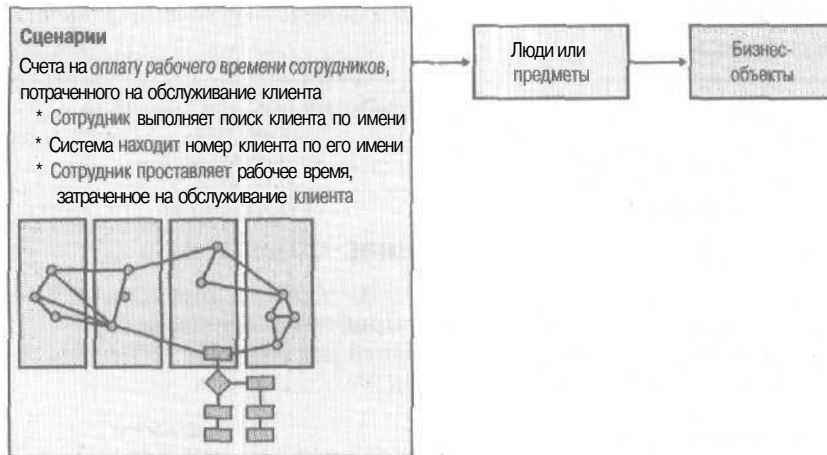


Рис. 5-5. Определение бизнес-объектов

Некоторые СИС не содержат объектов в явном виде, даже если объекты необходимы для выполнения требуемых бизнес-операций. Объекты подчас скрыты в тексте предложений (все зависит от того, как написан сценарий). Поищите скрытые объекты в структурах, системах, устройствах, предметах и событиях. Чтобы выявить скрытые объекты, сценарии следует анализировать с точки зрения требуемой информации и поведения, не связанного с объектом.

Примеры бизнес-объектов

Поясним концепцию бизнес-объектов на примере следующих вариантов использования системы.

- Сотрудник заполняет таблицу, указывая подлежащие оплате рабочие часы.
- Сотрудник создает контракт с клиентом.
- Сотрудник просматривает предыдущие платежи этого клиента.
- Сотрудник привязывает оплачиваемое время к номеру клиента.

На основании этих ВИС удастся определить следующие бизнес-объекты:

- *сотрудник* выполняет действия в системе;
- *клиент* является субъектом действий сотрудника;
- *таблица* ~ это метод регистрации времени, потраченного сотрудником на выполнение задания;
- *контракт* — это соглашение между сотрудником и клиентом;
- *платежи* — счета-фактуры, выставленные клиенту за выполненную работу;
- *номер клиента* — способ идентификации клиента в системе.

Что такое сервисы

Сервис — это определенная схема поведения, которой должен следовать бизнес-объект, будь то операции, функции или преобразования как самого объекта, так и модификация им других объектов. Сервисы используются для реализации бизнес-правил, манипулирования данными и доступа к информации. Сервис может выполнять любые действия, которые удастся описать набором правил.

Определение сервисов

Чтобы определить сервисы объекта, следует вернуться к сценарию использования системы и выяснить, что должен делать объект, какие типы данных поддерживать и какие операции выполнять. Объект, работающий с информацией, также выполняет с ней различные операции, например вычисляет суммарные значения или определяет стоимость доставки.

Рис. 5-6 иллюстрирует определение действий, то есть сервисов, на основе СИС.



Рис. 5-6. Определение сервисов

Выявленный сервис сопоставляется **соответствующему** объекту. Объект в сценарии использования системы является либо субъектом действия, либо отвечает за его выполнение. Если трудно определить объект, чтобы назначить сервис, постарайтесь просмотреть все возможности, внимательно изучив ход сценария, шаг за шагом.

Сформулируйте возможности и обязанности сервиса в максимально **широких** понятиях. Кроме того, потребуется назначить сервису имя, определяющее его точно и **однозначно**. Если на этом этапе возникают затруднения, то назначение сервиса недостаточно глубоко понято и требует дальнейшей проработки в концептуальном дизайне.

Примеры сервисов

На рис. 5-7 показаны примеры сервисов, определенные на основе СИС.

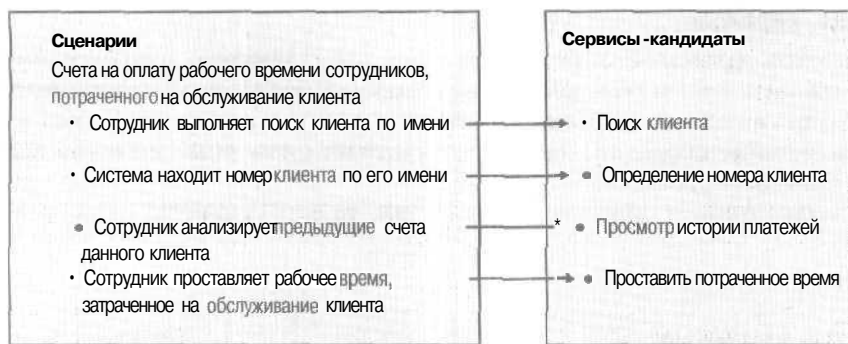


Рис. 5-7. Определение сервисов на основе СИС

В описании показанных сценариев.

- **Сотрудник ищет клиента по имени.** Этот СИС соответствует сервису *Поиск клиента* бизнес-объекта *Сотрудник*.
- **Система определяет номер клиента.** Этот СИС соответствует сервису *Определить номер клиента* бизнес-объекта *Клиент*.
- **Сотрудник регистрирует подлежащие оплате часы в таблице.** Этот СИС соответствует сервису *Заполнить таблицы* бизнес-объекта *Сотрудник*.
- **Сотрудник просматривает предыдущие счета, выставленные данному клиенту.** Этот СИС соответствует сервису *Просмотреть историю платежей* бизнес-объекта *Сотрудник*.
- **Сотрудник регистрирует время, затраченное на клиента данным номером.** Этот СИС соответствует сервису *Проставить потраченное время* бизнес-объекта *Сотрудник*.

Как определяются атрибуты

Атрибуты, или *свойства* объекта, — это определения данных, которые содержит объект. У каждого объекта собственный набор значений. Например, *Имя* один из атрибутов бизнес-объекта *Сотрудник*. У одного объекта атрибут *Имя* может принимать значение *Джон*, а в другом — *Джанет*. Набор значений атрибутов объекта в каждый момент времени называется его *состоянием*.

Определение атрибутов

Чтобы определить атрибуты объекта, вернитесь к сценариям использования системы: поищите слова или фразы, содержащие дополнительные определения объекта; например, длина моста, имя человека, название бренда и модель компьютера — все это атрибуты.

Обычно фактические свойства объектов описаны явно и предоставляют больше информации для определения атрибутов, чем СИС. Для корректного определения атрибутов члены команды используют знания об окружающем мире и опыт в предметной области. Например, если атрибут *Имя* определяется на основе СИС, то на основе этой информации члены команды могут разбить его на два: *Имя* и *Фамилия*. Часто такой уровень *детализации* достигается на стадии физического дизайна, однако, если у команды достаточно информации, чтобы приступить к этой операции раньше, ее можно начать во время логического дизайна.

Примечание У объектов может оказаться масса атрибутов, но проектная команда должна учитывать только имеющие отношение к делу. Например, атрибуты *Имя* и *Фамилия* присутствуют в большинстве приложений, а вот *Рост* и *Вес* скорее всего понадобятся лишь в системах для медицинских организаций. Чтобы не упустить важные атрибуты, документируйте на этом этапе их все — ненужные удалите позднее.

В процессе определения атрибутов анализируйте каждый бизнес-объект и попытайтесь ответить на несколько вопросов.

- Как описывается объект, в том числе целиком и как часть продукта?
- Как объект описывается в контексте функций продукта?
- Какую информацию содержит объект?
- Какую информацию объект должен поддерживать с течением времени?
- В каких состояниях может находиться объект?

Атрибуты, как правило, определяются в связи с объектом, Вам следует четко отмечать каждый атрибут, чтобы не путать их. Кроме того, записывайте структуру или тип атрибута, например текстовый он или числовой.

Совет В большинстве случаев атрибуты являются производными, а их вычисление должно записываться в виде сервиса или объекта. После создания списка атрибутов тщательно изучите каждый из них. Если одни атрибуты абсолютно не связаны с другими атрибутами одного объекта, то, может, стоит создать новый бизнес-объект.

Примеры атрибутов

Рассмотрим варианты использования системы в СИС, описывающем обработку заказа на поставку:

- у клиента есть номер счета, имя и адрес;
- для успешного выполнения задачи необходимо, чтобы клиент был кредитоспособным;
- в зависимости от кредитной истории клиент может запросить консультанта, который обслуживал его в последний раз.

На основании этих ВИС в таблице 5-3 даны определения бизнес-объектов с их атрибутами и значениями.

Таблица 5-3. Пример объекта с атрибутами и значениями

Бизнес-объект	Атрибут	Значения в одном из состояний
Клиент	Номер учетной записи	10076
	Название	Contoso, Ltd.
	Адрес	123 East Main
	Кредитоспособность	Положительна
	Последний консультант	Грег Чепмен

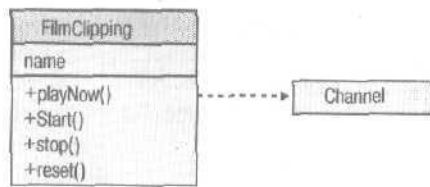
Как определяются отношения

Отношения отображают связи объектов между собой. Язык UML определяет четыре типа отношений: зависимость, обобщение, ассоциация и реализация.

- **Зависимость** (dependency) — *отношение* между двумя объектами, при котором изменение в одном объекте (независимом) оказывает влияние на поведение или сервис другого объекта (зависимого). Это отношение применяется, когда требуется показать, что один объект использует другой. Например, нагреватель воды зависит от водопровода. На UML-диаграммах зависимость между двумя логическими объектами показывается пунктирной линией со стрелкой.
- **Ассоциация** (association) — структурное отношение, описывающее соединение объектов. *Агрегация* — особый тип ассоциации, представляющий отношения между целым и его частями. Например, «заказ содержит параметры». Как правило, при агрегации целое управляет «жизнью» своих частей. Такая форма отношений называется *композицией*. Графически ассоциация изображается сплошной линией с пустым ромбом на конце, а композиция — сплошной линией с закрашенным ромбом на *конце*.
- **Обобщение** (generalization) — отношение между объединяющим объектом (называемым родителем) и специализированным или характерным объектом (называемым потомком). Например, класс *Менеджер* является особым (характерным) типом класса *Сотрудник*. Потомок наследует свойства своих родителей, в первую очередь атрибуты и операции. Обобщение означает, что потомок может замещать родителя в любых взаимодействиях, но обратное не верно. На UML-диаграммах обобщение изображается в виде сплошной линии с незакрашенной стрелкой, указывающей на родителя.
- **Реализация** (realization) — отношения между классами, в которой один абстрактный класс определяет контракт, который другой класс должен выполнять. При моделировании вы обнаружите массу абстракций, представляющих предметы реального мира и создаваемого продукта, например класс *Клиент* в Web-системе обработки заказов. Каждая из этих абстракций может существовать в виде отдельных экземпляров. В общем случае элементы моделирования, на основании которых могут создаваться экземпляры, называются *классами*. У класса свои особенности структуры и поведения. Все экземпляры класса характеризуются едиными правилами поведения, но значения их атрибутов обычно различаются.

Отношения реализации существуют между интерфейсами и реализующими их классами и компонентами, а также между ВИС и диаграммами сотрудничества (collaboration). Графически обобщение изображается как нечто среднее между обобщением и зависимостью — пунктирная линия и незакрашенная стрелка, указывающая на родителя.

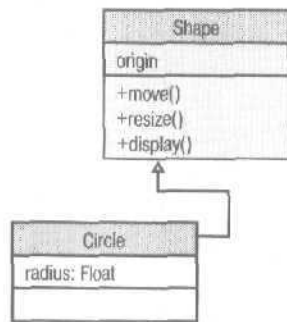
Рис. 5-8 иллюстрирует графическое представление четырех типов отношений.



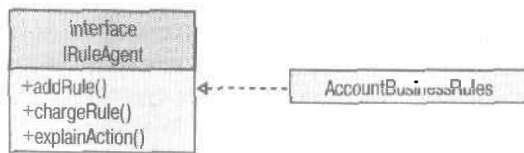
Зависимость:
отношение между объектами
«набор телепрограмм» и «канал»



Ассоциация:
отношение между объектами
-сотрудник» и «компания»



Обобщение:
отношение между объектами
«круг» и «геометрическая фигура»



Реализация:
отношение между объектами
«интерфейс» и «метод»

Рис. 5-8. Типы отношений в UML

Рис. 5-9 иллюстрирует два типа ассоциации — агрегацию и композицию.

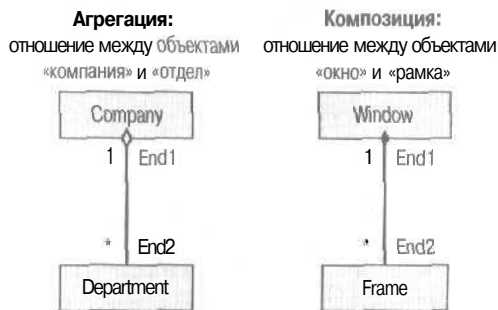


Рис. 5-9. Агрегация и композиция

Определение отношений на основе ВИС

Чтобы определить отношения по ВИС, изучите сценарий использования, стараясь выделить информацию о физическом расположении, направленных дей-

Занятие 3. Документирование результатов логического дизайна

Результатом логического дизайна становится множество информации, которая применяется для создания дизайна решения, например: логическая модель объектов, логическая модель данных и высокоуровневый дизайн пользовательского интерфейса. Существуют несколько инструментов для документирования результатов и создания диаграмм. На этом занятии рассказано, как создавать и документировать результаты логического дизайна.

За дополнительной информацией о создании диаграмм на основе логического дизайна обращайтесь к главе 2.

Изучив материал этого занятия, вы сможете:

- ✓ использовать CRC-карты для изображения взаимодействия между объектами;
- ✓ использовать диаграммы последовательностей для изображения взаимодействия между объектами;
- ✓ создать логическую модель объектов для изображения взаимодействия между ними;
- ✓ создать высокоуровневый дизайн пользовательского интерфейса;
- ✓ создать высокоуровневый проект базы данных.

Продолжительность занятия — около 20 минут.

Моделирование отношений

Существует несколько методов документирования результатов логического дизайна, в том числе CRC-карты (Class-Responsibility-Collaboration) и *диаграммы последовательностей* (sequence diagrams).

Примечание Вы вправе комбинировать CRC-карты и диаграммы последовательностей для проверки объектов и их отношений, обнаруженных в ходе анализа.

CRC-карты

CRC-карты (Class-Responsibility-Collaboration) помогают команде сосредоточиться на высокоуровневых обязанностях класса без детализации методов и атрибутов. CRC-карты используют при мозговом штурме для выявления обязанностей класса, а на основании последних — его сервисов. В CRC-картах указываются все классы, с которыми связан и должен взаимодействовать данный класс, и определяются отношения между классами. Проверяют CRC-карты, воссоздавая на их основе сценарий использования системы.

Пример CRC-карты

Заказ

Роли: Поддержка информации о заказе

Обязанности

Сотрудничество

Определить наличие товара на складе

Получить информацию о складских запасах по товару, определенному в заказе

Определить цену

Проверить платеж

Получить адрес у клиента

Отправить по адресу доставки

Проблемы

Где хранить информацию о заказе?

Диаграммы последовательности

Диаграммы последовательности (sequence diagrams) показывают действующие лица и объекты, участвующие во взаимодействии, вместе с инициируемыми ими событиями в хронологическом порядке. Вертикальная линия на диаграмме последовательностей обозначает время жизни объекта. Стрелка между временем жизни двух объектов обозначает сообщение, форму взаимодействия между ними. Получение сообщения обычно рассматривается как событие. Порядок появления сообщений показан на странице сверху вниз.

В дизайне решения сложно понять поток управления и последовательность действий целиком. Для этого и предназначены диаграммы последовательности, они помогают ясно увидеть последовательность событий.

На рис. 5-12 показана диаграмма последовательности в логическом дизайне для компании Adventure Works Cycles.

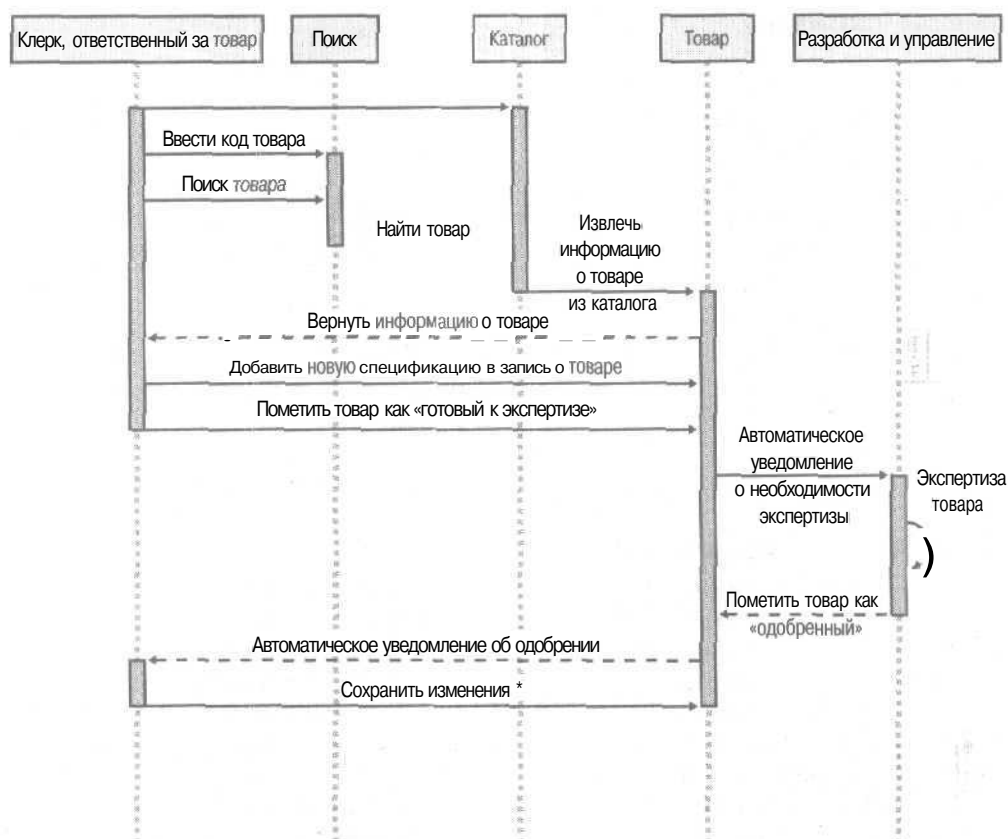


Рис. 5-12. Диаграмма последовательности

Как создается логическая модель объектов

Главный результат логического дизайна — проект решения в виде логической модели объектов. Последняя формируется из объектов, сервисов, атрибутов и отношений, созданных ранее в процессе логического дизайна,

Особенности построения логической модели объектов

Создавая логическую модель объектов, важно принять во внимание все существенные требования бизнеса и пользователей: безопасность, поддержку других языков, локализацию, аудит и ведение журнала, обработку ошибок, интеграцию с существующими системами и управление состоянием. Также, создавая логическую модель объектов, важно учесть все ограничения бизнеса.

Пример логической модели объектов

На рис. 5-13 показан пример логической модели объектов для компании Adventure Works Cycles. Здесь учтены требования по безопасности и ведению журнала, определенные в процессе концептуального дизайна.

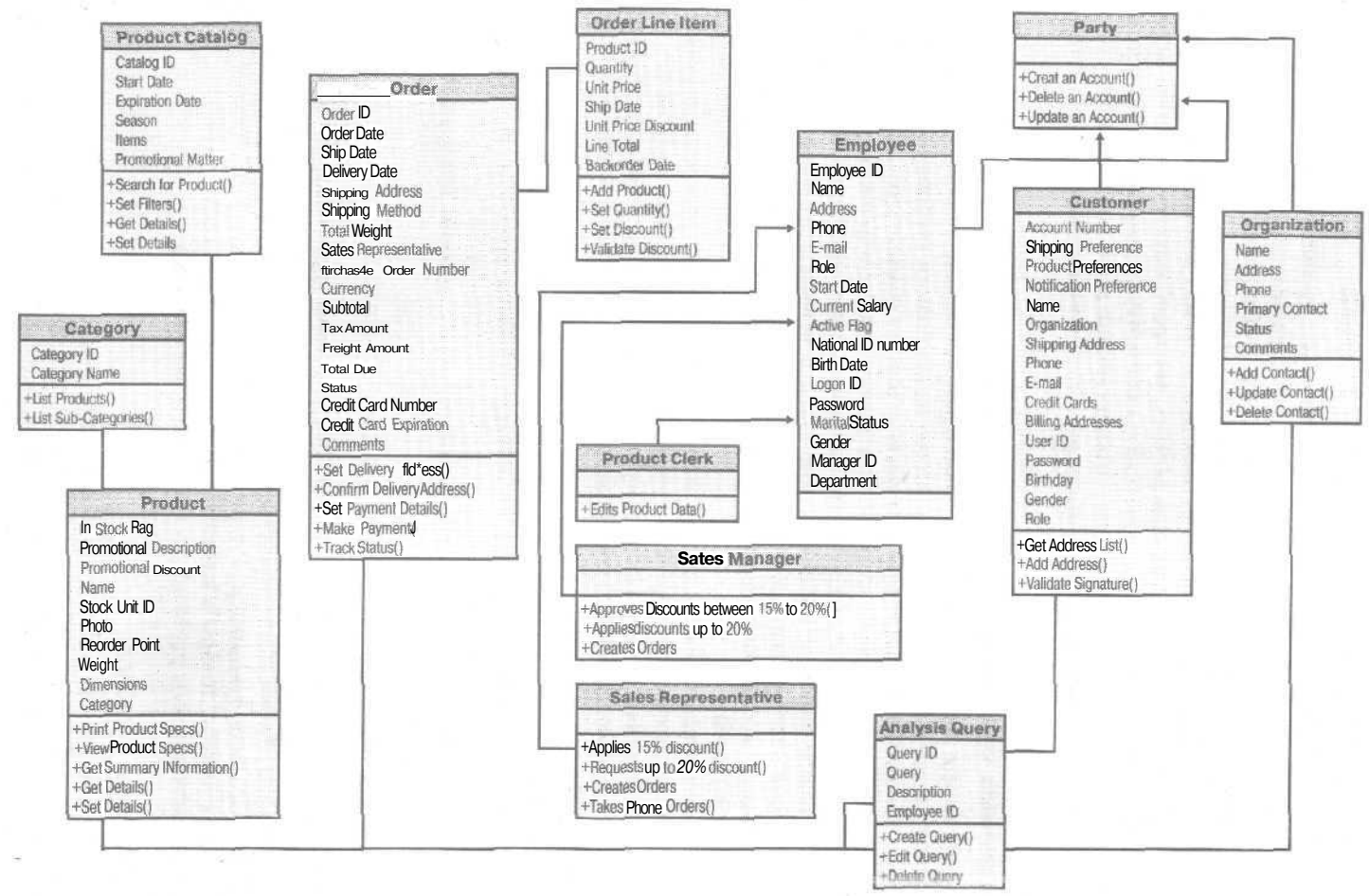


Рис. 5-13. Логическая модель объектов

Как создается логическая модель данных

Для представления логического дизайна используют логическую модель объектов или данных. Однако проектная команда иногда создает обе модели, представляя логический проект с разных сторон. Это необходимо, когда одна из моделей представляет какую-либо часть проекта особо четко.

Логический дизайн — это промежуточный этап между концептуальным и физическим дизайном. Создавая модель данных, вы преобразуете концептуальные требования к данным (они определяются при концептуальном дизайне) в реальные объекты-сущности и отношения, отображающие реальное взаимодействие данных. Полученная информация помогает в дальнейшем моделировать физический дизайн.

Определение сущностей и атрибутов

При переходе к логической стадии дизайна данных одна из первоочередных задач заключается в формулировке сущностей на основании требований к данным и другой связанной информации. *Сущностью* (entity) обычно считают человека, место, элемент или понятие, которое определяет данные или о котором данные собираются и хранятся. Атрибут — это характеристика, представляющая собой дополнительное определение и описание свойств экземпляра сущности. У сущности обычно несколько атрибутов.

Вот пример, в котором выделенные курсивом слова показывают кандидаты на роль сущностей:

«Консультанты еженедельно вносят отработанные ими часы в *табель*. Табель затем пересылается *помощнику по административной работе*, который вносит *часы* в программу выставления счетов. *Помощник по административной работе* затем пересылает *счета* за отработанные часы *клиентам*».

После определения сущностей следует определить необходимые атрибуты — они описывают сущности решения. Например, атрибуты автомобиля — его цвет, марка, модель и год выпуска.

При реализации физического дизайна атрибуты обычно превращаются в столбцы таблиц базы данных.

Определение таблицы

Объекты, определенные на стадии анализа, — весьма вероятные кандидаты на роль таблиц и обычно преобразуются в таблицы базы данных во время физического дизайна. Таблицы предназначены для объектов, *информацию* о которых требуется хранить. Например, в простой системе обработки заказов такие объекты, как клиенты, товары и каталог, преобразуются в таблицу, а в более сложных системах объекты размещаются в нескольких таблицах.

Определение столбцов

Атрибуты объекта становятся столбцами таблицы, связанной с объектом. Например, *Имя*, *Фамилия* и *Должность* могут быть атрибутами (а следовательно, и столбцами) таблицы *Сотрудник*. В каждой строке таблицы хранятся поля одного экземпляра объекта.

Определение отношений

Далее необходимо определить все ассоциации между таблицами — они представляют отношения между объектами. Например, между Employee (Сотрудник) и Department (Отдел) существует связь — в каждом отделе есть сотрудники и каждый сотрудник относится к отделу. Можете также использовать *численность* (cardinality) и *множественность* (multiplicity) для более точного определения связи.

Численность — это определяющее свойство связи, которое указывает количество экземпляров сущности, разрешенных с каждой стороны отношения. Например, в каждый момент времени консультант может работать над не более, чем одним проектом. *Множественность* определяет диапазон численностей, который допускает сущность.

Подробнее об определении отношений — в главе 8.

Примечание Располагая достаточной информацией, в процессе логического дизайна можно создать логическую модель данных, включающую столбцы и определения основных и внешних ключей. Однако такой уровень детализации обычно достигается на стадии физического дизайна.

На рис. 5-14 показан типичный логический дизайн базы данных.

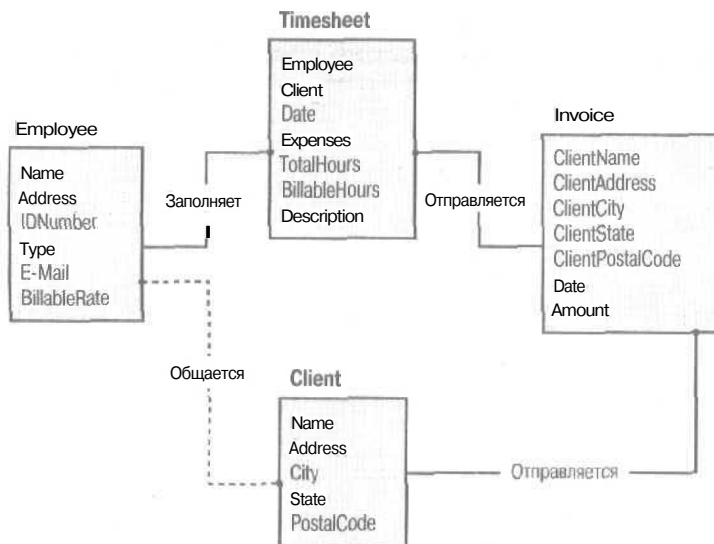


Рис. 5-14. Логический дизайн базы данных

Создание высокоуровневого дизайна пользовательского интерфейса

Используя объекты, сервисы, атрибуты и отношения, определенные на стадии анализа, команда переходит к созданию высокоуровневого дизайна пользовательского интерфейса и базы данных, то есть описывает порядок работы готового продукта с точки зрения конечного пользователя.

Список объектов и сервисов позволяет понять, какие функции ожидают от системы пользователи. Обычно эта информация необходима для дизайна элементов пользовательского интерфейса — кнопок, полей и меню.

Примечание В большинстве проектов львиная доля дизайна пользовательского интерфейса выполняется в процессе физического дизайна. Однако, если команда считает, что располагает достаточной информацией, эту работу можно выполнить и на стадии логического дизайна.

Вот пример СИС:

Клиент выбирает Каталог для просмотра. В корне выбранного каталога отображаются Категории и Товары. Клиент может выбрать Товар для просмотра подробной информации о нем или Категорию - для просмотра Товаров и подкатегорий в ней.

На основании этого сценария команда создает следующий высокоуровневый дизайн пользовательского интерфейса:

- страница «Информация о продукте», где отображается информация о продуктах: описание, цена и наличие на складе;
- страница «Информация о клиенте»: регистрационные и личные данные.

Сценарий можно расширить, предоставив клиенту возможность покупки выбранного товара. В этом случае в пользовательском интерфейсе необходимо предусмотреть:

- страницу «Информация о заказе», «Оформление заказа» и «Состояние заказа»;
- историю о предыдущих покупках клиента.

Примечание Подробнее о дизайне пользовательского интерфейса — в главе 7.

Занятие 4. Оптимизация логического дизайна

На стадии анализа проектная команда разбирает сценарии использования системы и определяет решение в виде объектов, сервисов, атрибутов и отношений между объектами. Далее переходят к стадии оптимизации, основная задача которой — уточнить и оптимизировать дизайн, проверив его на предмет соответствия СИС и требованиям.

На этом занятии вы узнаете, как уточняется список объектов, а также как проверяется бизнес-модель и выбирается модель управления в логическом дизайне.

Изучив материал этого занятия, вы сможете:

- ✓ уточнить список определенных при анализе объектов;
- ✓ проверить существующую бизнес-модель;
- ✓ выбрать модель управления в модели логического дизайна.

Продолжительность занятия — около 15 минут.

Уточнение множества объектов

На стадии анализа проектная команда определяет и уточняет, какие объекты действительно необходимы для создания системы. Выбирая необходимый набор объектов, принимайте во внимание ряд соображений;

- если два объекта предоставляют одинаковую информацию или управляют одним действием, их следует объединить в единый объект и дать ему более содержательное имя;
- объект должен быть характерным. Некоторые кандидаты на роль объектов создают на основе дополнительной информации, полученной из сценариев. Подобные объекты не обязательно некорректны, но их все-таки желательно сделать «реальными», менее абстрактными. Кроме того, стоит выяснить, укладываются ли они в границы проекта;
- если атрибут должен существовать в решении сам по себе, переназначьте его как объект;
- иногда приходится создать новый объект(ы) для управления и координации набора сервисов;
- для уточнения списка объектов иногда полезны сервисы. Для этого в каждом сервисе определите объекты, которыми он оперирует.

Проверка существующей логической модели объектов

Существующая логическая модель объектов проверяется на предмет соответствия требованиям. Также применяется индивидуальная проверка объектов и прототип СИС.

Проверка модели на предмет соответствия требованиям

Это наиболее важная задача при оптимизации и завершении работы над логическим дизайном. Следует убедиться, что в логической модели объектов учтены все требования. При обнаружении новых или неохваченных требований их следует немедленно учесть в модели. Только после этого можно переходить к физическому дизайну.

Внимание! Логическая модель данных, в которой пропущено хотя бы одно требование, считается незаконченной.

Индивидуальная проверка объектов

В ходе этой проверки определяются входные и выходные данные объекта, а также возможности и набор функций, который он должен предоставлять. Выходные данные и поведение должны точно прогнозироваться для любых входных данных. Также следует проверить независимые части объекта.

Индивидуальная проверка объектов упрощает интеграцию систем, так как позволяет независимо тестировать отдельные части системы перед их объединением в единый продукт. Если проверка показала корректность составляющих частей, проектная команда может смело приступать к сборке системы.

Прогон СИС

Метод индивидуальной проверки объектов позволяет проверить независимые функции бизнес-объекта, тем не менее проектной команде следует выяснить, как совокупность объектов решает задачи, описанные в сценариях. Здесь возможны неординарные ситуации. Взаимозависимость объектов уточняется, когда вы выполните весь сценарий и удостоверитесь, что абсолютно все требования сценария удовлетворяются одной из комбинаций объектов. Один из вариантов — организовать ролевую игру по сценарию: назначить на роли объектов членов команды и «прокрутить» сценарий. Ролевая игра также позволяет обнаружить нестыковки, возникающие из-за того, что разные люди по-разному интерпретируют задачи того или иного сервиса.

Проверяя в каждый момент времени по одному шагу сценария, удастся определить, какие сервисы нужны и в какой последовательности. Двигаясь от отправной точки СИС и до его конца, выясните все объекты, сервисы которых необходимы для выполнения сценария.

Любое сообщение, которыми обмениваются объекты, — это взаимодействие между объектами. Чтобы определить тип взаимодействия, проанализируйте запросы, отправляемые каждым объектом, и ответьте на следующие вопросы:

- располагает ли объект достаточной информацией, чтобы вернуть ответ;
- откуда пришла информация, содержащаяся в запросе: из данных внутреннего состояния объекта или из запроса, инициировавшего выполнение операции;
- полагается ли объект на внутреннее состояние объекта-сервера, обрабатывающего запрос;
- есть ли в информации, отправленной от источника к приемнику, ссылки на объекты или иные части системы.

Также проектная команда должна изучить входные параметры каждого запроса и определить, предоставляют ли они информацию, чувствительную к контексту, в которой объект-приемник не нуждается. Следует также выявить случаи зависимости данных от внешнего контекста — это иногда не позволяет повторно использовать объект в других контекстах. Указанные проблемы решаются путем перераспределения обязанностей между объектами или создания новых объектов.

Прогоняя сценарий, обратите внимание на ряд моментов.

- Выявите все случаи зависимости операции от существования или целостности определенных бизнес-объектов.
- Выясните, где возможны проблемы, связанные с целостностью, последовательностью или параллельностью операций. Требуется ли последовательный порядок выполнения операции во всех частях транзакции?
- Определите все критические временные отрезки. Необходимо ли направлять ответ немедленно или действие можно отложить?
- Учтите все организационные вопросы, такие, как выполнение транзакции в нескольких функциональных областях одновременно.
- Выявите бизнес-правила, которые реализованы более, чем в одном бизнес-объекте.
- Определите требования к аудиту и контролю. Определите ответственных лиц. Проверьте ситуации, в которых ответственность распределяется между несколькими сотрудниками.
- Выясните частоту выполнения операции: постоянно или периодически?
- Определите местоположения и зависимости между ними.
- Определите, зависит ли сервис, управляющий транзакциями от сервисов, реализованных в настоящее время в других бизнес-объектах.

Выбор модели управления в логическом дизайне

Исследование взаимодействия объектов в различных сценариях показывает, что операции должны выполняться в определенной последовательности. Один из возможных способов такого исследования — использовать диаграммы состояний. Выяснив поток управления, выстраивают последовательность взаимодействий объектов. Управление в логическом дизайне:

- обеспечивает транзактную целостность сценария;
- координирует сервисы между объектами;
- определяет взаимозависимости объектов.

Для успешного выполнения операции часто требуется соблюдать определенный порядок действий. Иногда события возникают одновременно. В некоторых сервисах предусмотрены периоды ожидания, которые необходимо выдержать до продолжения обработки. Эти реалии должна учитывать проектная команда в процессе логического дизайна, чтобы гарантировать выполнение нужных действий в подходящие моменты времени и в нужной последовательности, а также с учетом других действий, происходящих в системе.

Управление может быть синхронным и асинхронным. *Синхронное* реализуется, когда, обращаясь к сервису объекта, вызывающая сторона ждет возвращения управления. В таких сценариях управление передается от вызывающего объекта вызываемому и операция выполняется немедленно. *Вызывающий* объект приостанавливает выполнение, пока вызванный объект не завершит работу. По завершении операции управление возвращается вызываемому объекту.

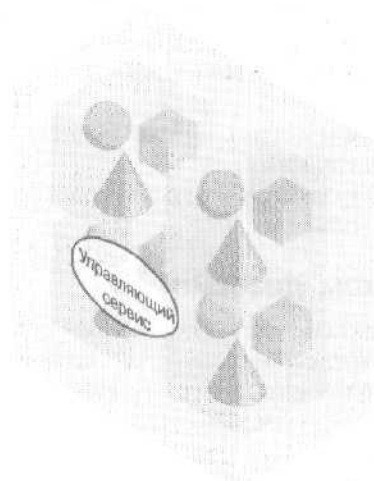
При *асинхронном управлении* клиент отправляет запрос и продолжает выполнение других задач. Сервис обрабатывает запрос и уведомляет клиента о завершении обработки. При этом клиент не приостанавливает работу и способен выполнять другие действия, не связанные с данной операцией, пока сервис обрабатывает его запрос. Общее управление не передается от вызывающего объекта вызываемому. Это означает, что вызывающий объект должен эффективно управлять целостностью своих данных, к которым могут обратиться одновременно несколько объектов, и координировать доступ к общим ресурсам.

Совет В распределенных системах рекомендуется создавать дополнительные объекты, которые берут на себя управление, соблюдение последовательности операций и зависимостей между ними. Следует изолировать зависимости и выделить те сервисы, которые часто меняются вместе с изменениями в бизнесе.

Модели управления

Рис. 5-15 иллюстрирует две возможные общие модели управления объектами.

Высокоуровневая инкапсуляция



Управляющий объект

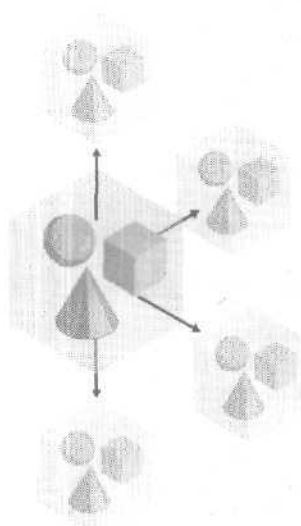


Рис. 5-15. Модели управления

В первой модели используется высокоуровневая инкапсуляция управляемых объектов. Внешний объект владеет и управляет всеми логическими ресурсами, необходимыми для работы внутренних объектов. Поскольку внешний объект содержит все необходимое для работы, проектирование упрощается. Подобный контроль и, возможно, упрощение делает структуру более жесткой, негибкой.

Например, внутренние объекты способны использовать только объекты, располагающиеся на одном уровне с ними или доступные через один из сервисов внешнего объекта. По сути, это однопользовательская модель, поскольку внутренние объекты доступны только через один внешний объект. Внешний объект отвечает за корректную маршрутизацию всех поступающих сообщений между внутренними объектами.

Вторая модель гибче, но сложнее в проектировании. В этой модели создается управляющий объект на одном уровне с управляемыми им ресурсами. Фактически это многопользовательская система — у управляющего объекта нет исключительных прав на объекты, так как они одновременно доступны другим частям системы. Управляющий объект не знает о других взаимодействиях управляемых объектов. В этой модели труднее обеспечить целостность ресурсов.

Пользовательский интерфейс управления

Две модели управления — высокоуровневая инкапсуляция и управляющий объект — не учитывают управление, которое получают пользователи, инициируя события в объектах. Интерфейс интерактивных систем отдает все рычаги управления пользователю. В действительности в большинстве систем комбинируется пользовательское управление и управление, «защитное» в наборе управляющих объектов в системе.

В конечном счете управление и зависимости совместно определяют последовательность событий. Для управления транзакциями применяются объекты, однако последовательность событий инициируют пользователи.

Практикум. Определение объектов в логическом дизайне



При выполнении упражнений по определению объектов для включения в логический дизайн используйте знания, полученные на занятиях.

Упражнение 1. Определение объектов на основании ВИС

Ниже приведен список возможных ВИС для проекта Web-сайта компании Adventure Works Cycles. Изучите каждый ВИС и укажите все объекты, которые следует включить в логический дизайн. Ответы на вопросы вы найдете в приложении.

1. При заполнении корзины покупателя проверяется наличие определенного товара на складе.
2. Новый пользователь создает учетную запись, чтобы размещать заказы.
3. Пользователь добавляет товар в корзину.
4. Пользователь находит и открывает свой заказ для проверки его состояния.

Упражнение 2. Создание матрицы сервисов

Откройте файлы *C05Ex2.doc*, *Usage Scenario 06.1 View Catalog.doc* (сценарий «Просмотр каталога»), *Usage Scenario 06.1.1 Add Items in Catalog.doc* (сценарий «Добавление элементов в каталог») и *Usage Scenario 06.3.1 Edit Items in Catalog.doc* (сценарий «Редактирование элементов каталога») из папки *\SolutionDocuments\Chapter05* на прилагаемом к книге компакт-диске. Используя документы сценариев использования системы, завершите матрицу сервисов в *C05Ex2.vsd* для бизнес-объекта Catalog Items (элементы каталога). Позаботьтесь об определении сервисов, действующих лиц, обязанностей и схем сотрудничества для всех сервисов, а также о соответствии каждой записи одному из СИС. В файле *C05Ex2.doc* вы найдете пример матрицы сервисов. Один из возможных ответов к этому упражнению вы найдете в файле *C05Ex2_Answer.doc*.

Упражнение 3. Создание диаграммы последовательности

Откройте файл Microsoft Visio *C05Ex3.vsd* (папка *\SolutionDocuments\Chapter05*) и перейдите на вкладку Sequence Diagram (диаграмма последовательности). Эта вкладка пуста. Представьте себе, что клиент посетил Web-сайт розничной торговли компании Adventure Works Cycles. Используйте такой СИС для создания диаграммы последовательности, которая иллюстрирует схему взаимодействия между объектами:

Клиент открывает каталог, ищет позиции в каталоге и добавляет товар в новый заказ.

Один из возможных ответов вы найдете в файле *C05Ex3_Answer.vsd*.

Резюме

- Логический дизайн — это процесс описания продукта в терминах организации, структуры, синтаксиса и взаимодействия его частей с точки зрения проектной команды.
- Этап логического дизайна состоит из двух перекрывающихся стадий:
 - стадии анализа, во время которой определяются бизнес-объекты, сервисы, атрибуты и отношения между объектами;
 - стадии оптимизации, во время которой команда уточняет список бизнес-объектов и на основании их определяет новые объекты и сценарии-
- Результаты логического дизайна таковы:
 - логическая модель объектов — набор объектов и соответствующих им сервисов, атрибутов и отношений;
 - высокоуровневый дизайн пользовательского интерфейса;
 - логическая модель данных.
- Объекты — это люди или предметы, описанные в сценариях использования системы.
- Сервис — это схема поведения бизнес-объекта, то есть операции, которые он выполняет.
- Атрибуты — это определение данных (свойств), относящихся к объекту.
- Отношения отображают связи между объектами.
- CRC-карты помогают сконцентрировать внимание на высокоуровневых отношениях класса, а не описывать детально его методы и атрибуты. CRC-карты также помогают определить отношения между различными объектами в логическом дизайне.
- Диаграммы последовательности показывают действующие лица и объекты, участвующие во взаимодействиях, в виде хронологически упорядоченного списка инициируемых ими событий.
- Список объектов и сервисов позволяет создать высокоуровневый дизайн пользовательского интерфейса продукта.
- В процессе создания дизайна базы данных команда определяет:
 - объекты-сущности, атрибуты и отношения;
 - таблицы;
 - столбцы;
 - отношения.
- В процессе проверки проекта исследуют логическую модель объектов на предмет соответствия имеющимся требованиям.
- В процессе индивидуальной проверки объектов определяются входные и выходные данные, возможности объекта, а также набор функций, которые он должен предоставлять.
- В процессе проверки проекта при пошаговом прогоне СИС определяется, какие сервисы необходимы и в какой последовательности.
- Управление позволяет выстроить порядок взаимодействия объектов.
- Синхронное управление реализуется, когда при вызове сервисов объекта вызывающий объект блокируется до возвращения управления.
- Асинхронное управление реализуется, когда при вызове сервисов объекта вызывающий объект продолжает выполнять не связанные с этим вызовом задачи, пока сервис не уведомит о результатах выполнения операции.

- При использовании метода управления с высокоуровневой инкапсуляцией внешний объект владеет и управляет всеми логическими ресурсами, необходимыми для работы внутренних объектов.
- При использовании метода управления с управляющим объектом, управляющий объект функционирует на том же уровне, что и управляемые им ресурсы.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Из каких стадий состоит логический дизайн?
2. Каковы результаты логического дизайна?
3. Следует ли на этапе логического дизайна уделять особое внимание технологиям?
4. Каковы преимущества логического дизайна?
5. Как в СИС определяются сервисы?
6. Как в СИС определяются атрибуты?
7. Что такое диаграмма последовательности?
8. Как проектируются таблицы и столбцы в хранилище данных создаваемой системы?
9. Зачем уточняется список объектов?
10. Как выполняется индивидуальная проверка объектов?
11. В чем цель управления в логическом дизайне?
12. Работая над логическим дизайном системы, вы обнаружили не описанный ранее сценарий. Что следует предпринять?
13. Каковы обязанности роли тестировщика в процессе логического дизайна?

ГЛАВА 6

Создание физического дизайна

Занятие 1. Основные сведения о физическом дизайне	182
Занятие 2. Стадия анализа	190
Занятие 3. Стадия рационализации	196
Занятие 4. Реализация физического дизайна	204
Практикум. Создание физического дизайна	209
Резюме	210
Закрепление материала	212

В этой главе

Наряду с концептуальным и логическим дизайном на этапе планирования проектная команда создает физический дизайн системы. Из этой главы вы узнаете о его назначении и результатах, а также о задачах, из которых он состоит. Здесь также описаны четыре стадии создания физического дизайна: исследование, анализ, рационализация и реализация.

Примечание Как в концептуальном и логическом дизайне, результаты физического дизайна документируются в функциональной спецификации, которая относится к результатам этапа планирования. Подробнее о функциональных спецификациях — в главах 4 и 10.

О физическом дизайне уровня представления рассказывается в главе 7, а об уровне данных — в главе 8.

Прежде всего

Для изучения материалов этой главы необходимо:

- знать модель процессов Microsoft Solutions Framework (MSF);
- знать результаты этапа создания общей картины решения и три стадии концептуального дизайна: исследование, анализ и оптимизацию;
- иметь представление об особенностях этапа логического дизайна в MSF.

Занятие 1. Основные сведения о физическом дизайне

Физический дизайн — последний шаг этапа планирования в модели процессов MSF. Проектная команда переходит к нему после того, как все ее члены подтвердят, что на этапе логического дизайна получено достаточно информации. На этапе физического дизайна на концептуальный и логический дизайн налагаются технологические ограничения. Поскольку физический дизайн «вырастает» из этих двух типов дизайна, его успех сильно зависит от **тщательности** их проработки, кроме того, этот факт гарантирует, что физический дизайн удовлетворяет требования бизнеса и пользователей.

Примечание Чтобы освежить в памяти сведения о трех видах дизайна на этапе планирования, обратитесь к таблице 5-1 из главы 5.

На этом занятии вы узнаете о назначении и целях физического дизайна. Здесь описаны стадии и результаты, а также обязанности различных ролей в физическом дизайне.

Изучив материал этого занятия, вы сможете:

- ✓ описать физический дизайн;
- ✓ рассказать о целях физического дизайна;
- ✓ перечислить стадии физического дизайна;
- ✓ рассказать об **ответственности**, которую несут различные MSF-роли на данном этапе;
- ✓ перечислить результаты физического дизайна;
- ✓ описать назначение и результаты стадии исследования.

Продолжительность занятия — около 15 минут.

Физический дизайн

Это третий этап **при** планировании в модели процессов MSF. Физический дизайн — это описание компонентов, сервисов и технологий продукта с точки зрения требований по разработке с учетом будущих частей решения и их взаимодействия. Рис. 6-1 иллюстрирует место физического дизайна в модели процессов MSF.

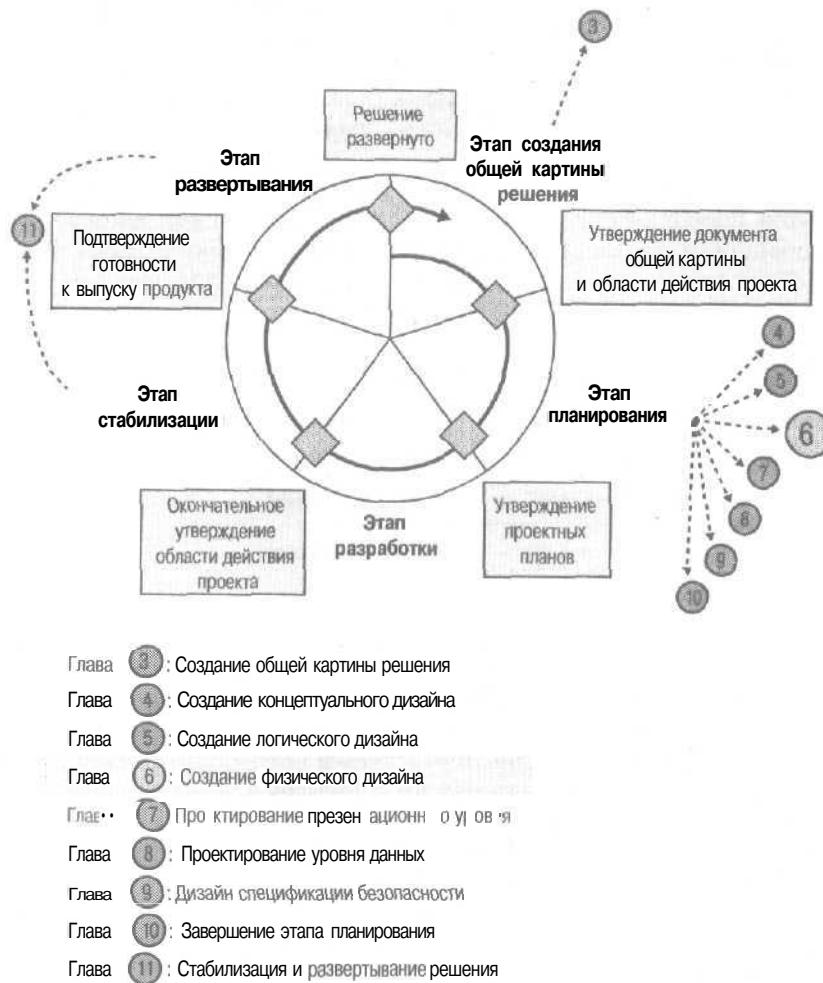


Рис. 6-1. Где мы сейчас находимся в процессе проектирования в MSF

Работая над физическим дизайном, команда создает новые виды дизайна на основе уже построенных и совершенствует архитектуру продукта. В проектах на логическую модель налагаются реальные технологические ограничения, в том числе зависящие от средств разработки и среды развертывания решения. Команда также учитывает такие особенности проекта, как безопасность, доступность, масштабируемость, управляемость и производительность, причем оставаясь при этом в рамках проекта и его требований.

Исходными данными для физического дизайна являются все созданные к тому моменту артефакты, в том числе созданные во время логического дизайна логические модели объектов и данных и высокоуровневый дизайн пользовательского интерфейса. Некоторые артефакты, в том числе и план проекта, могут претерпеть незначительные изменения. С их учетом устанавливаются график прохождения контрольных точек физического дизайна.

В процессе физического дизайна проектная команда сокращает разрыв между логическим дизайном решения и его реализацией, описывая, как предполагается реализовать решение. Цель концептуального и логического дизайна — выявить особенности бизнеса и требований, а также спроектировать продукт в соответствии с ними. Суть физического дизайна — тонкости реализации решения.

В результате физического дизайна команда представляет спецификацию набора компонентов, сборок Microsoft .NET, двоичных файлов и компоновочных библиотек; особенности пользовательского интерфейса решения; схему базы данных, такие объекты базы данных, как триггеры, индексы и хранимые процедуры; а также детальную информацию обо всех отчетах, которые будут создаваться в продукте.

Рис. 6-2 иллюстрирует цель физического дизайна.

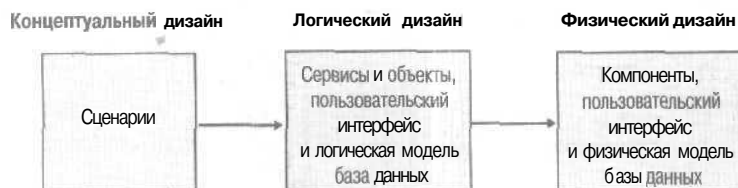


Рис. 6-2. Физический дизайн

В качестве примера рассмотрим проектирование и постройку дома. В логическом дизайне определяются такие требования, как требуемые номинальная мощность электрических коммуникаций, уровень освещенности, давление и расход водопровода. В физическом дизайне определяется необходимый набор электрического оборудования, требования к каждому прибору и электропроводке, а также параметры электрической сети.

При переходе от логического дизайна к физическому команда использует многоуровневую архитектуру, основанную на сервисах.

Примечание Различные архитектуры приложений описываются в главе 4.

Область действия физического дизайна

В таблице 6-1 описаны особенности области действия физического дизайна.

Таблица 6-1. Область действия физического дизайна

Физический дизайн не является	Но позволяет
Программированием	Создать детализированные спецификации компонентов, которые предполагается программировать; определить места размещения компонентов
Развертыванием технологий	Определить технологии, которые предполагается применять для разработки продукта

В процессе физического дизайна проектная команда разрабатывает спецификации компонентов и топологию развертывания, на основании которых группа разработчиков (программистов) будет создавать решение. В процессе построения продукта команда разработки учитывает целевую топологию.

Помните, что в процессе физического дизайна продолжается проектирование решения, а не разработка версии продукта, подлежащей выпуску и разворачиванию.

Различия между логическим и физическим дизайном

Во время логического дизайна проблема рассматривается с точки зрения проектной команды, а во время физического дизайна — с точки зрения группы разработчиков. При работе над логическим дизайном команда документирует бизнес-операции, ограничения и допущения, в процессе физического дизайна — описывает решение, учитывая ограничения выбранных технологий и среды развертывания. В этом смысле в физическом дизайне продукт рассматривается с более «технической» точки зрения.

По существу, физический дизайн — это развитие логического дизайна, ориентированное на реализацию.

Цели физического дизайна

На этапе физического дизайна решается ряд задач.

- **Выявить наиболее подходящие технологии.** В процессе физического дизайна проектная команда оценивает технологии и определяет оптимальные для разрабатываемого решения.
- **Преобразовать логический дизайн в модели физического дизайна.** Выполняя эту работу, команда на основе результатов логического дизайна создает гибкую спецификацию с указанием компонентов. Так, приложение описывается с точки зрения разработчиков, что позволяет приступить к разработке продукта, удовлетворяющего требованиям.
- **Создать базовую версию, основу для начала разработки.** Помимо моделей и стратегий команда определяет роли и обязанности разработчиков, а также процессы.
- **Определение момента, когда достигнута контрольная точка «План проекта утвержден».** Эта контрольная точка свидетельствует об окончании работы над базовым физическим дизайном. В этот момент команда повторно оценивает риски, обновляет приоритеты и завершает сметы ресурсов и календарных графиков.

Роли и обязанности членов команды в процессе физического дизайна

В таблице 6-2 описаны задачи каждой из ролей команды в процессе физического дизайна.

Таблица 6-2. Роли и их обязанности в процессе физического дизайна

Роль	Основные обязанности	Дополнительные обязанности
Менеджеры решения	Управление ожиданиями пользователей и создание плана взаимодействия	Подготовка к разворачиванию решения
Менеджеры программы	Управление процессом физического дизайна и создание функциональной спецификации	Определение плана проекта, в том числе ресурсов, календарных графиков и оценки рисков

(см. следующую страницу)

Таблица 6-2. Роли и их обязанности в процессе физического дизайна (окончание)

Роль	Основные обязанности	Дополнительные обязанности
Разработчики	Представление результатов физического дизайна: проектных моделей, планов, календарных графиков и смет разработки	Оценка технологий, построение прототипов (при необходимости) и подготовка среды разработки
Тестировщики	Оценка и проверка функционального наполнения и целостности физического дизайна на основании СИС	Определение детализированных планов тестирования и подготовка среды для тестирования и контроля качества
Специалисты по удобству использования продукта	Оценка физического дизайна на предмет удовлетворения требований пользователей и разработка плана справочной системы	Разработка плана обучения пользователей
Менеджеры по выпуску	Оценка влияния инфраструктуры на физический дизайн	Описание инфраструктуры и процедурных требований, а также вариантов развертывания

Результаты физического дизайна

По завершении физического дизайна проектная команда располагает достаточным количеством проектной документации, чтобы приступить к созданию продукта. Конкретный состав документации зависит от решения, но мы перечислим самые типичные:

- **диаграммы классов;**
- **модели компонентов, диаграммы последовательностей или диаграммы действий;**
- **схема базы данных;**
- **базовый вариант развертывания**, в том числе:
 - топология сети, отражающая расположение оборудования и его подключение;
 - топология компонентов и данных, которая отражает расположение компонентов, сервисов и хранилищ данных решения, увязанная с топологией сети;
- **спецификации компонентов**, описывающие внутреннюю структуру и интерфейсы компонентов;
- **стратегия объединения сервисов в компоненты и стратегия развертывания компонентов в сети**. Здесь же иногда описывается предварительный план развертывания;
- **модель программирования**, определяющая, как конкретно предполагается реализовывать решение. Здесь описывается выбранная модель поддержки состояний объектов и режимов подключений, организация потоков, порядок обработки ошибок, меры безопасности и порядок документирования кода,

Стадии физического дизайна

Физический дизайн — это последняя фаза этапа планирования, который завершается достижением контрольной точки «Утверждение плана проекта». Как раз перед достижением этой точки проектная команда завершает физический дизайн,

На рис. 6-3 показана связь физического дизайна с концептуальным и логическим дизайном на этапе планирования.

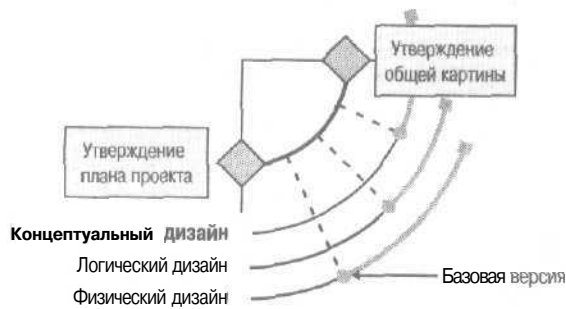


Рис. 6-3. Место физического дизайна на этапе планирования

На рис. 6-4 показаны четыре стадии физического дизайна и связанные с ними базовые версии.



Рис. 6-4. Стадии физического дизайна

Далее перечислены основные задачи четырех стадий физического дизайна.

- **Исследование** — на этой стадии команда определяет:
 - физические ограничения и требования;
 - все инфраструктурные изменения или особенности.
- **Анализ** — команда выполняет следующие задачи:
 - разрабатывает предварительную модель развертывания;
 - выбирает технологии, которые предполагается применять для разработки решения.
- **Рационализация** — задачи этой стадии таковы:
 - определение стратегии развертывания и объединения сервисов в наборы;
 - объединение в наборы компонентов и сервисов;
 - распределение и развертывание компонентов в сетевой топологии.

- **Реализация** — на этой стадии команда определяет:
 - модель программирования;
 - интерфейсы, атрибуты и сервисы компонентов.

Примечание Базовая версия реализации становится базовой версией физического дизайна.

Стадия исследования

На этапе физического дизайна команда фокусируется на создании технического решения, основанного на логическом дизайне. Чтобы создать подобный продукт, команда должна учесть ограничения, в том числе налагаемые архитектурой, бизнес-процессами и инфраструктурой предприятия. Кроме того, команде необходимо проанализировать требования к архитектуре и работе решения, в том числе безопасность, доступность, масштабируемость и управляемость будущего продукта. Например, система должна обрабатывать определенное количество транзакций в секунду. На стадии исследования команда выявляет эти ограничения и требования.

Результаты стадии исследования

Результаты стадии исследования описывают существующую инфраструктуру бизнеса и ложатся в основу анализа, рационализации и спецификации. В результатах базовой версии исследования описывается:

- существующая сетевая инфраструктура;
- существующая топология данных;
- существующая топология компонентов;
- физические прикладные требования;
- обновленные оценки рисков и планы по их предотвращению.

Определение физических требований и ограничений

На протяжении всего проектирования собирается и анализируется информация о требованиях и ограничениях бизнеса. Во время физического дизайна основное внимание уделяется физическим требованиям и ограничениям, оказывающим влияние на разработку решения.

Источником физических требований является существующая бизнес-среда и архитектура предприятия. Вот типичные физические требования к решению:

- производительность;
- стоимость и выгода;
- простота использования;
- простота развертывания;
- простота поддержки;
- надежность;
- возможность повторного использования.

А типичные физические ограничения таковы:

- бюджет;
- календарный график;
- топология сети;
- топология данных;

- топология компонентов;
- особенности технологий;
- безопасность.

Разрешение конфликтов между требованиями и ограничениями

Часто требования и ограничения противоречат друг другу. Раннее выявление и устранение подобных конфликтов позволяет снизить вероятность возникновения проблем в дальнейшем. Если проблема все-таки возникает, план ее решения уже есть. Например, приложение может требовать пропускную способность сети в 100 Мбит/с, тогда как существующая сетевая инфраструктура поддерживает только 10 Мбит/с.

Такое противоречие обычно разрешают так.

- Определяют жизненно важные для проекта требования. Выявляют конфликты между требованиями, одновременно анализируя влияние каждого из ограничений. До начала разработки предлагают как можно больше альтернативных компромиссных решений.
- Выявляют места инфраструктуры, где требования противоречат ограничениям.
- Анализируют разрыв между требованиями и ограничениями и определяют, следует ли принимать какие-либо решения для разрешения конфликта. В описанной ситуации вы вправе принять решение модернизировать сеть, увеличив пропускную способность до 100 Мбит/с. Подобное решение следует принимать как можно раньше, чтобы не создать нереализуемое решение.
- Проводят мозговой штурм при участии всех групп, связанных с проектом, — представителей бизнеса, пользователей, проектной команды и группы разработчиков.

Предлагается несколько способов устранения расхождения между требованиями и ограничениями.

- **Оставить все как есть.** Такой подход предполагает, что расхождение допустимо в первоначальной версии продукта. Четко опишите последствия такого расхождения. Кроме того, все заинтересованные лица должны согласиться с допустимостью подобного расхождения.
- **Найти способ обойти противоречие.** Обходной вариант не всегда оказывается оптимальным в долгосрочной перспективе, однако иногда его не избежать, например, если ресурсы проекта ограничены.
- » **Отложить удовлетворение требования на более поздние стадии проекта.** Проектная команда может решить реализовать требование попозже и изменить ограничения, предоставив заказчику достаточные аргументы и указав последствия подобного шага.

Занятие 2. Стадия анализа

На этой стадии команда создает и совершенствует модели физического дизайна, используя документацию логического дизайна. Кроме того, как и на любом из этапов, команда уточняет артефакты, связанные как с дизайном, так и со спецификацией (UML-модели, требования и варианты использования системы), а также те, которые связаны с проектом (документы, посвященные различным видам риска, планы проекта, календарные графики и каталог действующих субъектов). Физический дизайн предусматривает выбор технологий-кандидатов, определенных на основании требований к приложению. После выбора возможных технологий команда создает предварительную модель развертывания.

Из этого занятия вы узнаете, как совершенствуются UML-модели, а также как выбрать технологии-кандидаты и создать предварительную модель развертывания.

Изучив материал этого занятия, вы сможете:

- ✓ усовершенствовать UML-модели логического дизайна;
- ✓ создать предварительную модель развертывания.

Продолжительность занятия — около 10 минут.

Совершенствование UML моделей

По завершении логического дизайна, команда располагает UML-моделями объектов, сервисов, атрибутов и отношений. Для управления сложными частями проекта, как правило, используются артефакты, которые лучше всего соответствуют намерениям и решениям команды, в том числе:

- перечень объектов и сервисов;
- диаграммы классов;
- диаграммы последовательности;
- диаграммы действий;
- диаграммы компонентов.

В процессе физического дизайна команда совершенствует эти модели.

Перечень объектов и сервисов

На стадии анализа изучите перечень сервисов, чтобы:

- распределить сервисы по категориям на основе MSF-модели приложений, основанных на сервисах:
 - пользовательские сервисы;
 - бизнес-сервисы;
 - сервисы данных;
 - системные сервисы;
- выявить сервисы, оставшиеся незамеченными.

Команда пытается определить сервисы, которые были пропущены в процессе логического дизайна, в том числе системные сервисы или особые технические сервисы для преобразования данных. Помните, что каждый скрытый сервис следует проверить на соответствие целям разработки, а затем — на предмет удовлетворения требований.

Диаграммы классов

Диаграммы классов применяются в логическом дизайне для отображения статической структуры модели объектов приложения. В процессе физического дизайна команда предпринимает следующие действия по уточнению и совершенствованию диаграмм классов:

- преобразует логические объекты в определения классов вместе с их интерфейсами;
- определяет объекты, не выявленные в процессе логического дизайна, такие, как объекты, основанные на сервисах (их также называют *общими сервисами*);
- консолидируют логические объекты (при необходимости);
- распределяют объекты по категориям согласно модели сервисов:
 - **объекты логических границ** — это потенциальные пользовательские сервисы;
 - **объекты логического управления** — это потенциальные бизнес-сервисы;
 - **объекты логических объектов-сущностей** — это потенциальные сервисы данных.

Возможны исключения из этих правил, поэтому, прежде чем принять решение, команда должна тщательно изучить все обстоятельства.

- Уточнить методы (в первую очередь их параметры), рассмотреть возможность использования перегрузки, объединить или разделить методы и определить способы обработки переданных значений.
- Уточнить атрибуты. Следует свести до минимума количество открытых (public) атрибутов в приложениях, основанных на серверной архитектуре без сохранения состояния. Во время физического дизайна команда работает в основном со *внутренними защищенными* (internal protected) атрибутами, которые используются объектами-наследниками.

На рис. 6-5 показана диаграмма классов для компонентов Order (Заказ).

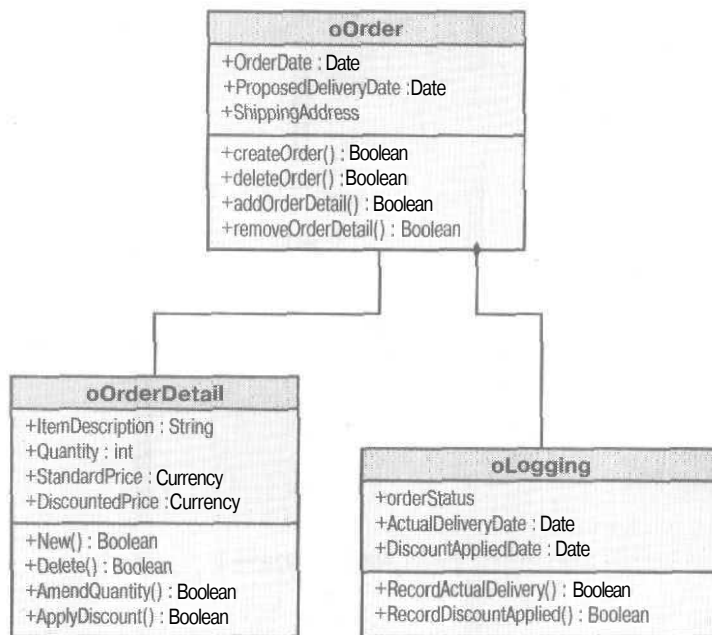


Рис. 6-5. Диаграмма классов

Диаграммы последовательностей

Они представляют взаимодействие между объектами и динамику объектной модели и используются, как правило, для наглядного отображения сложных отношений между классами, в которых бывает непросто разобраться, изучая одни лишь статические методы и атрибуты классов. В процессе физического дизайна команда:

- обновляет классы на основе усовершенствованных классов физического дизайна;
- совершенствует диаграммы последовательностей с включением взаимодействий между классами или сервисами и с учетом физических ограничений или технологических требований;
- определяет дополнительные сообщения (методы), инициируемые новыми физическими классами.

На рис. 6-6 показана диаграмма последовательностей для объектов Product (товар) и Catalog (каталог).

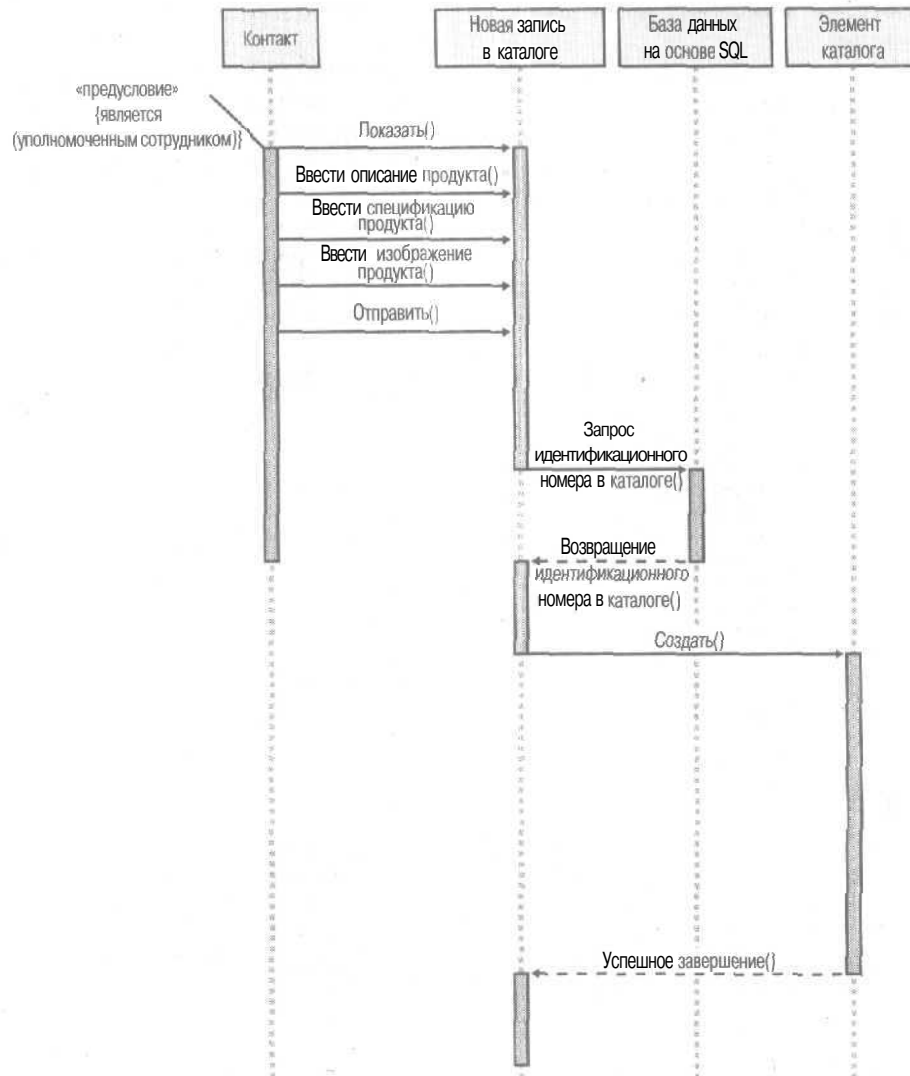


Рис. 6-6. Диаграмма последовательности

Диаграммы действий

Диаграммы действий (activity diagrams) используются для представления переходов между состояниями и последовательности работы приложения. Диаграммы действий можно использовать вместо диаграмм последовательностей и наоборот. В процессе физического дизайна проектная команда совершенствует диаграммы действий, учитывая в них требования, связанные с особенностями физической платформы и технологий, а также определяя возможные последовательности рабочих операций (workflow).

Диаграммы компонентов

Они необходимы для отображения зависимостей между компонентами и их наборами. Как и к диаграммам последовательностей и действий, к ним прибегают в основном в более сложных ситуациях. В процессе физического дизайна диаграммы компонентов создаются с целью:

- более наглядно представить зависимости между компонентами;
- уточнить решения, касающиеся объединения компонентов в наборы.

На рис. 6-7 показана диаграмма компонентов для процесса заказа товара.

Для работы с результатами анализа с применением OLAP-кубов, полученных с SQL-сервера, на стороне клиента применяется Excel. Кубы могут располагаться на стороне как клиента, так и сервера



Рис. 6-7. Диаграмма компонентов

Создание предварительной модели развертывания

На основе таких артефактов, как архитектура приложения, структура команды, календарный график проекта, требования, документ оценки рисков и технологии-кандидаты, команда создает *предварительную модель развертывания*. Она содержит топологию сети, данных и компонентов, а также позволяет команде проекта и другим заинтересованным лицам оценить общую картину дизайна. Как уже говорилось, в процессе физического дизайна команда предлагает варианты топологии системы, которые еще не утверждены окончательно.

Топология сети

Топология сети (рис. 6-8) — это инфраструктурная карта, показывающая расположение и схему подключения оборудования. Здесь показаны рабочие станции и серверы и описаны их функции. Кроме того, топология отображает инфраструктуру сети, объединяющую компьютеры.

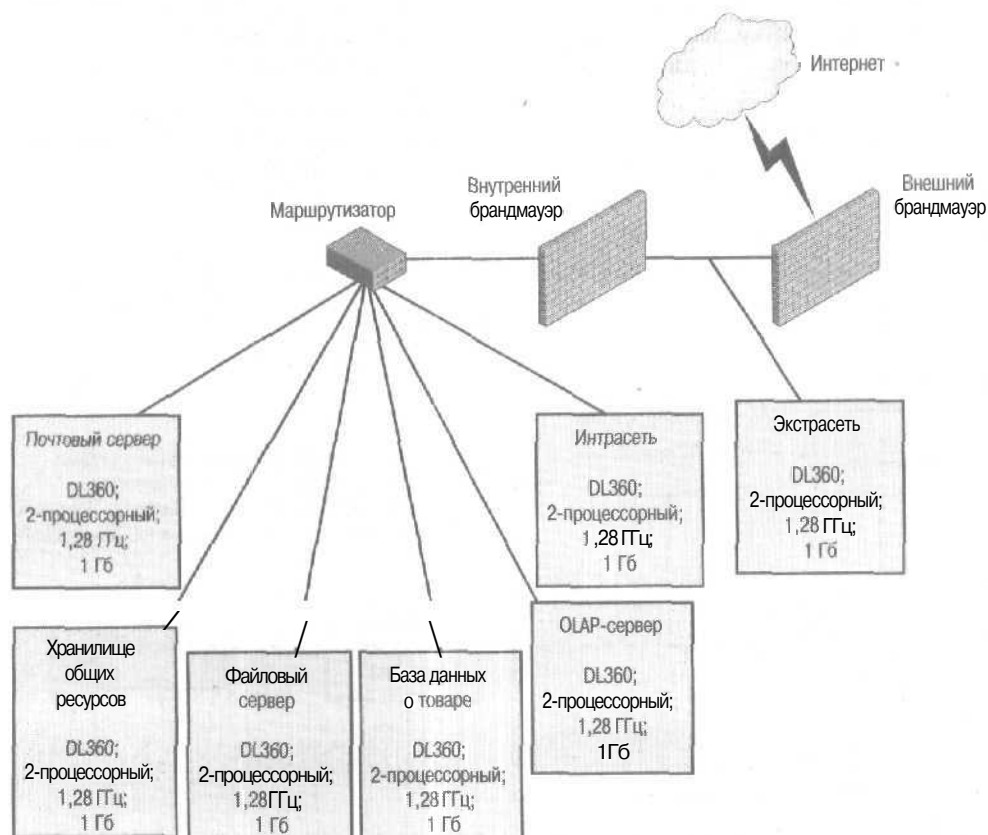


Рис. 6-8. Топология сети

Примечание Здесь показана неполная топология сети. Полная версия сетевой топологии компании Adventure Works Cycles вы найдете в документе *AWC - Network.vsd* в папке `\SolutionDocuments\Chapter06` на прилагаемом к книге компакт-диске.

Занятие 2

и данных

Топология компонентов и данных — это схема, на которой показано расположение и их сервисов с учетом привязки к топологии сети, а также распределение по различным уровням сервисов. Если рассматривается абсолютно новый проект, должна существовать вертикальная структура, отражающая текущее состояние предприятия. На этом этапе в проект вводятся новые компоненты и сервисы, необходимые для решения задачи. Приводится пример топологии компонентов и данных в компании Adventure Works Cycles.

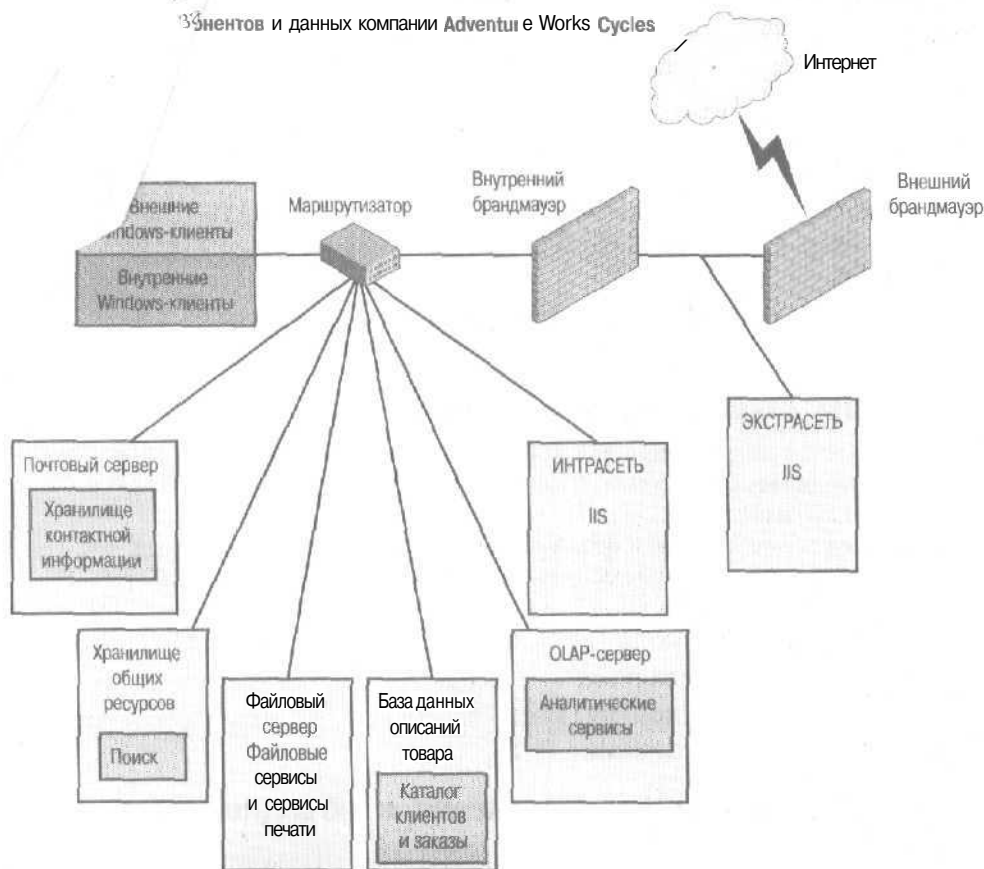


Рис. 6-9. Топология компонентов и данных

Примечание Здесь показана неполная топология сети. Полную версию сетевой топологии компании Adventure Works Cycles вы найдете в документе *AWC Component Topology.vsd* в папке *\Solution Documents\Chapter06* на прилагаемом к книге компакт-диске.

Занятие 3. Стадия рационализации

На этой стадии проектная команда проектирует основанные на компонентах архитектуры приложения и распределения сервисов по сервисным уровням.

Здесь описываются результаты стадии рационализации, а также работы стратегии распределения и объединения спроектированных наборов. Кроме того, вы узнаете, как способы обеспечения взаимосвязи влияют на порядок объединения сервисов в наборы. Вы также проверите и усовершенствуете топологию компонентов.

Изучив материал этого занятия, вы сможете:

- ✓ определить результаты базовой версии стадии рационализации;
- ✓ создать стратегии распределения и объединения;
- ✓ описать связность и сочленение;
- ✓ преобразовать сервисы в компоненты;
- ✓ распределить предварительные варианты компонентов по уровням;
- ✓ создать модель развертывания;
- ✓ проверить и усовершенствовать распределение и объединение в наборы.

Продолжительность занятия — около 10 минут.

Результаты стадии рационализации

В результатах стадии рационализации описываются технологии, стратегии и топологии создаваемой системы. Вот некоторые результаты базовой версии стадии рационализации:

- стратегия распределения и объединения в наборы;
- предварительные варианты основанных на сервисах компонентов;
- модели развертывания:
 - будущая топология сети;
 - будущая топология данных;
 - будущая топология компонентов;
- базовая модель развертывания.

Создание стратегии распределения и объединения в наборы

Стадия рационализации предусматривает итерационный подход, применяемый проектной командой для оптимизации проекта. Одна из задач стадии — распределение сервисов и их объединение в компонентах.

Стратегия распределения — это обоснование выбранного расположения отдельных сервисов в архитектуре решения. Распределение выполняется на основе сервисов, а не компонентов.

Стратегия объединения в наборы — это обоснование распределения сервисов по компонентам. В рамках одной системы может быть несколько стратегий. Например, общепринято разнесение бизнес-сервисов по бизнес-объектам

(обычно это называют «бизнес-фасадом») без включения бизнес-правил непосредственно в интерфейсы классов. Подобный подход позволяет команде создавать уровень бизнес-правил, содержащий большинство бизнес-правил (например, применение авторизации для ограничения возможностей по предоставлению скидки). Последующие изменения и обновления уровня бизнес-правил не затрагивают бизнес-объекты. Этот уровень позволяет свести все изменения *лишь* к модификации интерфейсов взаимодействия с этими объектами. Стратегия обычно продумывается заранее.

В процессе определения подходящей общей стратегии распределения и объединения в наборы необходимо учесть особенности управления состоянием и производительность продукта.

Управление состоянием

Управление состоянием (state management) — это процесс, с помощью которого система поддерживает состояния и информацию Web-страниц между обращениями к серверу. Далее перечислены некоторые методы управления состоянием в Web-решениях.

- **Хранение информации о состоянии на стороне клиента.** Для хранения этой информации используются объекты, которые содержат данные о состоянии в промежутках между обращениями к серверу.
- **Строки SQL-запросов** — это информация, которая добавляется в конец URL-адреса страницы. Например, номер товара и идентификатор категории.
- **Cookie-файлы** — это небольшие фрагменты данных, которые хранятся в текстовом файле в файловой системе клиента или в памяти, выделенной для сеанса клиентского браузера.
- **Состояние приложения** — это глобальный механизм хранения данных, доступный из всех страниц Web-приложения. Он применяется для хранения информации между обращениями к серверу и переходами между страницами.
- **Состояние сеанса** похоже на состояние приложения, но ограничивается текущим сеансом браузера. У различных пользователей, работающих с системой, состояния сеанса разнятся. Кроме того, пользователю, прервавшему работу и возобновившему ее позже, назначается новое состояние сеанса.

Проектные требования

- **Масштабируемость** — это возможность быстро и легко модернизировать систему, чтобы она могла обслуживать больше операций или пользователей.
- **Производительность** — мера скорости отклика системы и скорость, с которой она справляется с прикладными задачами.
- **Управляемость** — простота и легкость управления системой на любом уровне.
- **Повторное использование** — мера возможности использовать уже созданные в других приложениях компоненты.
- **Бизнес-контекст** указывает на круг бизнес-подразделений, например важных для бухгалтерии или отдела продаж.
- **Глубина детализации** — количество сервисов и объектов, объединенных в одном компоненте.

Определяя стратегию распределения и объединения в наборы сервисов бизнес-решения, команда должна учитывать физические требования и ограничения создаваемой системы.

При использовании нескольких стратегий следует соблюдать разумный баланс между требованиями и ограничениями для создаваемого продукта. Например, при выборе стратегии, ориентированной на высокую производительность, пострадает масштабируемость.

Связность и сочленение

Одной из особенностей хорошего плана компонентов является сильная связность и свободное сочленение (*loose coupling*). *Связность* (*cohesion*) — это степень связи между различными внутренними элементами компонента, а *сочленение* (*coupling*) — между разными компонентами.

Связность

Компонент отличается сильной связностью, если его сервисы тесно взаимодействуют между собой. Надежность компонента напрямую зависит от глубины связей между его сервисами. Связность может быть как полезной, так и нежелательной — все определяется ее природой.

- **Функциональная связность.** Компонент выполняет только одну задачу. Это наиболее сильный тип связности.
- **Последовательная связность.** Компонент объединяет операции, выполняемые в жестко определенном порядке. Причем во всей цепочке совместно используются одни данные.
- **Коммуникативная связность.** В операциях одного компонента используются одни данные, но сами операции не связаны между собой. Этот тип связности позволяет свести до минимума издержки на обеспечение связи в приложении.
- **Временная связность.** Операции объединяются на основании того, что выполняются одновременно.

Не всякая связность полезна. Перечисленные далее типы связности часто нарушают организацию приложения и затрудняют его понимание, отладку и внесение изменений.

- **Процедурная связность.** Операции группируются на основании выполнения в определенном порядке, но в отличие от последовательной связности, в них не используются одни данные.
- **Случайная связность.** Операции группируются, несмотря на отсутствие разумно обоснованной связи между ними.

Сочленение

Сочленение бывает сильным или свободным. При выполнении своих функций сильно связанный компонент значительно зависит от внешних компонентов, а свободно связанный компонент вообще не зависит или слабо зависит от внешних компонентов.

Как правило, чем слабее связь между компонентами, тем проще работать с отдельными компонентами, не боясь серьезно нарушить работу приложения. Компоненты должны как можно меньше зависеть друг от друга. Но если зависимость все-таки существует, она должна быть максимально ясной и понятной — это существенно облегчает определение интерфейсов. Есть и другая причина требования четкого описания зависимостей в проекте — это позволяет снизить риск появления цепных ошибок.

Объединение компонентов в наборы

Основная цель стадии рационализации — распределение сервисов по уровням и объединение их в наборы. На первом шаге этого процесса сервисы распределяются по уровням: пользовательскому, бизнес-сервисам и сервисам данных.

Распределение начинают с определения высокоуровневых сервисов в физической модели объектов и разбиения их на отдельные сервисные уровни.

Затем для каждого бизнес-объекта оставшиеся низкоуровневые сервисы группируются по трем уровням: пользовательским сервисам, бизнес-сервисам и сервисам данных (рис. 6-10).

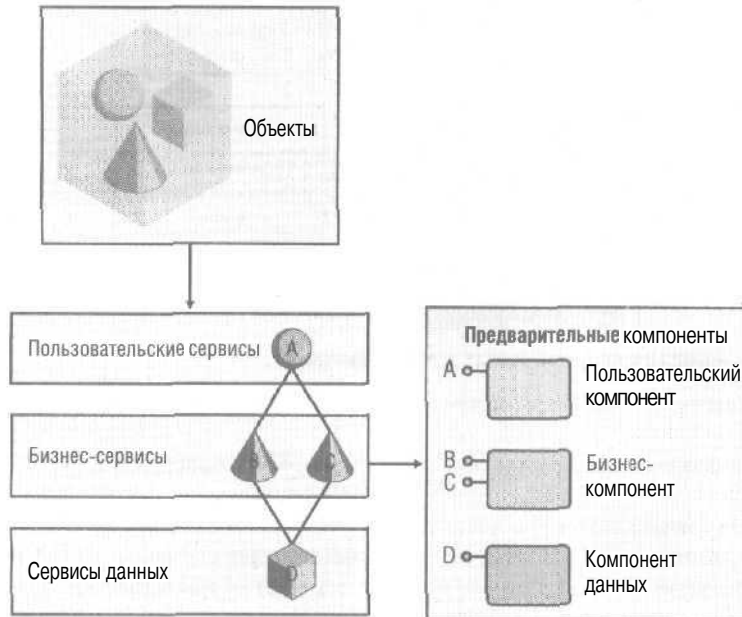


Рис. 6-10. Три уровня сервисов

Распределение предварительных компонентов

После объединения сервисов в компоненты определяется положение последних в топологии сети и создается топология компонентов. Чтобы начать процесс распределения, команда определяет категории сервисов — пользовательские, бизнес-сервисы и сервисы данных — для каждого узла топологии сети. Эти категории служат основой для распределения. Стратегия распределения развивается по мере проверки дизайна на основании требований к решению.

Рис. 6-11 иллюстрирует три уровня сервисов и соответствующих компонентов в простом приложении.

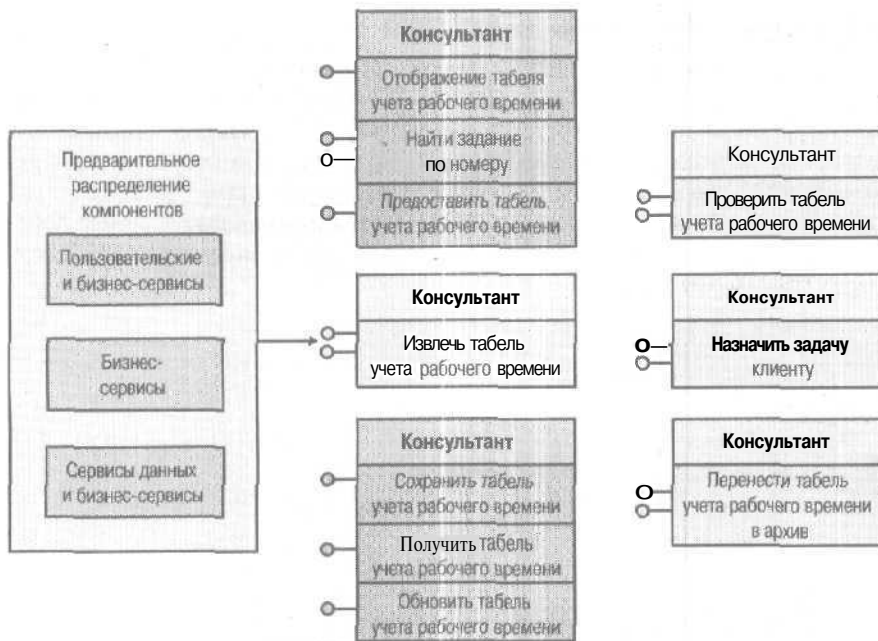


Рис. 6-11. Распределение предварительных компонентов

Задача распределения по уровням облегчается, если следовать нескольким рекомендациям.

- Пользовательские сервисы желательно размещать на Web-серверах и клиентских компьютерах.
- Бизнес-сервисы размещают на серверах приложений или Web-серверах.
- Сервисы данных располагают в местах размещения данных согласно топологии, в том числе на серверах баз данных или в иных местах хранения информации.
- После определения мест расположения уровней сервисов предварительные компоненты размещаются на соответствующих уровнях сервисов. В результате создается исходная топология компонентов, которая будет развиваться и уточняться во время всего процесса рационализации.

Рис. 6-12 иллюстрирует логическое деление на три уровня сервиса в компонентной модели типовой системы обработки заказов.

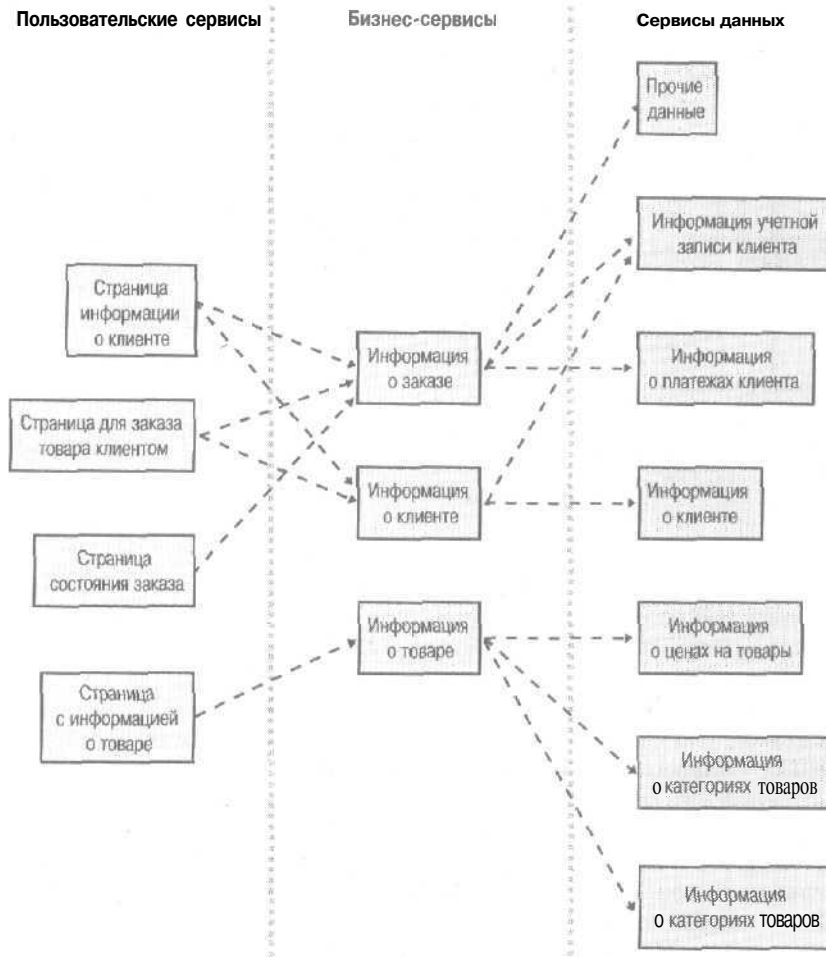


Рис. 6-12. Пример компонентной модели

Создание модели развертывания

Эта модель показывает место приложения и его сервисов в реальной топологии серверов. Модель развертывания необходима группам разработчиков и управления выпуском для проектирования и планирования топологии и конфигурации серверов.

На рис. 6-13 показан пример модели развертывания.



Рис. 6-13. Пример модели развертывания

Существует пять наборов серверов;

- серверы приложений, на которых располагаются Web-приложения и компоненты;
- серверы баз данных, на которых размещаются базы данных;
- серверы порталов, управляющие всеми функциями порталов, в том числе обработкой HTTP-запросов от серверов приложений;
- серверы каталогов или безопасности, которые управляют данными по безопасности, например подробными данными о сотрудниках;
- серверы интеграции приложений и данных, которые поддерживают связь с дополнительными подсистемами, например Microsoft BizTalk Server или Microsoft Host Integration Server.

Проверка и уточнение распределения и объединения в наборы

После создания моделей развертывания и компонентов проектная команда проверяет и уточняет их.

Проверка моделей компонентов и развертывания

В процессе физического дизайна команда должна проверять топологию компонентов на предмет соответствия стратегии и требованиям. Проверка осуществляется постоянно, что позволяет команде выполнять проектирование итеративно.

Оптимальное решение создается за счет многократного прохождения цикла проверки и тестирования, а также использования прототипов для тестирования и корректировки распределения и объединения в наборы компонентов. Итерации завершаются, когда все значительные улучшения уже внесены и остаются мелкие, «косметические» корректировки.

Уточнение моделей компонентов и развертывания

Ключ к уточнению компонентов — работа с сервисами, а не с компонентами. Чтобы улучшить топологию компонентов, откажитесь от предварительного объединения компонентов в наборы и перераспределите сервисы так, чтобы они удовлетворяли потребностям продукта.

Например, если требование гласит, что пользователям следует предоставить возможность «прокручивать» полный список товаров без задержек, для оптимальной производительности можно перенести некоторые сервисы данных на клиент.

После того как сервисы перераспределены, необходимо заново объединить их в компоненты. Сервисы объединяются в соответствии с определенными ранее стратегиями. Например, если главная цель — простота развертывания, проектная команда может предпочесть разместить на клиентском компьютере один крупный компонент вместо множества мелких.

Совет Обязательно учитывайте особенности связности и сочленения при повторном объединении и распределении. Высокая связность и свободное сочленение — это идеал, к которому следует стремиться, хотя он и не всегда достижим на практике из-за особых требований к продукту.

Занятие 4. Реализация физического дизайна

На стадии реализации команда определяет модель программирования, используемую группой разработчиков, интерфейсы и внутреннюю структуру каждого компонента. Результатами стадии реализации являются модель программирования, спецификации интерфейсов компонентов, атрибуты и сервисы, а также базовая спецификация компонентов. Степень детализации каждого из перечисленных результатов зависит от уровня взаимодействия между менеджерами и группой разработчиков.

На этом занятии рассказывается, как реализуется решение, а также о модели программирования. Вы научитесь определять интерфейсы и внутреннюю структуру компонентов и узнаете, как создается дизайн пользовательского интерфейса и модели баз данных.

Изучив материал этого занятия, вы сможете:

- ✓ описать модели программирования;
- ✓ рассказать о принципах проектирования интерфейсов компонентов;
- ✓ создать внутреннюю структуру компонентов;
- ✓ описать физическую модель пользовательского интерфейса;
- ✓ описать модель базы данных.

Продолжительность занятия — около 10 минут.

Что такое модель программирования

Физический дизайн представляет решение с точки зрения разработчика, поэтому проектная команда должна предоставить разработчикам спецификации компонентов и перечень технологий, выбранных для создания продукта. В спецификациях компонентов достаточно информации для программистов, чтобы они могли приступить к разработке компонентов решения. Спецификация описывает интерфейсы и область действия сервисов и атрибутов компонентов. Она напрямую связана с выбранной моделью программирования.

В модели программирования описывается структура компонентов, причем она основывается на целях, для которых создается продукт, и на технологиях, выбранных для реализации.

Назначение модели программирования

Модель программирования описывает, каким образом команда разработки должна применять выбранные технологии, и состоит из спецификаций программирования, или стандартов, которым следуют во время построения продукта. Модель программирования устанавливает принципы, обеспечивающие единый подход к реализации компонентов и расширяющие возможности по их поддержке.

Стандарты, предписываемые моделью программирования, могут отличаться в различных частях и на разных уровнях сервисов приложения. Стандарты физического дизайна — это, по сути, реализация принципов, заложенных в архитектуре. Если команда решает реализовать сервер без поддержки состояний и клиента, поддерживающего полную информацию о состоянии, соответствующие атрибуты удаляются из определений классов именно на стадии физического дизайна.

В качестве основного метода пересылки наборов данных клиенту, поддерживающему состоянию, можно задействовать отсоединенные наборы записей .NET.

Проектные особенности модели программирования

- **Технологии реализации** — это языки программирования, интерфейсы прикладного программирования (API), серверы, серверные и прочие технологии, необходимые для реализации приложения. Чтобы эффективно использовать эти технологии, необходимо определиться с моделью программирования. Например, COM+ 2.0 требует применения **однопоточных** (single-threaded) внутрипроцессных (in-process) компонентов без возможности повторного вхождения (single-entrant).
- **Объекты с и без поддержки состояний.** Поддержка состояний может оказать прямое влияние на производительность, масштабируемость и сложность реализации компонента. В отличие от объектов без поддержки состояний **объекты, поддерживающие состояния**, хранят определенную информацию о взаимодействии с клиентом между вызовами. Создавая объект с поддержкой состояний, особенно в **Web-приложениях**, необходимо заранее решить, где и как будет храниться информация о состоянии. С другой стороны, в объектах без поддержки состояний вся необходимая информация отправляется и принимается в момент вызова или при завершении транзакции.
- **Внутрипроцессные и внепроцессные вызовы функций.** Внутрипроцессные компоненты работают в одном процессе, за счет чего удается избежать издержек, связанных с **маршалингом**, и повысить производительность приложения. Внепроцессные компоненты работают в отдельном от вызывающего клиента процессе, поэтому в них приходится прибегать к **маршалингу**. Быстродействие приложения при этом снижается.
- **Связность и сочленение.** Решая вопрос о том, как применить эти принципы в модели программирования, следует отдавать предпочтение сильно связным компонентам и свободно сочлененным объектам. Методы и свойства **объекта** с сильной связностью тесно соединены; объекты этого компонента также отличаются значительная взаимосвязь. **Свободно** сочлененные объекты не так сильно зависят от интерфейса или состояния других объектов. Взаимодействие между слабо связанными объектами **реализуется** посредством сообщений.
- **Модели с сохранением и без сохранения подключения.** Для нормальной работы в распределенной среде компоненты, участвующие в предоставлении сервиса, должны поддерживать постоянные подключения в режиме реального времени. В случае разрыва этих соединений происходит сбой взаимодействующих компонентов. Поскольку компоненты **реального времени**, как правило, должны работать в режиме с поддержкой постоянного подключения, от компонентов, предназначенных для работы в среде без поддержки постоянных подключений, требуется способность восстанавливать подключение **при необходимости**.
- **Синхронная и асинхронная модель программирования.** В синхронной модели программирования вызывающий компонент блокируется, пока **вызванный** интерфейс не закончит обработку запроса и не вернет управление. Асинхронная модель программирования позволяет компонентам отправлять сообщения другим **компонентам** и продолжать работу, не ожидая немедленного ответа. Компоненты, рассчитанные на асинхронные взаимодействия, сложнее в реализации, хотя есть технологии, значительно упрощающие их про-

граммирование, например COM+ Queued Components. Компонент, поддерживающий асинхронную модель, легче масштабировать, поскольку отдельные компоненты не блокируются и им не приходится ждать, пока другой процесс завершит свою работу.

- **Потоковые модели.** Выбор потоковой модели для компонента — задача не из легких, поскольку она определяется функциями, выполняемыми компонентом. Компонент, занятый интенсивными операциями ввода-вывода, скорее всего должен поддерживать модель свободных потоков (free threading) — это позволяет ему быстро реагировать на запросы клиентов, поддерживая вызовы интерфейсов во время *задержек*, связанных с вводом-выводом. Объекты, взаимодействующие с пользователем, обычно основаны на отделенных потоках (apartment threading) для синхронизации входящих **COM-вызовов** с операциями с окнами.
- **Обработка ошибок.** Поскольку не бывает «идеальных» компонентов и среды, компоненты должны поддерживать обработку ошибок. Некоторые модели программирования и развертывания ограничивают перечень доступных методов обработки ошибок. Например, очень трудно получить доступ к сообщению, записанному в журнал на клиенте, и отправить тому, кто сможет разобраться с возникшими проблемами.
- **Безопасность.** Существует четыре основных способа обеспечения безопасности компонентов и сервисов:
 - безопасность компонента обеспечивается на уровне методов, интерфейсов или целого компонента;
 - защита базы данных вступает в действие, когда имеют место операции с данными;
 - безопасность пользовательского контекста реализуется с применением системных механизмов безопасности или «защитой» в приложение защиты;
 - механизм безопасности, основанный на ролях, предусматривает создание групп.
- **Распределение.** Тщательно продумайте метод распределения приложения. Помните о том, что три логических уровня не обязательно физически разделены. Например, некоторые **бизнес-сервисы** лучше всего разместить на клиенте. Поэтому рекомендуется предусмотреть именно столько физических уровней, сколько требуется вашему приложению в соответствии с целями бизнеса. Иногда все компоненты собраны в одном месте.

Знание и опыт технических специалистов не являются составной частью модели программирования, которую они реализуют, тем не менее это очень важный момент, который необходимо принимать во внимание.

Примечание Как правило, не удастся «втиснуть» все компоненты в одну модель программирования. В приложении обычно используется множество моделей программирования в зависимости от требований к конкретным компонентам.

Определение интерфейсов компонента

После описания программной модели проектная команда определяет порядок взаимодействия компонентов между собой. Взаимодействия документируются как интерфейсы компонентов с описанием порядка получения доступа к сервисам и атрибутам этих интерфейсов. Интерфейс представляет один или более

сервисов и является механизмом запроса сервиса компонента для выполнения операции, а также способом получения информации об атрибутах. Внешние по отношению к компоненту структуры также содержатся в интерфейсе компонента. Интерфейс компонента:

- представляет связь вида «поставщик — потребитель» между компонентами;
- является средством доступа к сервисам компонента;
- представляет набор связанных методов, представляющих собой часть сервиса;
- содержит атрибуты объектов, содержащихся в компоненте с поддержкой состояния.

Спецификация интерфейса компонента, как правило, описывает все способы доступа к компоненту и примеры того, как получить доступ каждым из этих способов. Работа над спецификацией завершается только тогда, когда команда разработки полностью создаст компонент.

Создавая интерфейсы компонента, следует помнить, что:

- опубликованный интерфейс считается неизменным;
- измененный существующий интерфейс должен публиковаться как новый компонент или новый интерфейс;
- типы данных опубликованных интерфейсов должны поддерживаться потребителями их сервисов.

Существует только один способ получить доступ к сервисам компонента — использовать опубликованный интерфейс компонента. Недостаточно качественно определенный интерфейс может нарушить нормальную работу других частей продукта.

В большинстве языков программирования разработчикам доступен язык IDL (Interface Definition Language — язык определения интерфейсов), который применяется для определения интерфейса в процессе кодирования компонента. Подобное определение позволяет гарантировать целостность и управление изменениями, когда над созданием компонента работает несколько программистов.

Все применяемые при разработке языки разнятся синтаксисом и сложностью определения интерфейсов компонентов. Языки наподобие Microsoft Visual Basic скрывают большинство сложностей интерфейса от разработчиков, а Microsoft Visual C++ предоставляет массу возможностей контроля и доступа к интерфейсам, но за эти возможности приходится расплачиваться — программировать на C++ намного сложнее. Как уже говорилось, физический дизайн не зависит от выбранного языка реализации. В процессе физического дизайна создается полностью законченная версия определения сигнатур в псевдокоде, IDL или на выбранном языке программирования, а на этапе разработки эти спецификации превращаются в конкретную реализацию с применением выбранных инструментов, языков и технологий.

Физический дизайн модели пользовательского интерфейса

Уровень представления дает возможность пользователям работать с системой, посредством механизма взаимодействия между пользователем и уровнем бизнес-сервисов. Существует два типа пользователей: люди, которым требуется интерфейс, через который они работают с системой, и другие компьютерные системы. Хотя последним и не нужен пользовательский интерфейс, им все равно необходим «посредник». Как правило, таким посредником становится уровень

пользовательских сервисов или интерфейс Web-сервисов, который обеспечивает взаимодействие этими двумя уровнями.

Примечание Пользовательский интерфейс — это инструмент визуального взаимодействия пользователя с системой. Пользовательские сервисы поддерживают навигацию, проверку и логику обработки ошибок.

В процессе проектирования пользовательского интерфейса и уровня пользовательских сервисов, проектной команде требуются результаты концептуального дизайна, в том числе:

- требования и ограничения решения;
- сценарии использования будущей системы;
- модели потоков работ;
- профили пользователей;
- описания задач;
- пользовательская терминология и концепции.

Примечание Подробнее о дизайне пользовательского интерфейса рассказывается в главе 7.

Физический дизайн модели базы данных

В процессе логического дизайна команда, ответственная за создание базы данных, анализирует всевозможные способы структурирования информационных потребностей, определенных на этапе концептуального дизайна, в виде логической модели базы данных.

В процессе физического дизайна, команда базы данных должна учесть:

- физические ограничения базы данных, такие, как размеры памяти и дискового пространства;
- особенности настройки производительности, такие, как механизмы выявления взаимоблокировок (deadlock), индексирование и узкие места;
- основные и внешние ключи;
- триггеры;
- строение хранимых процедур;
- особенности объектной модели приложения;
- спецификации индексирования;
- разбиение данных на разделы (partitioning);
- перенос данных из существующих систем баз данных;
- процедурные особенности, в том числе механизм перехода при сбое в кластере, резервное копирование и обновление данных.

Примечание Подробнее о дизайне базы данных — в главе 8.

Практикум. Создание физического дизайна



При выполнении этого практикума вам придется использовать полученные знания для создания модели классов и диаграммы модели компонентов.

Упражнение 1. Создание модели классов

Откройте файл *C06Ex1.vsd* из папки `\SolutionDocuments\Chapter06\` с прилагаемого к книге компакт-диске и изучите логический дизайн. На его основании спроектируйте физическую модель классов для объектов *Catalog*, *Catalog Item*, *ManageCatalogItem* и *Search*. Вы вправе удалять или добавлять объекты и их атрибуты и операции. Один из возможных ответов к этому упражнению вы найдете в файле *C06Ex1_Answer.vsd*.

Ниже приводятся некоторые из выкладок и решений, содержащихся в файле *C06Ex1_Answer.vsd*.

Функции логического объекта *ManageCatalogItem* лучше всего реализовать с применением операций объекта *Catalog*: *addCatalogItem*, *removeCatalogItem*, *countSelectedItems* и других. Удаляем этот объект из диаграммы.

Клиенту необходима возможность выбирать товары для покупки из каталога, не прекращая его просмотра. Группа выбранных товаров может много раз меняться до окончательного формирования и отправки заказа. Чтобы удовлетворить это требование, добавлен новый объект: *ShoppingCart* (корзина).

Упражнение 2. Создание диаграммы модели компонентов

Откройте файл *C06Ex2.vsd* (папка `\SolutionDocuments\Chapter06\`) и перейдите на вкладку *Component Model*. Эта страница пуста. Создайте модель компонентов для объектов, определенных в упражнении 1. Считайте, что:

- торговые представители хранят контактную информацию о клиентах в Microsoft Outlook;
- информация о Web-клиентах содержится в базе данных Microsoft SQL Server и управляется при помощи компонента авторизации.

Один из возможных ответов к этому упражнению вы найдете в файле *C06Ex2_Answer.vsd* в той же папке.

Резюме

- В процессе физического дизайна проектная команда определяет сервисы и технологии, поддерживаемые будущей системой.
- Физический дизайн — это процесс описания компонентов, сервисов и технологий решения с точки зрения разработчиков (программистов).
- Цели физического дизайна:
 - превратить логический дизайн в спецификации набора компонентов;
 - создать базовую версию для реализации дизайна;
 - определить технологии, оптимальные для программирования;
 - создать структурное представление решения с точки зрения группы разработчиков.
- Физический дизайн состоит из четырех стадий: исследования, анализа, рационализации и реализации.
- Физический дизайн дает новые результаты, в том числе:
 - диаграммы классов;
 - диаграммы последовательностей;
 - базовую модель развертывания;
 - модель программирования;
 - спецификации компонентов.
- К базовым результатам стадии исследования относятся:
 - существующая топология сети;
 - существующая топология данных;
 - существующая топология компонентов;
 - физические требования к приложению;
 - оценка рисков и план их предотвращения.
- Во время стадии исследования физического дизайна проектная команда определяет требования и ограничения продукта и пытается сократить разрыв между ними.
- В процессе анализа проектная команда совершенствует созданную в процессе логического дизайна UML-модель, включающую реестр объектов и сервисов, Диаграммы классов, диаграммы последовательностей и диаграммы действий.
- На стадии анализа проектная команда создает предварительную модель развертывания, в том числе топологии сети, данных и компонентов.
- Топология сети — это инфраструктурная карта, отражающая расположение оборудования и его подключение.
- Топология данных — это карта, показывающая расположение хранилищ данных в общей топологии сети.
- Топология компонентов — схема, отражающая расположение компонентов и их сервисов в общей топологии сети.
- Результаты стадии рационализации таковы:
 - стратегия распределения и объединения в наборы;
 - предварительный вариант основанных на сервисах компонентов;
 - модели развертывания;
 - базовая модель развертывания.

- Стратегия распределения — это логическое обоснование расположения каждого сервиса в архитектуре решения. Распределение основано на сервисах, а не на компонентах.
- Стратегия объединения в наборы — это логическое обоснование включения сервиса в определенный компонент. В рамках одного решения может быть несколько стратегий.
- Связность — это степень связи между различными внутренними элементами компонента.
- Сочленение — это степень связи компонента с другими компонентами.
- Для распределения сервисов выявите сервисы высокого уровня в модели бизнес-объектов и разбейте их на отдельные, относящиеся к определенным уровням сервисы.
- Для объединения сервисов в компоненты группируют низкоуровневые сервисы в три компонента: пользовательские сервисы, бизнес-сервисы и сервисы данных.
- Модель развертывания связывает приложение и его сервисы с реальной топологией серверов.
- Проектная команда должна проверить топологию компонентов на предмет соответствия стратегиям и требованиям создаваемого продукта.
- Для уточнения модели компонентов и развертывания команда должна отказаться от предварительного объединения компонентов и заново распределить сервисы в соответствии с требованиями.
- Во время стадии реализации команда выбирает модель программирования, интерфейс и внутреннюю структуру каждого компонента.
- Модель программирования описывает, как структурируются компоненты; это выполняется на основании целей решения и реализуемых технологий.
- Интерфейс компонента описывает, как получить доступ к его сервисам и атрибутам.
- Уровень представления (презентационный уровень) позволяет пользователям взаимодействовать с бизнес-системой и предоставляет механизм «общения» между пользователем и бизнес-сервисами системы.

Закрепление материала



л 1 Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Каковы цели физического дизайна?
2. Чем отличаются концептуальный, логический и физический дизайн?
3. Чем занимается группа разработчиков (программистов) в процессе физического дизайна?
4. Что включает в себя модель развертывания?
5. Чем занимается проектная команда на стадии исследования?
6. Как проектная команда сокращает разрыв между требованиями и ограничениями?
7. Как на стадии анализа проектная команда использует список объектов и сервисов, созданных в процессе логического дизайна?
8. Как проектная команда уточняет диаграмму классов на стадии анализа?
9. Как вы выбираете технологии-кандидаты для решения?
10. В чем различие между топологией сети и топологией данных модели развертывания?
11. В чем различие между стратегиями распределения и объединения?
12. В чем различие между связностью и сочленением?
13. Опишите назначение модели программирования.
14. Что такое интерфейс компонента?
15. Какие существуют типы пользователей на уровне пользовательских сервисов приложения?

ГЛАВА 7

Проектирование презентационного уровня

Занятие 1. Основы дизайна пользовательского интерфейса	215
Занятие 2. Проектирование пользовательского интерфейса	221
Занятие 3. Проектирование компонентов пользовательского процесса	233
Практикум. Создание пользовательского интерфейса	236
Резюме	237
Закрепление материала	238

В этой главе

Вы уже знакомы с процессом проектирования бизнес-решения (в особенности с этапами планирования и создания общей картины решения), состоящим из концептуального, логического и физического дизайна. Однако дизайн пользовательского приложения нельзя считать завершенным, если в нем не указаны способы взаимодействия пользователя с системой, происходящего на презентационном уровне. Проектирование пользовательского интерфейса приложения выполняется на стадии физического дизайна.

На рис. 7-1 показано, какое место в модели процессов MSF занимает дизайн презентационного уровня.

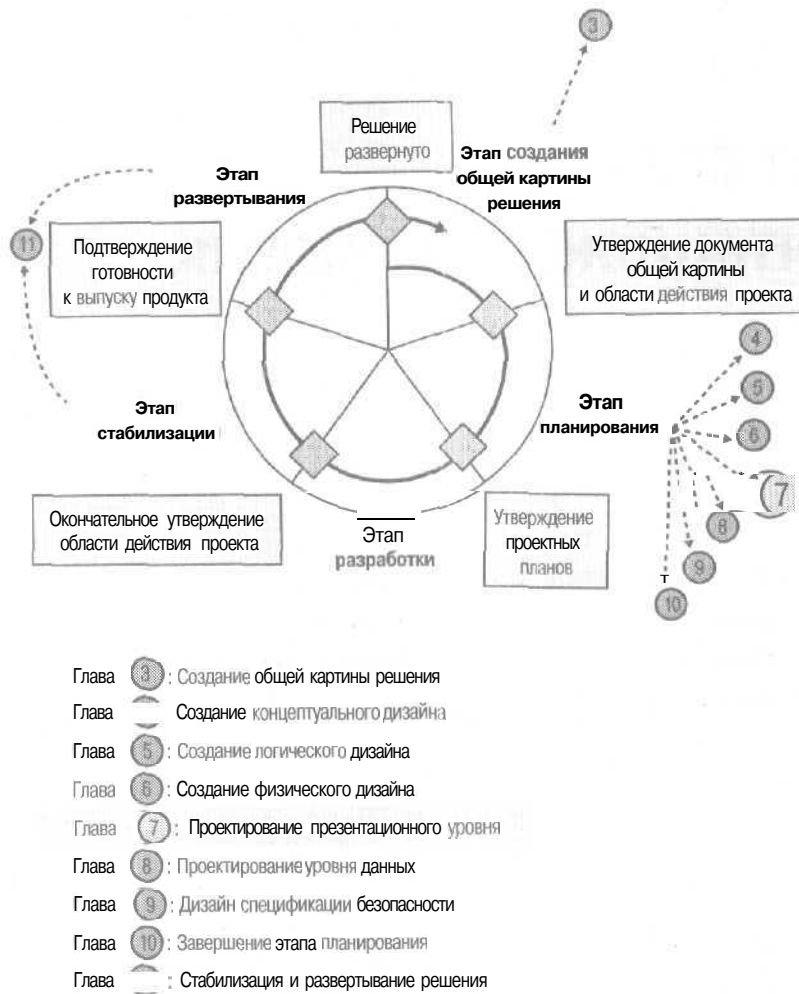


Рис. 7-1. Где мы сейчас находимся в процессе проектирования в MSF

В этой главе рассказывается о проектировании презентационного уровня, о двух составляющих этого уровня — компонентах пользовательского интерфейса и пользовательского процесса, а также о том, как их проектировать.

Прежде всего

Для изучения материалов этой главы необходимо:

- разбираться в модели процессов MSF;
- хорошо знать процесс разработки MSF.

Занятие 1. Основы дизайна пользовательского интерфейса

На этом занятии вы познакомитесь с азами дизайна пользовательского интерфейса: узнаете о различных его типах, целях создания дизайна и отличительных чертах удачного пользовательского интерфейса.

Изучив материал этого занятия, вы сможете:

- ✓ объяснить назначение элементов пользовательского интерфейса;
- ✓ определить принципы создания дизайна пользовательского интерфейса;
- ✓ определить метафоры и элементы, применяемые в дизайне пользовательского интерфейса;
- ✓ различать удачный и неудачный дизайн пользовательского интерфейса.

Продолжительность занятия — около 20 минут.

Презентационный уровень

Презентационный уровень — это часть бизнес-приложения, на котором происходит взаимодействие пользователя с уровнем бизнес-функций системы.

Элементы презентационного уровня

В простейших реализациях презентационного уровня используются *элементы пользовательского интерфейса* (user interface components) из состава графического пользовательского интерфейса (GUI), встроенного в ОС, например Microsoft Windows Forms и Microsoft ASP.NET Web Forms. Для организации более сложного диалога с пользователем создают *компоненты пользовательского процесса* (user process components), которые служат для компоновки компонентов пользовательского процесса и управления взаимодействием с пользователем (рис. 7-2).

Компоненты пользовательского процесса особенно полезны, когда способ доступа к отслеживаемому процессу меняется для различных пользователей. Например, приложению для розничной торговли могут потребоваться два типа пользовательского интерфейса: Интернет-магазин для заказчиков и приложение на основе Windows Forms для торговых представителей. Оба интерфейса позволяют выполнять одни и те же задачи: просматривать список доступных товаров, добавлять выбранные товары в «корзину» и оформлять заказ. Для упрощения поддержки лучше реализовать этот процесс в виде отдельного компонента пользовательского процесса.

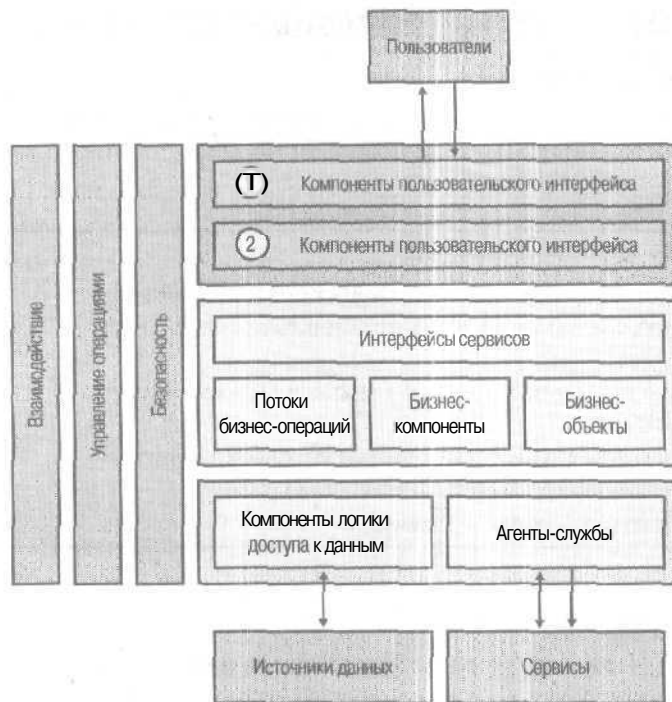


Рис. 7-2. Презентационный уровень

Исходные данные для дизайна презентационного уровня

Результаты, полученные в результате анализа, исследования и оптимизации на этапах создания общей картины решения и планирования, служат исходными данными для дизайна презентационного уровня. К ним относятся:

- требования и ограничения решения;
- сценарии использования системы;
- модели потоков операций;
- профили пользователей;
- описания задач.

Компоненты пользовательского интерфейса

Изучение презентационного уровня начнем со знакомства с пользовательским интерфейсом. В определенной степени это основная часть бизнес-приложения, ведь для большинства пользовательский интерфейс — это и есть само приложение. Стоит заметить, что успех и популярность бизнес-приложения в большой степени зависят от дизайна пользовательского интерфейса.

Компоненты пользовательского интерфейса управляют взаимодействием с пользователем. Они применяются для отображения информации и получения данных от пользователя, интерпретации событий, обусловленных действиями пользователя, изменения состояния интерфейса и информирования о степени выполнения задач.

В схеме «модель — вид — элемент управления» (Model-View-Controller, MVC) компоненты пользовательского интерфейса играют роль представления и/или эле-

мента управления. Согласно этой схеме приложение (или даже интерфейс приложения) делится на три части: модель (объект-приложение), представление (пользовательский вид) и механизм управления (пользовательский элемент управления).

Совет Деление на части — модель, представление и механизм управления — делает решение более гибким и способствует повторному использованию кода,

Если элемент пользовательского интерфейса служит для представления данных, он обеспечивает отображение информации на экране. Механизм управления вызывается, когда в результате действий пользователей бизнес-данные изменяются. Поведение механизма зависит от самих данных и того, с каким элементом интерфейса работает пользователь.

Функции компонентов пользовательского интерфейса

Элементы пользовательского интерфейса служат для отображения информации на экране, ввода и проверки пользовательских данных, а также интерпретации действий пользователя. Кроме того, пользовательский интерфейс должен фильтровать операции, разрешая только те, на которые у данного пользователя есть право.

Компоненты пользовательского интерфейса позволяют:

- получать данные от пользователя и реализовать помощь в виде визуальных подсказок (например, в виде всплывающих контекстных советов), проверять данные и предоставлять элементы управления, необходимые для выполнения задачи;
- перехватывать инициируемые пользователем события и вызывать элементы управления, а также уведомлять компоненты интерфейса о необходимости изменения способа отображения данных (путем выполнения операции или изменения данных в текущем пользовательском процессе);
- ограничивать разрешенные для ввода типы данных. Например, в поле *Age* (возраст) допускается ввод только целых числовых значений;
- проверять вводимые данные, например путем установки разрешенного диапазона или обеспечения обязательного ввода значения;
- выполнять несложные преобразования данных, предоставленных элементом управления, к виду, необходимому для работы элементов, расположенных на более низком уровне. В частности, компонент интерфейса отображает имя товара, но на низший уровень передает его идентификатор;
- форматировать данные (например, приведение даты в нужный формат);
- локализовать предоставляемые данные, например отображать «шапку» таблицы на родном языке пользователя (для чего применяются строки из файла ресурсов);
- отображать информацию о текущем состоянии, например о режиме работы приложения — автономном или сетевом;
- изменять внешний вид приложения в соответствии с предпочтениями пользователя или типом клиентского устройства.

Примечание Компоненты пользовательского интерфейса выполняют и другие функции: координируют необходимые для выполнения задачи действия, контролируют использование ресурсов, группируют информацию (для лучшей наглядности), преобразуют табличные данные в графики (для облегчения восприятия) и обеспечивают локальное кэширование (для повышения производительности).

Рекомендации по созданию дизайна пользовательского интерфейса

Чтобы создать действительно удобный и **эффективный** пользовательский интерфейс, следует хорошо понимать нужды пользователей и отлично представлять весь рабочий процесс.

Дизайн интерфейса должен обеспечивать выполнение задач интуитивно понятным для пользователя способом. Этого добиваются, привлекая к участию самих пользователей на всех этапах проектирования: от создания прототипа интерфейса до бета-тестирования и тестирования «пилотного» варианта.

От пользовательского интерфейса часто зависит успех приложения. Испытывая затруднения при работе с интерфейсом, пользователи видят недостатки приложения. Чтобы спроектировать и разработать интерфейс, действительно отвечающий требованиям пользователей, следует задавать им вопросы и учитывать их **пожелания**.

Вопросы, которые должен задать дизайнер

Есть ряд вопросов, ответы на которые необходимо получить и учитывать при проектировании пользовательского интерфейса:

- Как пользователи будут взаимодействовать с системой?
- Понятны ли пользователям понятия и термины интерфейса?
- Насколько отвечают решению задачи применяемые в дизайне пользовательского интерфейса метафоры?
- Предусмотрена ли возможность при необходимости выполнять автоматизированные действия вручную?
- Насколько легко доступны часто выполняемые задачи?
- Насколько корректно и полно описан рабочий процесс?
- Облегчает ли интерфейс работу пользователей?
- Доступна ли и насколько эффективна справочная информация?
- Есть ли возможность настроить интерфейс для собственных нужд?
- Существуют ли альтернативные способы выполнения задачи на случай неполадок (например, при отключении мыши)?

Особенности успешного дизайна интерфейса

Пользователи судят о приложении по его внешнему виду. Если интерфейс не нравится пользователям, все **старания** разработчиков бесполезны. Тем не менее не стоит жертвовать важными функциями в угоду «дружелюбности» интерфейса. Чтобы пользователи одобрили интерфейс, при его проектировании и разработке во главу угла следует поставить потребности клиентов и существующую последовательность операций. **Необходимо**, чтобы интерфейс был удобен для всех, в том числе для людей с нарушением зрения, слуха и подвижности.

Ниже перечислены характеристики удачно спроектированного и эффективного интерфейса.

- **Интуитивно понятный дизайн.** Проектируйте интерфейс так, чтобы пользователь без лишних подсказок мог разобраться в нем. Пользователь быстрее привыкнет к интерфейсу с **интуитивно** понятным дизайном, ведь интерфейс должен помогать взаимодействовать с системой. Эффективность достигается назначением элементам управления понятных названий и созданием контекстно-зависимой интерактивной справочной системы.
- **Продуманное использование площади экрана.** Тщательно планируйте наполнение интерфейса, стараясь оценить необходимый объем отображаемой ин-

формации и данных, вводимых пользователем. По возможности помещайте все необходимые данные и формы ввода в одном окне. Но иногда информации слишком много — в таких случаях используйте закладки или дочерние окна. Есть еще вариант: написать мастер, который в процессе ввода данных будет подсказывать пользователю.

- **Внешний вид.** Выбор внешнего вида интерфейса зависит от того, как часто пользователи обращаются к той или иной его части и как долго с ней работают.

Например, в формах ввода данных в приложении с Windows-интерфейсом не стоит использовать яркие цвета — это утомляет глаза. Также не выделяйте отдельные элементы определенным цветом (например, сообщения об ошибках часто выделяют красным), так как пользователь может изменить параметры палитры из-за активации «специальных возможностей» или из-за эстетических предпочтений.

В приложениях с Web-интерфейсом, напротив, применение ярких цветов уместно — это добавляет приложению привлекательность. Например, на страницах подтверждения ввода или отображения текущего состояния можно привлечь внимание пользователя на информацию ярким выделением. На ввод данных уходит больше времени, а подтверждение ввода и просмотр текущего состояния приложения занимают небольшую часть общего времени работы с приложением. Вы быстро научитесь создавать удобный пользовательский интерфейс, если будете следовать общепринятым принципам и стандартам, описанным в статье *Official Guidelines for User Interface Developers and Designers* библиотеки MSDN (<http://msdn.microsoft.com/library/en-us/dnwue/html/welcome.asp>).

Примечание Подробнее о дизайне пользовательских интерфейсов приложений, рассчитанных на работу в ОС семейства Windows, рассказано в книге «Microsoft Windows User Experience» (Microsoft Press, 1999).

- **Простота навигации.** Разные люди предпочитают различные средства доступа к компонентам интерфейса, поэтому проектируйте элементы так, чтобы к ним можно было обратиться не только с помощью мыши, но и клавиш Tab, стрелок и других быстрых клавиш. В некоторых случаях, как в примере с вводом данных, скорость навигации важнее наличия помощи при навигации. При создании клавишных комбинаций быстрого вызова стоит учесть характер операции. Например, вводу описания продукта лучше назначить комбинацию клавиш Ctrl+Alt+D*, чем Ctrl+Alt+P. — это понятнее.

- **Управляемая навигация.** Простота навигации очень важна, но не менее важно поддерживать порядок доступа к элементам. Например, в формах ввода или изменения данных разумно предусмотреть определенный порядок ввода значений. Однако будьте осторожны — не лишайте пользователей контроля над процессом ввода информации.

Один из козырей приложений с Windows-интерфейсом в том, что пользователям предоставляется полная свобода в выборе очередности ввода данных. Иногда в интерфейсе стоит наглядно показать цепочку действий. В статье «The Developing Phase» из библиотеки MSDN (<http://msdn.microsoft.com/library/>

* От английских слов Description — «описание» и Property — «свойство». — Прим. перев.

en-us/dnsolac/html/m05_develphase1.asp приводится пример Web-сайта, в котором используется эта возможность,

- **Заполнение полей данными по умолчанию.** Если в интерфейсе присутствуют поля с данными, которые всегда принимают значения по умолчанию, освободите пользователей от обязанности их заполнять, организовав автоматическое заполнение.
- **Проверка входных данных на корректность.** Очень важно убедиться в корректности введенных пользователем данных до того, как они поступят на обработку приложением. Необходимо задать момент проверки данных, например определить, что проверка должна выполняться при переходе фокуса на другое поле или непосредственно перед отправкой данных. Иногда (если существует зависимость данных в полях) стоит сочетать оба подхода.
- **Меню, панели инструментов и справочная система.** Проектируйте интерфейс так, чтобы все выполняемые приложением функции были доступны из меню и панелей инструментов. Кроме того, справочная система должна обеспечивать пользователей полной информацией обо всех предоставляемых возможностях.

Примечание Хотя меню и панели инструментов остаются основными методами ввода данных, стоит добавить другие варианты доступа к наиболее часто используемым функциям приложения — на случай отказа основных устройств ввода, например быстрые комбинации клавиш.

- **Эффективная обработка событий.** Взаимодействием пользователя с интерфейсом управляет код обработки событий. Очень важно, чтобы этот код выполнялся быстро, а время отклика было минимальным.

Занятие 2. Проектирование пользовательского интерфейса

Теперь, получив сведения об основах проектирования пользовательского интерфейса, пора приступить к самому проектированию. На этом занятии рассказывается о процессе проектирования пользовательского интерфейса и о его результатах.

Изучив материал этого занятия, вы сможете:

- ✓ создавать начальный дизайн пользовательского интерфейса;
- ✓ отличать дизайн с высокой и низкой детализацией;
- ✓ проектировать средства помощи пользователям приложения;
- ✓ выбирать *подходящую* для конкретного приложения модель пользовательского интерфейса;
- ✓ определять возможные технологии и конфигурации клиентского окружения;
- ✓ проверять дизайн интерфейса на соответствие требованиям;
- ✓ распознавать *составляющие* элементы процесса создания дизайна пользовательского интерфейса.

Продолжительность занятия — около 35 минут.

Создание начального проекта пользовательского интерфейса

Это первая стадия проектирования пользовательского интерфейса, на которой начальный проект интерфейса создается и предъявляется для оценки пользователями. Детализация не важна — она может быть как низкой (набросок от руки), так и высокой (прототип, созданный средствами Microsoft Visual Basic) (рис. 7-3).



Рис. 7-3. Иллюстрация разницы между низкой и высокой детализацией

Дизайн с низкой детализацией (low-fidelity design) содержит основную структуру и базовые функции интерфейса, а также возможные пути навигации. Он полезен для «мозговых штурмов», когда ответная реакция пользователей следует незамедлительно, быстро выявляются дефекты дизайна и сразу же предлагаются альтернативные варианты.

Дизайн с высокой детализацией (high-fidelity design) содержит подробную информацию о компоновке экрана и элементах интерфейса. Обычно он создается на основе дизайна с низкой детализацией. Такой дизайн с легче реализовывать и модифицировать, чем дизайн с низкой детализацией.

Вдобавок можно создать *карту навигации* (navigation map) по пользовательскому интерфейсу, которая иллюстрирует, какие компоненты вызываются в ответ на то или иное событие *пользовательского* интерфейса. В карте упор делается на описание последовательности операций (workflow), поэтому не включайте в нее проверку на корректность вводимых данных и обработку ошибок.

Привлекая к участию пользователей на ранних стадиях и на протяжении всего процесса проектирования, вы *сокращаете* число ошибок на стадии разработки и повышаете вероятность того, что пользователям *понравится* дизайн интерфейса.

Совет по планированию Постарайтесь вначале предоставить пользователям не прототип, а вид интерфейса с *низкой* детализацией. При создании начальной версии проекта такой дизайн предпочтительнее. Детализированный дизайн можно представить пользователям позже, после того как вы получите больше информации. Такой подход укрепит их в мысли, что перед ними *настоящий* проект, и позволит почувствовать себя соавторами конечной версии дизайна.

В начальных вариантах дизайна необходимо использовать *элементы*, согласованные с пользователями. Термины и понятия проекта должны быть «родными» для пользователей и понятными им. Словарь, выработанный на этапах создания общей картины решения и планирования, годится в качестве основы *используемого* списка *терминов*. Тем не менее для лучшего понимания *используемых* терминов и понятий стоит проводить встречи с пользователями.

Система помощи пользователям

Создавая приложения, часто пренебрегают или неудачно реализуют *помощь* пользователям. Существует множество вариантов поддержки пользователей и оказания им помощи.

Интерактивная справочная система

Интерактивная справочная система предоставляет пользователю информацию по запросу непосредственно в процессе работы с приложением. Справочная служба устанавливается одновременно с приложением, предоставляется на отдельном *компакт-диске* или разворачивается в локальной сети или Интернете. Это важная часть приложения, оперативно *предоставляющая* ответы на стандартные вопросы, которые возникают у пользователей при работе с системой.

Встроенная справочная система бывает контекстно-зависимой или в виде набора ссылок на статьи. Первая предоставляет информацию о конкретном поле или области пользовательского интерфейса. В других видах справочной системы доступ пользователей к содержимому возможен в определенном порядке (по

списку тем или в алфавитном порядке), иногда они поддерживают поиск информации.

Примечание При проектировании справочной системы позаботьтесь об ответах на вопросы, возникающие при решении конкретных проблем. Подобная помощь очень поможет слабо подготовленным пользователям.

Всплывающие подсказки

Всплывающая подсказка (tooltip) — это крошечное окно, появляющееся, когда пользователь подводит указатель мышки к элементу управления или пункту меню, и описывающее назначение элемента. Всплывающими подсказками практически в обязательном порядке оснащаются панели инструментов, но они успешно применяются и во многих других частях интерфейса.

Совет Как и в других частях интерфейса, тщательно следите, чтобы описание элемента было четкими и ясно передавало его назначение.

Строки состояния

Строки состояния (status display) полезны для отображения указаний или сообщений, не помещающихся в отведенное для всплывающей подсказки окно. Для реализации строк состояния применяются элементы управления «строка состояния» и «надпись».

Мастера

Мастер (wizard) помогает пользователю, управляя его действиями при какой-либо последовательности операций. Обычно мастера помогают выполнить определенную, достаточно сложную задачу, решить которую иными средствами удастся, лишь затратив массу времени на ее освоение и выполнение. Они также полезны при предоставлении дополнительной информации пользователям, имеющим достаточный опыт в выполнении базовых действий, но желающих освоить более сложные операции.

Специальные возможности

Специальные возможности (accessibility aids) — специализированные программы и устройства, облегчающие работу с приложением людям с различными нарушениями. Существует множество подобных программ, в том числе:

- «увеличители» экрана, позволяющие сделать крупнее изображение выбранной части экрана;
- программы озвучивания информации на экране, преобразующие графические и текстовые данные в речь;
- системы распознавания голоса, позволяющие выполнять ввод информации с голоса, а не с клавиатуры или мыши.

При создании таких приложений необходимо следовать определенным принципам:

- приложения должны поддерживать определенные системные параметры палитры, размеров экрана, шрифтов, звука и ввода. Это позволит унифици-

ровать пользовательский интерфейс всех приложений на компьютере пользователя;

- необходимо предусмотреть возможность доступа ко всем функциям приложения с клавиатуры. Это позволит вполне комфортно работать без таких указательных устройств, как мышь;
- звуковая информация в приложениях должна сопровождаться другими сигналами, то есть звуковые сообщения должны дублироваться другими средствами, например сообщениями на экране.

Примечание Более подробную информацию о принципах проектирования приложений со специальными возможностями вы найдете в библиотеке MSDN (<http://msdn.microsoft.com>) и на специальном сайте Microsoft (<http://www.microsoft.com/enable>).

Выбор модели пользовательского интерфейса

При проектировании приложения важно выбрать наиболее подходящую модель пользовательского интерфейса, так как это влияет на процесс развертывания, способы взаимодействия пользователей с данными, пути поддержки текущего состояния на протяжении диалога приложения и пользователя. К наиболее распространенным моделям и технологиям реализации пользовательского интерфейса относятся:

- стандартный пользовательский интерфейс Windows;
- Web-интерфейс;
- интерфейс мобильных устройств;
- интерфейс на основе документов.

Стандартный интерфейс Windows

Стандартный интерфейс Windows применяется, когда необходимо обеспечить работу пользователей в автономном режиме и когда необходима богатая функциональность системы. Он также позволяет эффективно управлять состоянием и обеспечивает постоянство (persistence) данных, а также все преимущества локальной обработки данных. Существует три основных категории автономных пользовательских интерфейсов:

- **Полнофункциональный пользовательский интерфейс для рабочих станций и планшетных компьютеров на основе Windows Forms.** Характерен для приложений, созданных с применением Windows Forms и стандартных элементов управления Windows. В подобных приложениях реализуется полный или практически полный цикл обработки данных. Пользовательский интерфейс такого рода предоставляет разработчику полный контроль над рабочим окружением, а также внешним видом и возможностями приложения.
- **Встроенный HTML-код.** Кроме Windows Forms при создании пользовательского интерфейса в Windows-приложение можно встроить дополнительный HTML-код. Это предоставляет большую гибкость во время исполнения (при работе в сети HTML-ресурсы могут загружаться из внешних источников или даже баз данных) и предоставляет возможность настройки с учетом потребностей пользователя. Однако для загрузки и вывода HTML-страниц, а также привязки элементов управления к функциям приложения необходимо писать дополнительный код.

Внимание! Создавая встроенный HTML-код на этапе планирования обязательно убедитесь, что его нельзя использовать для внедрения зловердных сценариев.

- **Встраиваемые модули (add-in).** Иногда варианты использования системы подсказывают, что пользовательский интерфейс приложения лучше реализовать в виде модуля, встраиваемого в другие приложения (например, в Microsoft Office, AutoDesk AutoCAD, CRM-приложения, инженерные инструментальные средства и т. п.). В этом случае разработчики могут применять всю логику доступа к данным и отображения информации основного приложения – остается лишь добавить код, отражающий специфику бизнеса, который обеспечит сбор данных и работу бизнес-логики.
- **Удаленный доступ.** При помощи удаленного рабочего стола Windows XP и службы терминалов Windows пользователи могут удаленно получить доступ к компьютерам и запускать на них приложения. Между клиентским компьютером и удаленным пользователем передаются только данные пользовательского интерфейса. Все остальные функции (проверка данных и процессорные вычисления) выполняются на удаленном компьютере.

Web-интерфейс

В Microsoft .NET пользовательский Web-интерфейс разрабатывается средствами ASP.NET. Эта технология предоставляет богатую функциями среду, позволяющую создавать сложные Web-интерфейсы. Вот лишь некоторые из возможностей ASP.NET:

- унифицированная среда разработки;
- привязка данных к пользовательскому интерфейсу;
- интерфейс на основе компонентов с элементами управления;
- встроенная модель безопасности каркаса .NET;
- широкие возможности по поддержке кэширования и управления состоянием;
- доступность, производительность и масштабируемость Web-обработки данных.

Интерфейсы для мобильных устройств

Мобильные устройства, например КПК, телефоны с поддержкой WAP (Wireless Application Protocol), становятся все популярнее, поэтому формируется новый круг задач — создание пользовательских интерфейсов для мобильных устройств.

Информация на мобильных устройствах должна уместиться на значительно меньшей площади экрана, а интерфейс должен оставаться достаточно удобным в работе. Взаимодействие с пользователем на мобильных устройствах затруднено, поэтому в интерфейсах таких систем объем вводимых данных сведен до минимума. Обычный в таких случаях принцип таков: пользователи заранее записывают возможные данные на мобильное устройство средствами клиентской программы на обычной рабочей станции, а затем просто отмечают нужные значения на мобильном устройстве. Например, в приложении электронной коммерции заранее вводят данные кредитной карты на Web-сайте, а на мобильном устройстве просто выбирают нужную кредитную карту из списка зарегистрированных — не нужно набирать всю информацию о кредитной карте с крошечкой клавиатуры мобильного телефона.

Пользовательский интерфейс на основе документов

При разработке некоторых приложений можно воспользоваться тем фактом, что пользователи вводят и просматривают информацию в виде документов. Пользовательский интерфейс на основе документов предоставляет ряд функций.

- **Отображение данных.** Пользовательские данные представляются в виде документа, например данные о покупателе — в виде документа Microsoft Word, отчет о продажах — в виде таблицы Microsoft Excel, а график работ — в виде диаграммы Гантта в Microsoft Project.
- **Сбор данных.** Возможна такая ситуация: торговые представители вводят информацию о заказчиках в формы Word, бухгалтеры вводят бухгалтерскую информацию в таблицы Excel, проектировщики размещают проектные данные в файлах Microsoft Visio.

Выбор клиентской среды

Выбор клиентской среды для пользовательского интерфейса бизнес-приложения определяется способом работы пользователей с приложением и видом подключения к системам, обеспечивающим работу приложения.

Если пользователи работают в локальной или высокоскоростной глобальной сети, основным вариант — использование полнофункциональной среды, или так называемого «толстого» клиента (rich client). В таких случаях скорость передачи данных между компьютерами пользователей и системой, к которой они обращаются, практически не лимитирована. Вдобавок обработку данных в приложении легко распределить между серверами и клиентами.

Другой вариант — «тонкий» клиент (thin client). Обычно он реализуется в виде Web-браузера или удаленного рабочего стола и предназначается удаленным и мобильным пользователям или пользователям с медленным (например, модемным) доступом. Для последних скорость связи очень важна, поэтому необходим интерфейс, иницирующий очень небольшой трафик. Но иногда без «толстого» клиента не обойтись, например, если для работы требуются функции, которые не удастся реализовать иначе.

Выбирая «толстый» или «тонкий» клиент, необходимо учесть ряд обстоятельств.

- **Клиентские устройства.** Приложение, предназначенное для работы самых разнообразных клиентских устройств, лучше всего организовать в виде «тонкого» клиента. Если приложение будет работать как на компьютерах под управлением Windows, так и на КПК, телефонах и интерактивных телеприставках, то предпочтительнее именно такой вариант.

Особую осторожность следует соблюдать при работе в гетерогенных средах, так как возможности клиента часто оказываются ограниченными. Например, в браузере может отсутствовать поддержка ActiveX-элементов или виртуальная машина Java. Если в качестве интерфейса приложение используется браузер, тип которого заранее не известен, рекомендуется применить серверные элементы управления ASRNET — сервер самостоятельно определит возможности браузера и предоставит подходящие элементы управления. Если интерфейс предназначается для мобильных устройств, для тех же нужд стоит применить Mobile Internet Controls.

- **Графика.** Если в приложении интенсивно используется графика, например проекты CAD или видео-приложения, предпочтительнее «толстый» клиент. Если приложение не слишком перегружено графикой и в нем не применяется анимация и другие «украшательства», его вполне можно реализовать в виде

«тонкого» клиента. Использование динамического HTML позволяет создавать графические эффекты, но за счет увеличения размера Web-страниц. При этом доступная функциональность зависит от возможностей браузера, на который рассчитано приложение.

- **Интерактивность.** Если в приложении полно интерактивных функций и оно рассчитано на активную работу пользователя, решение лучше реализовать в виде «тонкого» клиента. В значительной степени интерактивные приложения отличает динамический пользовательский интерфейс, часто меняющийся в зависимости от вводимых данных (примеры: текстовые редакторы, игры. CAD-пакеты), поддержка «перетаскивания» объектов, проверка вводимых данных «на лету» и т.п.
 - **Пропускная способность сети.** Приложение, выполненное в виде «тонкого» клиента, обычно больше нагружает сеть, особенно когда в приложении используются серверные Web-элементы, так как практически каждый щелчок кнопки мыши инициирует обращение к серверу. Это же относится к клиентам службы терминалов и удаленного рабочего стола, потому что информация о действиях пользователей и ответы сервера передаются через сеть. Поэтому в сети с низкой пропускной способностью выгоднее «толстый» клиент.
 - **Автономный клиент.** Если необходимо обеспечить работу клиента в отсутствие подключения к корпоративной сети, используйте «толстый» клиент. Это же относится к приложениям для служащих, часто бывающих в разъездах или работающих на территории заказчиков.
 - **Операции, требующие процессорных ресурсов.** Если для обслуживания пользователей требуются значительные процессорные ресурсы (например, системы CAD и графические приложения), их следует выполнять на локальной машине, в противном случае возможна перегрузка сервера. Приложения, нуждающиеся в вычислительных мощностях, лучше всего реализовать как «толстый» клиент.
 - **Блокировки в базе данных.** Если в приложении интенсивно используется параллельный доступ и необходимо активно управлять им, решение лучше создавать как «толстый» клиент. Например, приложение, использующее жесткую блокировку данных, не может работать как «тонкий» клиент, так как блокировки освобождаются только после обработки каждой страницы.
 - **Локальные ресурсы.** Когда приложению нужен доступ к таким локальным ресурсам, как файлы (хранилище), реестр (конфигурационные данные), базы данных (хранилище) или другие локальные устройства, предпочтительнее «толстый» клиент.
 - **Безопасность.** В среде где требуется высокая безопасность приложения, «тонкий» клиент имеет ряд очевидных преимуществ:
 - выполнение высоко привилегированных операций ограничивается сервером;
 - код, который злоумышленник мог бы использовать в собственных интересах, не размещается на рабочих станциях (ясно, что подразумевается отсутствие активного содержимого);
 - конфигурацию рабочей станции можно заблокировать;
 - конфиденциальная информация хранится только в центре хранения данных.
- У приложений, реализованных в виде «толстого» клиента и использующих службу терминалов, есть ряд преимуществ:
- авторизация на уровне пользователей, механизм управления доступом из кода;

- аутентификация;
- безопасность связи и управления состоянием;
- аудит.

Создание прототипа пользовательского интерфейса

После выбора дизайна пользовательского интерфейса переходят к созданию прототипа пользовательского интерфейса, взяв за основу данные интервью, документы с требованиями, варианты и сценарии использования системы, диаграммы действий, созданные на этапе планирования.

Пример сценария использования системы

Далее описывается ВИС для торговых представителей и менеджеров по продажам, созданный на основе информации, собранной на предыдущих стадиях планирования.

Вариант использования системы для компании Adventure Works Cycles

Название ВИС: Предоставление скидки на товар

Сокращенное название: Предоставление скидки на товар

Идентификатор ВИС: UC 05.5.1

Идентификаторы требований: 2.1.1.2.1.2, 2.1.3

Описание: Торговый представитель создает заказ для конкретного заказчика. Представитель желает предоставить заказчику скидку на один или более товаров заказа. Скидка влияет на общую стоимость заказа, и конечная сумма пересчитывается. Менеджеры по продажам также уполномочены предоставлять скидки на заказы.

В настоящее время торговым представителям разрешается без дополнительного согласования предоставлять скидки до 15%, а менеджерам по продажам - до 20%.

Действующие субъекты: Торговый представитель, менеджер по продажам

Предварительные условия: Действующий субъект имеет права на просмотр данных о заказчиках в продажах. Заказ находится в стадии создания,

Последовательность развития событий

Исключения

- | | |
|---|---|
| <ol style="list-style-type: none"> 1. Действующий субъект решает предоставить скидку на заказ. 2. Приложение запрашивает, в каком заказе следует предусмотреть скидку. 3. Действующий субъект выбирает заказ. 4. Приложение предоставляет информацию о заказе (включая сведения о текущих скидках) и запрашивает размер скидки, 5. Действующий субъект вводит размер скидки. | <p>Товар не зиден в текущем представлении.</p> <p>Информация полностью не доступна.</p> |
|---|---|

6. Приложение проверяет размер скидки и предоставляет информацию о заказе, в том числе скорректированную сумму заказа.
- Если размер скидки превышает разрешенный, система выдает предупреждение.
 - ВИС продолжается с пункта 5: действующему субъекту предлагается повторить ввод размера скидки.
7. Информация передается в подсистему доставки.

Постусловия:

ВИС, которые используют или расширяют данный:

1. Расширяет ВИС 05.5: «Предоставление скидки на заказ».

Нерешенные задачи: Менеджеры по продажам обладают полномочиями предоставлять скидки большего размера, чем торговые представители. Это указано в расширенном ВИС 5.5.1.1

Ответственный: Майк Дансельйо

История изменений:

Дата: 19 декабря 2002 г.

Автор: Ян Ли

Описание: Начальная версия

Примечание При проектировании пользовательского интерфейса следует учитывать предварительные условия, основной и альтернативный пути развития событий, а также расширения, определенные в сценариях использования. Все эти обстоятельства должны учитываться в пользовательском интерфейсе.

Пример требований

Далее описаны некоторые требования, выявленные и уточненные во время этапа планирования ВИС для компании Adventure Works Cycles:

- торговому представителю необходимо предоставить возможность искать информацию на сайте среди всех товаров или в определенном каталоге;
- торговому представителю необходимо разрешить просматривать категории товаров;
- торговому представителю необходимо предоставить возможность просматривать такие сведения о товаре, как цены и информация о рекламных акциях;
- торговому представителю необходимо разрешить просматривать спецификации товаров;
- на странице необходимо предусмотреть дополнительные ссылки для навигации: «На главную страницу», «О компании», «Отслеживание заказов», «Цены и условия доставки», «Контакты» и заявление об авторском праве;
- торговому представителю необходимо предоставить возможность просматривать отзывы покупателей о продуктах;
- торговому представителю следует разрешить предоставлять покупателям скидки до 15%;
- торговому представителю необходимо дать возможность предоставлять покупателям скидки в размере в пределах 16–20% с предварительного разрешения менеджера по продажам;

- менеджеру по продажам следует разрешить искать информацию на сайте: среди всех товаров или же в определенном каталоге;
- менеджеру по продажам необходимо предоставить возможность просматривать категории товаров;
- менеджеру по продажам необходимо разрешить просматривать такие сведения о товаре, как цены и информация о рекламных акциях;
- менеджеру по продажам следует предоставить возможность просматривать спецификации товаров;
- менеджеру по продажам необходимо разрешить просматривать отзывы покупателей о продуктах;
- менеджеру по продажам следует дать возможность предоставлять покупателям скидки до 20%;
- менеджер по продажам вправе одобрять предоставление покупателям скидки до 20%.

Пользовательский интерфейс Web-сайта компании Adventure Works Cycles должен полностью удовлетворять этим требованиям.

Образцы прототипов пользовательского интерфейса

На рис. 7-4 показан прототип пользовательского интерфейса, созданный в соответствии с требованиями, полученными на основании вариантов и сценариев использования системы. Прототип содержит информацию, предоставляемую торговому представителю на странице с информацией о товарах.

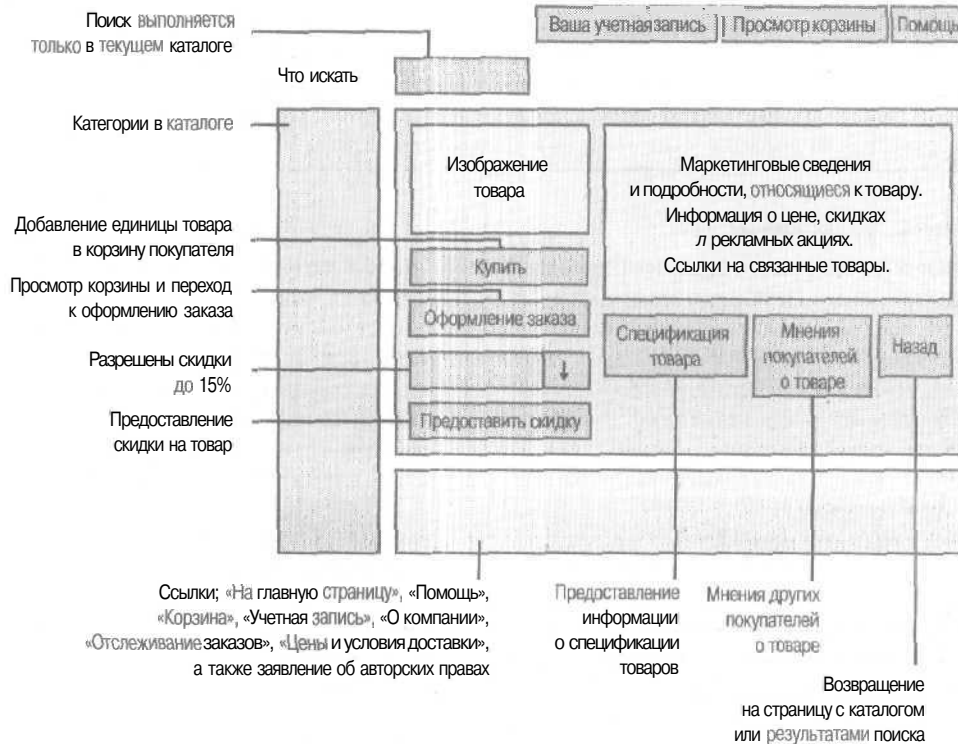


Рис. 7-4. Прототип страницы с информацией о товаре, предназначенной для торговых представителей

На рис. 7-5 показан другой прототип интерфейса, который содержит сведения о товаре, необходимые менеджеру по продажам.

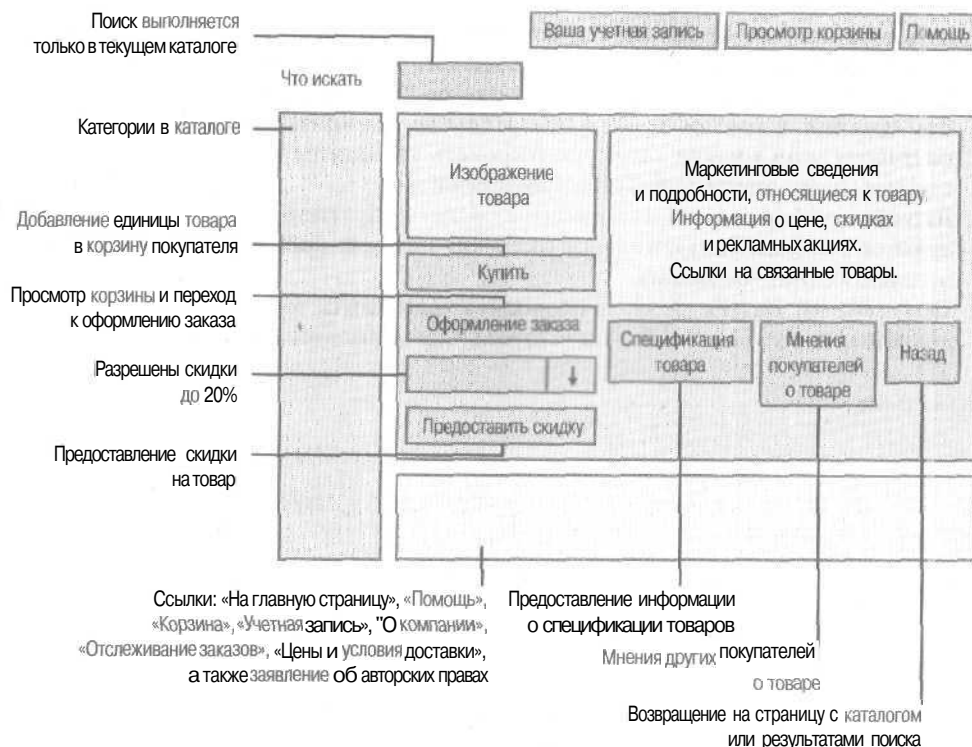


Рис. 7-5. Прототип страницы с информацией о товаре, предназначенной для менеджеров по продажам

Проверка дизайна интерфейса на соответствие требованиям пользователя

После создания прототипов пользовательского интерфейса необходимо проверить, насколько они соответствуют требованиям, вариантам и сценариям использования системы, а также логическому дизайну. Если какое-либо требование не нашло отражения в пользовательском интерфейсе, необходимо исправить ситуацию до предоставления прототипа покупателям.

Карты навигации и потоки данных проверяются на соответствие ожидаемым, Карты необходимо проверить на соответствие СИС следующих этапов, чтобы убедиться, что пользовательский интерфейс соответствует сценарию.

На завершающем этапе процесса проверки пользователи должны оценить дизайн и подтвердить его соответствие требованиям.

Следующий этап — создание прототипов с ограниченной функциональностью, позволяющих проверить удобство использования интерфейса.

Результаты процесса создания дизайна пользовательского интерфейса

Подобно остальным процессам проектирования бизнес-решения, процесс дизайна интерфейса предусматривает определенные результаты.

- Группа разработчиков, заказчик и пользователи договариваются о принципах дизайна, в том числе и об обязательных элементах приложения. Группа разработчиков должна задокументировать принципы построения дизайна и стандартные элементы будущего интерфейса.
- Дизайн должен содержать описание средств обратной связи (таких, как индикатор выполнения) и порядок оказания помощи пользователям (таких, как всплывающие подсказки).
- Полученный дизайн должен позволять выполнить тестирование на основе архивных документов и будущих сценариев использования системы.

Занятие 3. Проектирование компонентов пользовательского процесса

Взаимодействие пользователя с приложением должно быть предсказуемым. Допустим, на Web-сайте компании Adventure Works Cycles пользователю разрешается вводить сведения о товаре, просматривать полную стоимость, указывать сведения об оплате и адрес доставки. Этот процесс предусматривает отображение и ввод данных в различных компонентах пользовательского интерфейса, при этом состояние процесса (заказанные товары, сведения о кредитной карте и т.п.) должно сохраняться при переходе с одного этапа процесса на другой. Создав компоненты пользовательского процесса, вы сможете согласовать работу пользователя страниц или форм пользовательского интерфейса.

На этом занятии рассказывается о функциях компонентов пользовательского процесса и методах их проектирования.

Изучив материал этого занятия, вы сможете:

- ✓ рассказать о функциях компонентов пользовательского процесса;
- ✓ разделить пользовательский интерфейс и пользовательский процесс;
- ✓ спроектировать пользовательские процессы.

Продолжительность занятия — около 20 минут.

Функции компонентов пользовательского процесса

У разделения функций взаимодействия с пользователем на компоненты пользовательского интерфейса и пользовательского процесса есть ряд преимуществ.

- Упрощается хранение информации о состоянии при длительном взаимодействии с пользователем, позволяя закрывать и восстанавливать клиентскую сессию в прежнем состоянии, даже если задействован другой интерфейс. Например, покупатель может добавить несколько товаров в корзину в Web-интерфейсе, а затем позвонить торговому представителю и завершить оформление заказа.
- Один и тот же пользовательский процесс может обслуживать несколько пользовательских интерфейсов. Например, в приложении для розничных продаж один-единственный пользовательский процесс может поддерживать добавление товара в корзину как посредством пользовательского Web-интерфейса, так и через приложение Windows Forms.

Рис. 7-6 иллюстрирует, чем отличаются пользовательский интерфейс и пользовательский процесс.

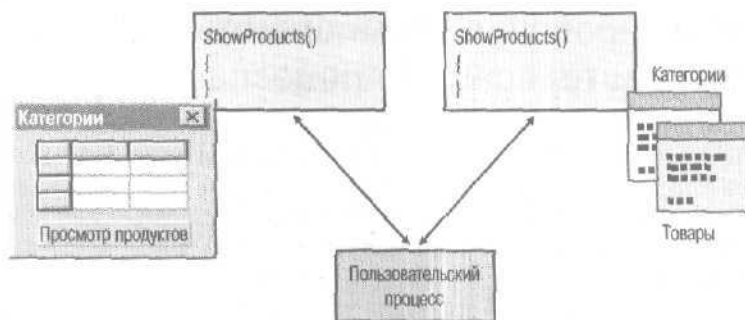


Рис. 7-6. Пользовательские интерфейсы и компоненты пользовательского процесса

Компоненты пользовательского процесса обычно реализуются в виде классов, предоставляющих свои методы для вызова из пользовательских интерфейсов. Отдельные методы инкапсулируют логику, необходимую для выполнения того или иного действия в интерфейсе. Пользовательский интерфейс создает экземпляр компонента пользовательского процесса и применяет его для прохождения всех этапов процесса.

Компоненты пользовательского процесса координируют вывод элементов пользовательского интерфейса. Они отделены от функций обработки и сбора данных, предоставляемых в элементах пользовательского интерфейса. При их проектировании необходимо учитывать возможность работы приложения в других странах, поэтому интерфейс должен быть локализуемым. Например, внутри приложения старайтесь задать не зависящие от региона форматы данных и строки в формате Unicode — это упростит доступ к компонентам пользовательского процесса из локализованного интерфейса.

Компоненты пользовательского процесса:

- предоставляют простой способ объединения элементов интерфейса в потоки взаимодействия с пользователем без изменения структуры потоков данных и логики управления;
- отделяют схему взаимодействия с пользователем от конкретной реализации или конкретного устройства, с которым тот работает;
- инкапсулируют влияние исключений на потоки пользовательского процесса;
- поддерживают состояние взаимодействия с пользователем;
- поддерживают внутренние данные, относящиеся к бизнес-логике и внутреннему состоянию приложения, при необходимости сохраняя данные в постоянной памяти.

Разделение пользовательского процесса и интерфейса

Перед проектированием компонентов пользовательского процесса необходимо разделить компоненты пользовательского процесса и интерфейса. Пользовательский процесс отделяют от пользовательского интерфейса в несколько стадий.

1. Выясните, какие бизнес-процессы будет поддерживать пользовательский процесс. Определите, как это выглядит с точки зрения пользователя. Обычно для этого сверяются с диаграммами последовательностей, вариантами и сценариями использования системы, созданными в процессе анализа требований.

2. Определите, какие данные необходимы для бизнес-процесса. Пользовательский процесс должен «уметь» при необходимости обращаться к этим данным.
3. Определите, какую дополнительную информацию о состоянии следует поддерживать во время сеанса пользователя для нормальной обработки и получения данных в пользовательском интерфейсе,
4. Создайте наглядные схемы потоков пользовательского процесса и способы получения и передачи потоков управления каждым элементом пользовательского интерфейса.

Примечание Реализация пользовательского взаимодействия с компонентами пользовательского процесса — непростая задача. До ее решения следует тщательно оценить, требуется ли приложению тот уровень согласования и абстракции, который предоставляется элементами пользовательского процесса.

Принципы проектирования пользовательских процессов

- Решите, необходимо ли управлять пользовательскими процессами в виде компонентов, отдельных от компонентов интерфейса. Разделение обычно требуется в приложениях с большим количеством диалоговых окон или в прикладных программах, где следует предусмотреть возможность настройки пользовательских процессов и/или подключаемых модулей.
- Определите место хранения информации о состоянии пользовательского процесса.
 - Если процесс работает в сети, информацию о промежуточных состояниях долгоживущих процессов лучше хранить в центральной базе данных Microsoft SQL Server. В автономных процессах данные предпочтительно хранить в локальных XML-файлах, отдельных хранилищах или локальной базе данных Microsoft SQL Server 2000 Desktop Engine (MSDE). На КПК информация о состоянии хранится в Microsoft SQL Server CE.
 - Если процесс короткоживущий и не нуждается в восстановлении в случае сбоя, информацию о состоянии можно хранить в оперативной памяти. Обычно так делают в «толстых» клиентах. В Web-приложениях для этого можно использовать объект *Session* среды ASPNET. Если приложение развернуто на ферме серверов, рекомендуется применять SQL Server. Фермы серверов — это группа Web-серверов, в которых организована балансировка нагрузки. При реализации Web-сервера на основе фермы, нагрузка распределяется между несколькими Web-серверами. В ASP.NET реализовано удаление лишних сеансов, хранимых в SQL Server, для предотвращения накопления ненужных данных.
- Компоненты пользовательского процесса должны поддерживать сериализацию, что позволяет обеспечить отказоустойчивость.
- Организуйте обработку исключений в компонентах пользовательского процесса, и распространите действие исключений на пользовательский интерфейс. Позаботьтесь, чтобы исключения, инициируемые компонентами пользовательского процесса, перехватывались элементами интерфейса.

Практикум. Создание пользовательского интерфейса



Сейчас вы используете полученные знания для создания пользовательского интерфейса.

Вам поручили создать прототип приложения для управления информацией о товарах. Товароведам эта система необходима для управления товарами в приложении для онлайн-торговли. Проект прототипа требуется создать в Microsoft PowerPoint. К приложению предъявляется ряд требований.

- Приложение должно позволять сотрудникам просматривать информацию о товаре по идентификационному номеру или выполнять поиск в базе данных товаров. Возможны два варианта поиска: в конкретном каталоге или по всем каталогам.
- Информацию о каждом товаре следует группировать по разделам: **название**, описание, номер, каталог, цены, дата создания записи, срок годности, дата внесения последнего изменения, изображение товара, спецификации, информация о рекламных акциях и скидках. Сотрудник должен иметь возможность задавать тип отображаемой информации и работать с выбранными данными, а не получать всю массу информации о товаре в одном окне.
- При добавлении нового товара должна создаваться новая запись о товаре и должен генерироваться новый идентификатор, после чего товаровед вводит разделы информации о товаре. Разделы бывают обязательными или необязательными. После ввода всей информации сотрудник должен явным образом сохранить изменения в базе данных.
- При редактировании информации о существующем товаре сотрудник должен найти товар в базе, явным образом объявить о предстоящем изменении сведений, выбрать нужные разделы и внести изменения. Далее необходимо явным образом **сохранить** коррективы в базе данных.
- Сотруднику разрешается отдельно отменять изменения в любом из разделов информации о товаре, то есть откат следует применять только к выбранному разделу. Например, сотрудник, изменивший описание, а затем информацию о продвижении товара, может отменить коррективы о продвижении товара, не затрагивая описание товара.
- Пользователю разрешается отменить все новые и измененные записи, выполненные с момента последнего сохранения записи о товаре.
- При удалении существующего товара, служащий должен найти товар в базе данных, явным образом указать намерение удалить запись и подтвердить факт удаления. Запись удаляется из базы данных путем изменения срока действия на предшествующую дату.

Не забудьте указать функции, доступные в пользовательском интерфейсе, и возможности, предоставляемые каждой его областью. Спроектировав пользовательский интерфейс, еще раз внимательно проверьте, насколько он соответствует требованиям. Одно из возможных решений вы найдете в презентации *C07Ex1_Answer.ppt*, расположенной в папке `\Solutions-Documents\Chapter07` на прилагающемся к книге компакт-диске.

Резюме

- Дизайн любого приложения нельзя считать завершенным, пока не определены способы взаимодействия пользователей с системой на презентационном уровне. Презентационный уровень служит мостом между пользователем и уровнем бизнес-функций системы.
- Простейшая реализация презентационного уровня содержит компоненты пользовательского интерфейса, например на основе Windows Forms и ASP.NET Web Forms. Для поддержки более сложного взаимодействия с пользователем организуют механизм координации действий компонентов процесса и управления взаимодействием с пользователем.
- Компоненты пользовательского интерфейса применяются для отображения информации и получения данных от пользователя, интерпретации событий, обусловленных действиями пользователя, изменения состояния интерфейса и информирования о степени выполнения задач. Кроме того, пользовательский интерфейс должен фильтровать операции, оставляя только разрешенные в данный момент времени,
- Отличительные характеристики удачно спроектированного пользовательского интерфейса:
 - интуитивно-понятный дизайн;
 - оптимальное использование экранного пространства;
 - простота навигации;
 - управление навигацией;
 - « меню, всплывающие окна и справочная система;
 - эффективная обработка событий.
- Существует несколько видов моделей пользовательского интерфейса:
 - стандартный интерфейс Windows;
 - Web-интерфейс;
 - интерфейс мобильных устройств;
 - интерфейс на основе документов.
- Существует несколько вариантов предоставления помощи пользователю:
 - встроенная справочная система;
 - подсказки;
 - строки состояния;
 - мастера;
 - специальные возможности.
- Компоненты пользовательского процесса обычно реализуются в виде классов .NET, которые предоставляют свои методы для вызова из пользовательских интерфейсов. Эти компоненты обеспечивают согласованное отображение компонентов пользовательского интерфейса. Они отделены от функций обработки и сбора данных, поддерживаемых компонентами пользовательского интерфейса.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Какова роль презентационного уровня в архитектуре бизнес-приложения?
2. Каковы отличительные черты удачного пользовательского интерфейса?
3. Чем отличается дизайн с высокой и низкой детализацией?
4. Какие средства доступны разработчикам для организации помощи пользователям?
5. Какие существуют типы моделей пользовательского приложения и в каких случаях их стоит применять?
6. В чем разница между компонентами пользовательского интерфейса и пользовательского процесса. Приведите пример, когда применение компонентов пользовательского процесса предпочтительнее.
7. Как отделяют интерфейс от пользовательского процесса?
8. Вы проектируете систему, осуществляющую запросы с использованием службы терминалов Windows. Какой тип пользовательского интерфейса следует выбрать?
9. На этапах создания общей картины решения и планирования выяснилось, что пользователи будущего решения работают на различном оборудовании в разных странах земного шара и в большинстве своем не имеют доступа к корпоративной интрасети. Какой тип клиента предпочтительнее в такой ситуации?
10. **Вы** закончили проектирование пользовательского интерфейса. Как проверить дизайн до его реализации?

Проектирование уровня данных

Занятие 1. Проектирование хранилища данных	240
Занятие 2. Оптимизация доступа к данным	251
Занятие 3. Проверка данных	259
Практикум. Создание схемы данных	265
Резюме	266
Закрепление материала	268

В этой главе

На этапе планирования модели процессов MSF проектная команда наряду с дизайном презентационного и бизнес-уровней создает дизайн уровня данных, о котором и пойдет речь в этой главе. Вы также узнаете об оптимизации доступа к данным и реализации проверки данных.

Прежде всего

Для изучения материалов этой главы необходимо:

- знание различных этапов модели процессов MSF;
- понимание этапа создания общей картины в модели процессов MSF;
- понимание задач и результатов концептуального и логического дизайна на этапе планирования;
- понимание принципов проектирования презентационного уровня и бизнес-уровня решения в процессе физического дизайна.

Занятие 1. Проектирование хранилища данных

Уровень данных (data layer) продукта состоит из хранилища и сервисов данных. *Хранилище данных* (data store) — это, как правило, база данных. На этом занятии вы познакомитесь с моделями данных и узнаете, как структурируются данные в различных моделях. Здесь рассказывается об определении *сущностей* и атрибутов в модели данных, а также о разработке таблиц и столбцов хранилища данных и реализации связей между различными сущностями.

Изучив материал этого занятия, вы сможете:

- ✓ описать схему базы данных;
- ✓ определить сущности и атрибуты;
- ✓ определить таблицы и колонки для хранилища данных;
- ✓ реализовать связи.

Продолжительность занятия — около 20 минут.

Схема базы данных

На этапе планирования проектная команда основное внимание уделяет анализу требований и созданию *удовлетворяющего им дизайна решения*. Таким образом, помимо определения функций будущего продукта проектная команда анализирует требования к данным и определяет, как их следует структурировать, как они будут храниться и проверяться и как обеспечить к ним доступ.

Изучение и анализ требований к данным начинается на этапе концептуального дизайна. Требования позволяют определить, что именно должно хранить и обрабатывать бизнес-решение. В процессе логического дизайна проектная команда выявляет набор сущностей данных на основании логической модели объектов, сценариев использования и таких артефактов данных, как схема, триггеры, ограничения и топология существующего хранилища данных. В процессе физического дизайна команда создает схему данных, определяя таблицы, связи, типы данных для полей и индексы и завершает работу над сервисами данных. Кроме того, планируются мероприятия по переносу данных, резервному копированию и восстановлению данных, а также обеспечению отказоустойчивости.

Определение схемы данных

База данных (БД) — это особым образом организованный набор значений данных, а схема БД определяет, как именно организованы данные в БД. В процессе физического дизайна члены проектной команды создают схему БД, чтобы определить, что именно нужно создавать, о том же, какими инструментами это будет реализовываться, следует думать позже.

В процессе логического дизайна команда описывает сущности и атрибуты, которые будут храниться в БД, и то, как пользователи будут получать к ним доступ, оперировать ими и просматривать их. В процессе физического дизайна команда создает схему базы данных, которая представляет собой спецификацию по созданию, чтению, изменению и удалению используемых в продукте данных.

В начале разработки схемы БД она сильно связана с логической моделью объектов. Схема определяет главные объекты-сущности, необходимые для будущего решения, их атрибуты и *связи* между этими сущностями. В большинстве

методов моделирования данных сущность определяется как абстрактное представление объекта реального мира. Вы научились создавать объекты-сущности и их атрибуты в процессе определения объектных моделей.

Примечание Окончательная схема данных не обязательно должна строго соответствовать логической модели объектов.

На рис. 8-1 показана часть схемы базы данных компании Adventure Works Cycles.

Как правило, объекты базы данных моделируются в диаграммах «сущность — связь» (entity relationship, ER). ER-диаграмма содержит сущности, атрибуты и связи и представляет собой высокоуровневое логическое представление данных. В ER-модели все данные рассматриваются как сформулированные факты о сущностях и их связях,

Примечание О сущностях, атрибутах и связях рассказывается далее в этой главе.

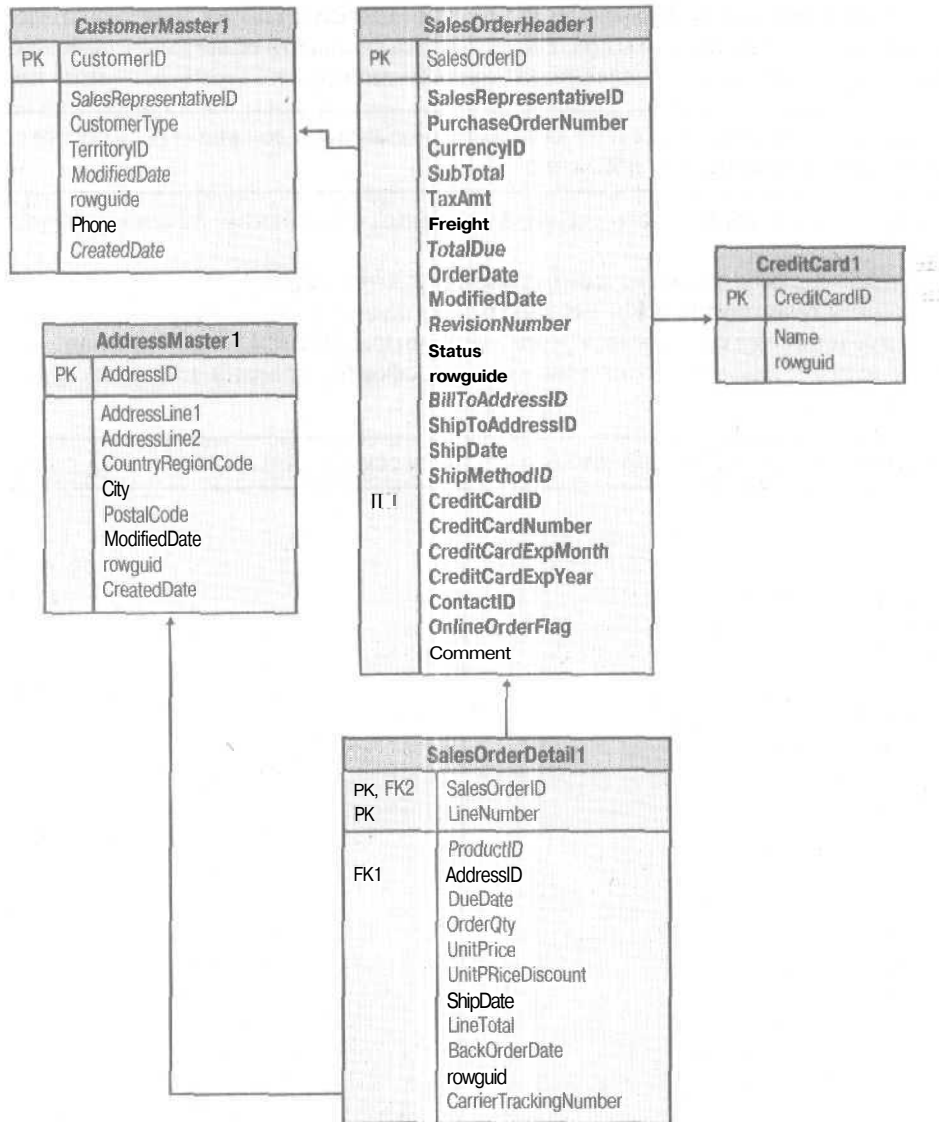


Рис. 8-1. Часть схемы БД информационной системы компании Adventure Works Cycles

Типы физических моделей данных

Помимо определения логического дизайна базы данных необходимо выбрать технологию физического хранения данных. Физическая модель данных системы управления базами данных (СУБД) определяет внутреннюю структуру, которая применяется в СУБД для управления данными. В этой структуре отражены типы таблиц базы данных, которые разрешается создавать, а также скорость доступа и универсальность базы данных. Далее перечислены наиболее распространенные типы физических моделей данных.

- **БД на плоских файлах, или неструктурированные БД.** В такой базе данных все данные располагаются в одном файле в виде набора строк и столбцов. В этой архитектуре отсутствует связь между различными плоскими файлами, поскольку ни одна из этих БД ничего не знает об остальных. Она поддерживает быстрое обновление и чтение данных за счет метода индексирования ISAM (indexed sequential access method — индексно-последовательный метод доступа). Технология ISAM применяется в унаследованных мэйнфреймовых БД и *небольших* базах данных, располагающихся на ПК.
- **Иерархические БД** способны хранить самую разнообразную информацию в самом различном формате. Их отличает расширяемость и гибкость, поэтому подобные БД используются, когда требования по хранению информации могут сильно отличаться или изменяться. Пример иерархической базы данных — сервер Microsoft Exchange, способный хранить самые различные типы информации в формате, который удовлетворяет требованиям *приложений* обмена сообщениями и поддержки совместной работы. Эти требования заключаются в возможности инкапсуляции в сообщениях самой разнородной информации.
- **Реляционные БД.** Здесь данные хранятся в наборе таблиц и столбцов. Реляционные базы данных сочетают преимущества плоских файлов и иерархических баз данных, обеспечивая хорошую производительность и гибкость в хранении данных. Благодаря возможности определять связи между таблицами на основании уникальных значений, реляционные базы данных — одни из самых популярных. Однако важно понимать, что другие модели также применяются и что разработчикам систем масштаба предприятия иногда приходится работать с несколькими типами баз данных одновременно. В реляционной модели основное внимание уделяется хранению, выборке и обеспечению целостности данных. Структурированный язык запросов SQL (Structured Query Language) позволяет эффективно извлекать связанные элементы данных вне зависимости от того, в одной или *нескольких* таблицах они хранятся. Целостность данных обеспечивается механизмом *правил* (rules) и *ограничений* (constraints).
- **Объектно-ориентированные БД.** Объекты базы данных являются объектами, описанными на одном (или нескольких) из популярных языков программирования. Объектно-ориентированные СУБД (ООСУБД) вносят в язык программирования «прозрачное» хранение данных, управление одновременным доступом, восстановление данных, ассоциативные запросы и другие *возможности* работы с информацией. ООСУБД применяются, когда необходимо *хранить* сложные данные и вместе с тем обеспечить высокую производительность. Хранение сложных данных характеризуется недостаточной точностью и уникальностью идентификации, связями вида «многие-ко-многим» и частым использованием кодов типов, как в реляционной модели.

В компании Adventure Works Cycles в настоящее время для хранения данных применяется Microsoft SQL Server 7.0. Проанализировав долгосрочные требования к хранилищу данных, проектная команда выбрала для создания продукта Microsoft SQL Server 9.0. В создаваемом приложении предполагается преобразовывать существующее хранилище данных с целью получения определенных преимуществ. Поэтому для информационной системы в компании Adventure Works Cycles предпочли реляционную модель. Физическая модель отражает целевую среду реализации.

Определение сущностей и атрибутов

В процессе логического дизайна проектная команда анализирует варианты и сценарии использования системы для определения объектов-сущностей и атрибутов. Сущности и атрибуты формируют базис логического дизайна и используются в процессе физического дизайна для моделирования физического дизайна будущего продукта. Логический дизайн позволяет гарантировать, что дизайн данных решения корректно отражает концептуальные требования. Однако реальная инфраструктура хранения данных оптимизируется для среды, в которой предполагается реализовать физическую модель данных.

Принципы определения сущностей

Определяя объекты-сущности в процессе логического дизайна данных помните, что объекты-сущности — это:

- объекты, информация о которых будет храниться. Примеры сущностей: «товар», «заказ» и «клиент»;
- отправная точка логического дизайна данных. Определение таких объектов-сущностей — первый шаг на пути к созданию дизайна базы данных;
- эквиваленты строк в одной или нескольких таблицах базы данных. Экземпляр объекта-сущности сопоставляется строке таблицы.

Рассмотрим следующие варианты использования системы:

Сотрудник заключает контракт с клиентом.

Сотрудник просматривает информацию о предыдущих расчетах с клиентом.

На основании этих ВИС можно определить следующие объекты:

- *Сотрудник* — тот, кто выполняет операции в системе;
- *Клиент* — адресат действий, выполняемых сотрудником;
- *Контракт* — соглашение между сотрудником и клиентом;
- *Счета* — *счета-фактуры*, которые были предъявлены клиенту за выполненную компанией работу.

Помимо того, что упомянутые сущности являются объектами в объектной модели решения, они также используются в логическом дизайне данных. В базе данных должна храниться информация о каждом экземпляре таких попадающих в область действия решения сущностей, как *Сотрудник*, *Клиент*, *Счет-фактура*. Помните, что описание объектов-сущностей помогает определить связи между ними.

Характеристики атрибутов

После определения сущностей необходимо определить атрибуты, которые будут храниться в базе данных. Необходимо знать, что атрибуты:

- описывают объект-сущность решения. Например, атрибуты автомобиля — это цвет, марка, модель и год выпуска. Размер, хотя и является характеристикой автомобиля, но не имеет отношения к создаваемому продукту и поэтому не выделяется в виде атрибута;
- не существуют сами по себе, а лишь в связи с сущностью. Например, такой атрибут, как цвет, сам по себе не описывает ничего материального, если только не применен к какому-либо объекту;
- определяют столбцы в таблицах базы данных. В процессе реализации физического дизайна атрибуты становятся **столбцами** в таблицах базы данных. Их полное описание состоит из типа, длины и любых применимых связей.

Например, сущность *Клиент* имеет такие атрибуты, как номер счета, имя и адрес. Необходимо хранить эту информацию для каждого экземпляра объекта-сущности *Клиент*.

Определение таблиц и столбцов

Результаты логического дизайна в процессе физического дизайна используются для создания компонентов, спецификаций пользовательского интерфейса и физического дизайна БД. Полученные в процессе логического дизайна объекты-сущности, атрибуты и ограничения преобразуются в таблицы, поля, связи и ограничения базы данных, которая таким образом становится физической реализацией логической модели.

Определение таблиц

Таблицы являются физическим представлением объектов-сущностей в реляционной базе данных. Они способны хранить самые разные данные — имена, адреса, изображения, аудио и видео файлы, документы Microsoft Word и т.п. Благодаря своей гибкости базы данных применяются для хранения не только простых текстовых данных, но и корпоративной базы знаний предприятия, независимо от формы этих знаний. База данных представляет связи между различными элементами данных.

Данные в таблице хранятся в виде *строк*, или записей, каждая из которых должна быть уникальной. Работать с записями в реляционной БД можно с применением языка XML (Extensible Markup Language). Его же часть используют для обмена данными между разными базами данных и даже предприятиями, не утруждая себя приведением к единой структуре баз данных участников обмена.

Традиционный формат взаимодействия с реляционными данными — ANSI-строки, а язык — SQL. Этот язык, напоминает английский и представляет операции, выполняемые с базой данных, в виде понятных человеку выражений, таких, как *Insert* (вставка), *Update* (обновление) и *Delete* (удаление). Большинство баз данных удовлетворяют ANSI-стандарту SQL, хотя его версии и расширения в разных системах отличаются.

Совет Таблицы можно привязывать к другим таблицами в рамках одного файла базы данных. Эта позволяет выполнять соединение (*join*) различных типов и выполнять нормализацию данных.

Определение столбцов

Данные в каждой таблице хранятся в *столбцах* (колонках), или полях, которые определяются на основании атрибутов объекта-сущности, представленной в виде таблицы. Каждое поле содержит различные элементы данных, например имя клиента.

Пример таблиц и столбцов

Один из объектов-сущностей системы компании Adventure Works Cycles, *SalesOrderDetail*, содержит детальную информацию о размещенном клиентом заказе. Атрибуты этого объекта-сущности — *SalesOrderID*, *ProductID*, *AddressID*, *UnitPrice* и *DueDate*. Поэтому база данных содержит таблицу *SalesOrderDetail*, которая состоит из столбцов *SalesOrderID*, *ProductID*, *AddressID*, *UnitPrice* и *DueDate*.

Назначение типов данных

Типы данных определяют, какого рода информация хранится в том или ином столбце. У каждого поля в базе данных есть свой тип, который позволяет вам и ядру базы данных *проверять*, насколько корректно значение, вводимое в поле, в контексте данного поля. Однако не забывайте, что корректный тип данных еще не гарантирует правильность самих данных. Например, поле целочисленного типа (*integer*) способно хранить числовые данные. Допустим, что бизнес-правила требуют, чтобы значения поля лежали в диапазоне 1—20. Если пользователь введет 25, то значение будет абсолютно правильным с точки зрения типа данных, однако ошибочным с точки зрения бизнеса.

Допустимые для конкретного поля типы данных зависят от типов данных, поддерживаемых самой СУБД. Определяя таблицы, выбирайте типы, которые позволят оптимизировать производительность, сэкономить дисковое пространство и расширить возможности наращивания системы. Большинство СУБД поддерживают два основных класса данных:

- **Системные типы данных.** Каждая СУБД содержит встроенные типы данных, например: *Integer*, *Character*, *Binary*. Некоторые СУБД поддерживают различные варианты этих типов, а также дополнительные типы.
- **Пользовательские типы данных.** Некоторые СУБД позволяют определять новые типы на основе системных. Например, в Microsoft SQL Server можно определить тип данных *state* (код штата) — поле длиной две единицы и содержащее тип *character*. Определение такого типа позволяет добиться единообразия во всех таблицах, содержащих поле с кодом штата. Все поля с типом *state* в любой таблице будут одинаковыми.

В таблице 8-1 показан стандартный набор типов, каждый из которых является разновидностью символьного, числового или двоичного типа.

Таблица 8-1. Стандартные типы данных

Тип данных	Описание
<i>Binary</i>	Бинарные данные фиксированной или переменной длины
<i>String</i>	Символьные данные фиксированной или переменной длины
<i>Date</i>	Данные в формате «дата — Время»
<i>Float</i>	Числовые данные с плавающей точкой (от $-1.79E+308$ до $1,79E+308$ в Microsoft SQL Server 2000)
<i>Decimal</i>	Числовые данные фиксированной точности и масштаба (от $-10^{70}+1$ до $10^{38}-1$ на SQL Server 2000)
<i>Integer</i>	Целочисленные данные. У различных СУБД разные варианты этого типа в верхними границами в 25 532 767 или 2 147 483 647
<i>(Long)Integer</i>	Целочисленные значения в более широком, чем <i>Integer</i> диапазоне
<i>Monetary</i>	Денежные единицы постоянного масштаба.
<i>(Double)Float</i>	Значения с плавающей точкой двойной точности на основе типа <i>Float</i> .

Типы также определяют способ отображения данных, например поля типа *Float*, *Money* и *integer* содержат числовые данные. Однако все эти типы хранятся, используются в вычислительных операциях и отображаются в различном формате. Данные хранятся в различных форматах, поэтому для их размещения требуется разный объем дискового пространства.

Примечание Типы вида *Double* способны хранить в два раза большие значения или мантиссу с большим количеством знаков, но они, как правило, занимают в два раза больше места в памяти.

В процессе физического дизайна следует внимательно проанализировать требования к каждому объекту данных и выбрать самый экономный тип данных, способный вместить все возможные значения того или иного объекта или атрибута. Все СУБД, включая Microsoft SQL Server, Oracle Database, Sybase Adaptive Server Enterprise, IBM DB2 и Informix, а также NCR Teradata Warehouse поддерживают собственные типы данных.

В таблице 8-2 показаны столбцы таблицы ProductMaster базы данных компании Adventure Works Cycles, размещаемые с использованием типов данных SQL.

Таблица 8-2. Таблица ProductMaster

Столбцы	Тип данных	Разрешается ли значение NULL
<i>ProductID</i>	<i>uniqueidentifier</i>	Нет
<i>Name</i>	<i>text</i>	Нет
<i>Product Number</i>	<i>text(25)</i>	Да
<i>DiscontinuedFlag</i>	<i>bit</i>	Да
<i>Make Flag</i>	<i>bit</i>	Нет
<i>StandardCost</i>	<i>money</i>	Да
<i>FinishedGoodsFlag</i>	<i>bit</i>	Нет
<i>Co/or</i>	<i>text(15)</i>	Да
<i>CreatedDate</i>	<i>datetime</i>	Нет
<i>ModifiedDate</i>	<i>datetime</i>	Да
<i>SafetyStockLevel</i>	<i>smallint</i>	Да
<i>Reorder Point</i>	<i>smallint</i>	Да
<i>ListPrice</i>	<i>money</i>	Да
<i>Size</i>	<i>text(50)</i>	Да
<i>Size UnitMeasureCode</i>	<i>char(3)</i>	Да
<i>Product Photo ID</i>	<i>int</i>	Да
<i>rowguid</i>	<i>LongBinary</i>	Да
<i>Weight UnitMeasureCode</i>	<i>char(3)</i>	Да
<i>Weight</i>	<i>float</i>	Да
<i>Days To Manufacture</i>	<i>int</i>	Да
<i>ProductLine</i>	<i>char(2)</i>	Да
<i>DealerPrice</i>	<i>money</i>	Да
<i>Class</i>	<i>char(2)</i>	Да
<i>Style</i>	<i>char(2)</i>	Да
<i>Product DescriptionID</i>	<i>int</i>	Да
<i>ProductSubCategoryID</i>	<i>smallint</i>	Да
<i>ProductModelID</i>	<i>int</i>	Да

Типы ключей

Ключи — важная составная часть реляционных БД. Они уникально определяют каждый экземпляр объекта-сущности в модели данных, а также обеспечивают механизм связи сущностей между собой. Реляционные базы поддерживают ключи различных типов.

- **Основные ключи** (primary keys) уникально определяют каждую строку данных в таблице. В качестве основного ключа обычно используют атрибут, уникальный для каждого экземпляра объекта-сущности. Например, *SalesOrderID* (идентификатор заказа) уникален для каждого заказа и является основным ключом таблицы *SalesOrderHeader*. В некоторых случаях необходимо создать искусственные атрибуты, уникально определяющие сущность. Большинство СУБД предоставляют различные механизмы для создания основных ключей, в том числе автоматическую генерацию уникальных идентификаторов и возможность создания *составных ключей* (composite keys), например состоящих из двух столбцов (*SalesOrderID* и *LineNumber* в таблице *SalesOrderDetail*).

Примечание Другой распространенный способ — использование *интеллектуальных ключей* (smart keys), то есть связанных с предметной областью, например BLDG001 идентифицирует записи в таблице, хранящей информацию о зданиях (building).

-
- **Внешние ключи** служат для создания связей между таблицами. Например, *ProductID* — внешний ключ таблицы *SalesOrderDetail* и одновременно основной ключ таблицы *ProductMaster*. Таким образом, он связывает таблицы *SalesOrderDetail* и *ProductMaster* и позволяет избежать хранения дублирующихся данных о товаре (например, названия и описания) в таблице *SalesOrderDetail*.

Примечание Полную схему базы данных компании Adventure Works Cycles вы найдете в файле *Adventure Works Cycles Data Schema.vsd* в папке *\SolutionDocuments\Chapter08* на прилагаемом к книге компакт-диске.

Организация связей

Возможны самые различные типы связей между таблицами БД. Точно так же, как объекты-сущности и атрибуты определяются в процессе логического дизайна и представляются в виде таблиц и столбцов в физическом дизайне, связи, определенные в процессе логического дизайна, должны представляться в базе данных на стадии физического дизайна.

В физическом дизайне базы данных связи между сущностями определяются созданием ключей, которые связывают таблицы, хранящие объекты различных типов. Существует несколько типов связей:

- «один-к-одному»;
- «один-ко-многим»;
- «многие-ко-многим».

Связи вида «один-к-одному»

В данном случае экземпляр одного объекта-сущности напрямую связан с одним экземпляром другого объекта-сущности. Например, каждую кафедру возглавля-

ет только один профессор и каждый профессор может возглавлять не более одной кафедры. Если в связи обязательно должны присутствовать оба объекта, то такой случай представляется одним из трех способов:

- **в виде одной таблицы.** Оба объекта совмещаются в одной таблице, а основной ключ объединенной таблиц используется в качестве составного ключа. Преимущество данного подхода — в отсутствии необходимости создания и поддержки отдельных таблиц. Этот способ избавляет синтаксический анализатор запросов от необходимости выполнения соединения (*join*), а также позволяет сэкономить дисковое пространство. Недостаток — если когда-либо в будущем связь изменится, решение задачи станет в копеечку;
- **в виде двух таблиц.** Каждый тип объектов хранится в собственной таблице, и основной ключ одного объекта добавляется в качестве внешнего ключа другого объекта. Часто это самый наглядный способ установления отношений вида «родитель — наследник» между объектами-сущностями. В таких ситуациях основной ключ объекта-родителя определяется как внешний ключ дочернего объекта, поскольку последний существует лишь при условии существования родительского объекта. Такой подход гарантирует уникальность значений каждого ключевого поля и позволяет проверить, каждый ли экземпляр объекта одного типа связан строго с одним экземпляром объекта другого типа;
- **в виде нескольких таблиц.** Если связь между объектами не обязательная и родительский объект может существовать в отсутствие соответствующей «дочки», лучше создать отдельные таблицы для каждого объекта и связи установить посредством внешних ключей. В качестве примера возьмем организацию, предоставляющую своим сотрудникам автомобили и оформляющую страховку на машину и водителя. Возможна ситуация, когда сотруднику не выделена машина или, наоборот, машина ни за кем не закреплена. Существование страховки также не гарантировано. Следовательно, между перечисленными тремя сущностями не обязательно существует связь. Можно создать три таблицы и еще одну, содержащую ключи из всех трех таблиц, или определить по два внешних ключа в каждой таблице.

На рис. 8-2 показана связь «один-к-одному» между двумя таблицами.

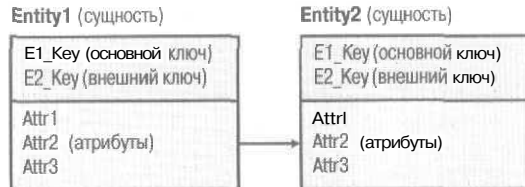


Рис. 8-2. Связь типа «один-к-одному»

Связи вида «один-ко-многим»

Физический дизайн связи «один-ко-многим» подразумевает наличие таблицы родительских объектов, например «клиенты», и таблицы дочерних объектов, например «заказы», в которой каждому родительскому объекту соответствуют несколько дочерних. В этом случае необходимы внешние ключи в дочернем объекте, определяющие связь. Установление связи подтверждает, что внешний ключ ссылается на корректный родительский объект.

Примечание Связь вида «один-ко-многим» довольно часто применяется при проектировании БД, поскольку позволяет эффективно использовать дисковое пространство.

Рис. 8-3. иллюстрирует связь «один-ко-многим» между двумя таблицами.

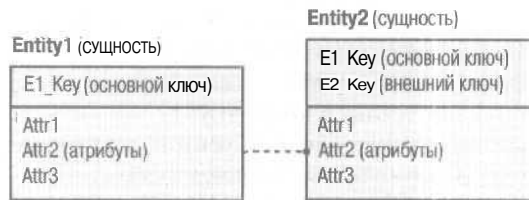


Рис. 8-3. Связь «один-ко-многим»

Связи вида «многие-ко-многим»

Большинство реляционных СУБД (в том числе SQL Server) не поддерживают напрямую представление связей вида «многие-ко-многим» иначе как с использованием денормализации (о которой вы узнаете на занятии 2). Большинство СУБД позволяют обойти эту проблему путем создания *таблицы соединения* (join table).

На рис. 8-4 между объектами Employee (Сотрудник) и Client (Клиент) существует связь «многие-ко-многим». Один сотрудник может заключать договоры со многими клиентами, точно так же каждый клиент имеет право заключать договоры с разными сотрудниками. Поскольку эту связь нельзя выразить напрямую, основной ключ каждого объекта используется в качестве внешнего ключа отдельной таблицы *Contracts* (договоры). Эта пара внешних ключей уникально определяет связь между таблицами *Employee* и *Client*.

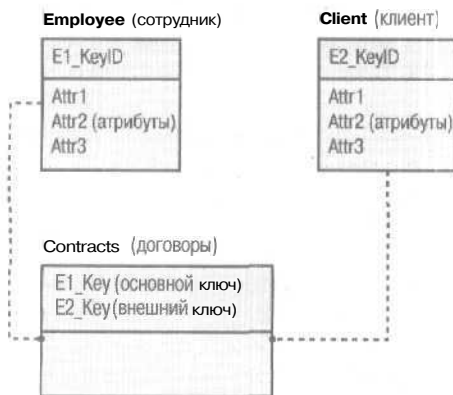


Рис. 8-4. Связь вида «многие-ко-многим»

Занятие 2. Оптимизация доступа к данным

Цель оптимизации заключается в сокращении времени отклика на запросы и **максимизация** пропускной способности сервера БД путем снижения сетевого трафика, потребления процессорного времени и сокращения числа операций дискового ввода-вывода. Для этого тщательно анализируют требования к приложению, логическую и физическую структуру данных и компромисс между конфликтующими операциями в базе данных, например между большим количеством **операций** вставки (интенсивная запись) и запросов на чтение.

На этом занятии перечислены общие рекомендации по оптимизации доступа к данным и рассказывается о методах решения этой задачи, в том числе об индексации, разбиении на разделы, нормализации и **денормализации**.

Изучив материал этого занятия, вы сможете:

- ✓ рассказать о методах оптимизации базы данных;
- ✓ описать, как индексирование сказывается на работе БД;
- ✓ описать цель разбиения данных на разделы;
- ✓ описать цель **нормализации** данных.

Продолжительность занятия — около 10 минут.

Методы оптимизации доступа к данным

Помнить о производительности необходимо на протяжении всего цикла проектирования, не откладывая «на потом», когда система уже реализована. Быстрота доступа к данным влияет на **общую** производительность приложения. Можно добиться значительного улучшения производительности, с самого начала **оптимизируя** как дизайн приложения, так и дизайн базы данных.

Оптимизация приложения

При разработке приложения, работающего с данными, следуйте следующим рекомендациям по программированию:

- сокращайте до минимума количество запросов, **возвращающих** наборы данных;
- минимизируйте размер наборов данных, стараясь включать в него меньше данных;
- избегайте *параллелизма* (concurrency), когда несколько пользователей одновременно редактируют одну и ту же запись, и позаботьтесь об эффективном механизме разрешения конфликтов, возникающих в подобных ситуациях;
- тщательно продумайте распределение нагрузки по обработке наборов данных между клиентом и сервером, особенно если речь идет о **Web-приложениях**.

Нагрузочное тестирование приложения

Существует только один способ выяснить, как поведет себя приложение, — протестировать его работу под нагрузкой с **помощью** специального инструментального средства, например Microsoft Web Application Stress. Без этого очень трудно обнаружить узкие места приложения.

Разумное использование транзакций

Транзакции должны быть как можно короче и применять их следует только там, где они действительно необходимы. Распределенные транзакции создают значительную дополнительную нагрузку, что моментально отрицательно сказывается на производительности приложения. Используйте их только тогда, когда без них действительно не обойтись.

Эффективная организация обмена информацией

Любое взаимодействие с пересечением границ между приложениями или процессами бьет по производительности. Частичное решение проблемы заключается в сокращении обмена данными между приложениями и потоками. Например, намного эффективнее вызвать метод, указав шесть аргументов, чем по отдельности указать шесть параметров объекта и лишь затем вызвать метод. Подобный подход к проектированию побуждает к исключению поддержки состояний из уровня бизнес-логики.

Оптимизация базы данных

Самые популярные методы:

- индексирование;
- разбиение на разделы;
- нормализация,

Примечание Подробнее об этих методах — далее в этой главе.

Для искоренения узких мест при чтении и записи в БД рекомендуется:

- определить возможные индексы, но не увлекаться чрезмерно индексированием;
- при использовании Microsoft SQL Server применять профайлер SQL Server Profiler и мастер настройки индексов Index Tuning Wizard;
- отслеживать загрузку процессора. Оптимальный уровень — примерно 75–80% процессорного времени;
- анализировать план выполнения запросов средствами Query Analyzer — это необходимо для оптимизации запросов;
- для максимальной производительности использовать хранимые процедуры;
- выполнять нормализацию данных, если часто выполняется запись;
- выполнять денормализацию, если часто осуществляется чтение.

Индексирование данных

Систему необходимо оптимизировать как для доступа, так и для обновления данных, и самый популярный способ решения этой задачи — индексирование. *Индекс* — это упорядоченный список строк в таблице, который позволяет СУБД ускорить операции поиска.

Назначение индекса

Структура индекса подобна дереву и содержит отсортированный список определенных значений данных. Запросы индексированных данных выполняются гораздо быстрее. Вместо того чтобы каждый раз сканировать всю таблицу в поисках нужного значения СУБД использует индекс для быстрого нахождения мес-

тоположения требуемых данных, поскольку индекс хранит указатель на данные в таблице.

Преимущества индексирования

- **Более быстрый доступ к данным.** Индексы в базе данных сродни индексу в книге, который позволяет быстро найти нужную информацию, не читая книгу целиком. В базе данных индекс позволяет СУБД находить данные без сканирования всей таблицы.
- **Целостность данных.** Некоторые СУБД применяют индексы для обеспечения уникальности каждой строки в таблице.

Типы индексов

Для оптимизации выборки данных используют два типа индексов.

- **Кластеризованный индекс.** В этом случае строки в таблице физически представляются в особом порядке. Этот тип индекса обеспечивает очень высокую производительность чтения. Кластеризованный индекс, как правило, определяется как основной ключ таблицы. (Большинство СУБД разрешает только один кластерный индекс на таблицу.) Например, в таблице *SalesOrderDetail* роль кластеризованного индекса выполняют столбцы *SalesOrderID* и *LineNumber*. Одно из ограничений кластеризованных индексов заключается в их способности замедлять операции записи, поскольку запись часто инициирует физическое переупорядочивание строк, на что отвлекаются значительные ресурсы.
- **Некластеризованный индекс** — это небольшая таблица с индексной информацией о столбце или группе столбцов. У таблицы может быть несколько некластеризованных индексов.

Разбиение данных на разделы

Со временем количество записей в таблице достигает уровня, когда методы оптимизации больше не позволяют ускорить доступ к данным. В таких ситуациях выполняют *разбиение таблиц на разделы (partitioning)*. Существует два вида разбиения: горизонтальное или вертикальное.

Горизонтальное разбиение

В случае *горизонтального разбиения (horizontal partitioning)* таблица с большим количеством записей «режется» на несколько таблиц с одинаковым набором столбцов, то есть каждая из подтаблиц содержит подмножество исходного набора данных. Например, в одной подтаблице содержатся клиенты с фамилиями, начинающимися на буквы с А по О, а в другой — с П по Я.

Вертикальное разбиение

Вертикальное разбиение (vertical partitioning) предусматривает разбиение таблицы с большим количеством столбцов на несколько разделов (подтаблиц) со строками, обладающими одинаковыми уникальными идентификаторами. Например, одна подтаблица содержит данные только для чтения, а другая — обновляемые данные. В средах с большим объемом данных разделы обычно разносят на различные серверы, чтобы обеспечивает дополнительное распределение нагрузки.

Нормализация данных

Нормализация — это процесс совершенствования логической модели для устранения дублирования данных. Нормализация предусматривает разбиение базы данных на несколько таблиц и определение связей между ними.

Теоретики баз данных разработали стандарты последовательно усиливающих ограничений, или *нормальных форм* БД. Применение нормальных форм дает в результате нормализованную базу данных. Существует по крайней мере пять широко известных стандартных уровней нормальных форм, причем каждый последующий налагает все более жесткие ограничения на дублирование данных.

Совет В реальной жизни чаще всего используется третья нормальная форма — компромисс между недо- и перенормализацией.

Преимущества нормализации

Нормализованные базы данных как правило содержат больше таблиц с меньшим количеством столбцов по сравнению с ненормализованными базами данных. Нормализация базы данных дает определенные результаты.

- **Сокращение дублирования информации.** Нормализованная база данных содержит меньше дублирующей информации по сравнению с ненормализованной. Это актуально, когда, например, необходимо хранить в базе данных информацию о табелях и счетах-фактурах. Хранение информации о табелях в таблице *Invoice* (Счет-фактура) приведет к появлению в ней большого количества повторяющихся данных: информация о *сотруднике*, проделанной работе, задачах и клиенте. Нормализация базы данных приводит к появлению отдельных связанных таблиц для хранения информации о табелях и счетах-фактурах, позволяя тем самым избежать дублирования.
- **Сокращение числа случаев нарушения целостности данных.** Нормализация снижает вероятность нарушения целостности данных благодаря наличию связей между таблицами. Например, если телефонный номер клиента хранится в нескольких таблицах или в нескольких строках одной таблицы, то может случиться так, что после его изменения он обновится не везде в базе данных. Если же номер хранится только в одной строке одной таблицы, шансы нарушить целостность данных значительно снижаются.
- **Более быстрое обновление (изменение) данных, включая операции вставки, обновления и удаления.** Нормализация ускоряет модификацию данных, например вывод имени и адреса клиента из таблицы *Invoice* приводит к уменьшению количества данных для обработки при работе со счетами-фактурами. Выведенные данные не теряются — они дублируются в таблице *Client*. Кроме того, уменьшение объема дублированной информации увеличивает скорость операций обновления, поскольку приходится модифицировать меньше значений данных в таблицах.

Примечание Нормализация иногда снижает быстродействие операций записи из-за конфликта блокировок и возможного увеличения размера строки после записи.

Первая нормальная форма

Первая стадия нормализации базы данных — позаботиться, чтобы таблицы находились в первой нормальной форме, то есть они должны удовлетворять следующим критериям:

- таблицы должны быть двумерными, а сами данные — организованы в виде столбцов и строк. **Объекты-сущности**, определенные в логической модели данных, преобразуются в двумерные таблицы базы данных;
- каждая **ячейка** должна содержать не более одного значения;
- каждый столбец должен содержать значения, одинаковые по смыслу, например недопустим смешанный столбец *Order Date/Delivery Dare* (дата заказа/ дата доставки).

На рис. 8-5 показана таблица *Timesheet* (табель рабочего времени), которая сначала не удовлетворяла требованиям первой нормальной формы, так как не имела уникального идентификатора и не все ее атрибуты однозначно относились к одному элементу информации. Кроме того, каждый ее столбец мог использоваться для хранения разнообразных по смыслу значений. В первой нормальной форме атрибут *Employee* (Сотрудник) разбит на два отдельных атрибута: *Employee FirstName* (Имя) и *Employee LastName* (фамилия).

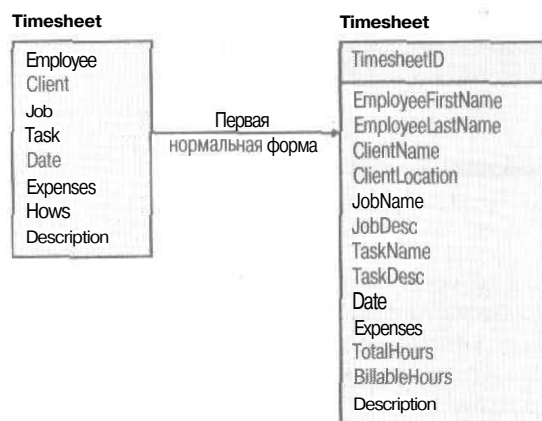


Рис. 8-5. Первая нормальная форма таблицы *Timesheet*

Вторая нормальная форма

Вторая нормальная форма — промежуточная на пути к третьей нормальной форме. Чтобы организовать данные во второй нормальной форме, необходимо найти повторяющиеся данные в атрибутах нескольких экземпляров сущности и вынести их в отдельную таблицу.

- Устраните повторяющиеся данные в объекте-сущности.
- Перенесите в отдельную таблицу атрибуты, зависящие только от части составного ключа.
- Объедините информацию, где это возможно.

На рис. 8-6 показана таблица *Timesheet*. Если клиент переезжает в другой город, база данных клиентов должны обновляться, так же, как и все таблицы, содержащие ссылки на этого клиента. Решение состоит в удалении информации о клиенте из таблицы *Timesheet* и замене ее на внешний ключ *ClientID*, ссылаю-

щийся на основной ключ *ClientID* таблицы *Client*. Если адрес клиента изменится, изменения затронут только таблицу *Client*. Точно также информация о сотруднике заменяется на внешний ключ *EmployeeID*, который связывает таблицу с конкретным сотрудником, который ввел соответствующую информацию. Во второй нормальной форме также устранена любая другая дублирующаяся информация. Атрибуты *JobName* (наименование должности) и *JobDesc* (описание должности) заменены на один атрибут *JobDesc*, поскольку он скорее всего содержит и наименование должности. Те же операции проведены с атрибутами *TaskName* и *TaskDesc*.

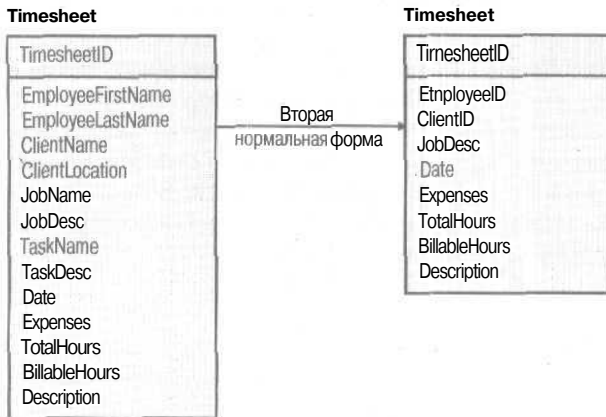


Рис. 8-6. Вторая нормальная форма таблицы *Timesheet*

Третья нормальная форма

Третья нормальная форма — уровень, к которому большинство команд старается довести дизайн данных. Здесь устраняются столбцы, существование которых не зависит от ключевого значения. Любые данные, не связанные напрямую с объектом-сущностью, как правило, переносятся в другую таблицу. Третьей нормальной формы в большинстве случаев вполне достаточно для целей создания решения:

- уберите все столбцы, существование которых не зависит от ключевого значения;
- в большинстве случаев стоит перенести данные, не связанные напрямую с объектом-сущностью, в отдельную таблицу;
- сократите или полностью устраните аномалии, возникающие при обновлении или удалении;
- убедитесь, что не осталось повторяющихся данных.

Третья нормальная форма помогает избежать аномалий обновления и удаления, поскольку все данные можно получить на основании внешнего ключа, а повторяющихся данных в рамках одной таблицы больше не существует. Этот уровень нормализации сильно повышает надежность базы данных, а дизайн становится намного эффективнее.

На рис. 8-7 показан приведенный к третьей нормальной форме объект-сущность *Timesheet*. Все, не зависящие от основного ключа атрибуты перемещены в отдельные таблицы, а их место заняли внешние ключи. Как и во второй нормальной форме, в третьей более эффективно устранены аномалии обновления и

удаления, поскольку все данные напрямую доступны из одной точки и не хранятся в разных частях БД.

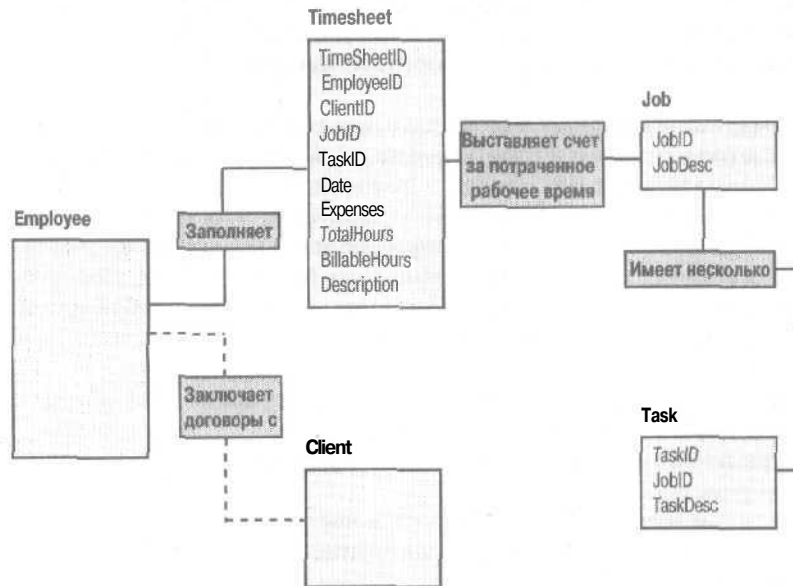


Рис. 8-7. Третья нормальная форма таблицы *Timesheet*

Денормализация таблиц

Ориентированная на обновление оптимизация физического дизайна данных весьма популярна. В большинстве систем данные изменяются довольно часто, поэтому важно оптимизировать систему для выполнения подобных операций.

Дизайн базы данных, основанный исключительно на правилах нормализации, приводит к появлению массы небольших таблиц. В этом случае снижается уровень дублирования данных и требуется меньше места на диске. Но для выборки необходимы сложные соединения (*join*).

Денормализация — это обратный по отношению к нормализации процесс, цель которого — создание таблиц с большим количеством полей, позволяющих сократить количество соединений в запросах. В зависимости от типа запросов и операций обновления, выполняющихся в базе данных, Денормализация позволяет значительно увеличить общую производительность. Поэтому она считается неплохим методом оптимизации несмотря на то, что многие СУБД содержат оптимизатор, позволяющий эффективно выполнять соединения.

Решая, стоит ли прибегать к денормализации, следует знать ее особенности.

- **Увеличивается дублирование данных.** Связанные данные перемещаются в одну таблицу, что позволяет ускорить выполнение запросов и уменьшить время обращения к диску.
- **Уменьшается количество таблиц, что упрощает программирование запросов и администрирование.** Недостаток — уменьшается количество мест хранения данных, в результате чего возрастает количество полей в таблицах и, возможно, некоторые данные дублируются в разных таблицах.

- **Данные, хранящиеся более, чем в одном месте, подвержены рассинхронизации,** когда обновляется только одна таблица, а не все.

Выбирая между высоконормализованным и разумно денормализованным физическим дизайном данных, следует учитывать, что стратегия выборочной и разумной денормализации часто позволяет добиться максимальной производительности.

Излишняя нормализация данных в определенных типах приложений вызывает проблемы с быстродействием и отрицательно влияет на скорость операций чтения и записи. Нормализация ER-модели — важный шаг на пути к созданию логического дизайна БД. Цель нормализации — добиться, чтобы существовал только один способ получить ту или иную информацию. Она устраняет дублирование данных и обеспечивает хранение корректных бизнес-правил. Логический дизайн базы данных не зависит от реализации, то есть от особенностей конкретной СУБД. Однако здесь подразумевается использование реляционной модели данных.

В процессе физического дизайна команда, проектирующая базу данных, должна учитывать многие факторы:

- физические ограничения базы данных, такие, как требования к памяти и дисковому пространству;
- возможности тонкой настройки производительности, такие, как определение взаимоблокировок (deadlocks), индексирование и узкие места;
- основные и внешние ключи;
- структуру триггеров;
- принципы создания хранимых процедур;
- нормализацию. Хотя третья нормальная форма часто удовлетворяет всем условиям, в некоторых случаях нелишне подумать о реализации четвертой или даже пятой нормальной формы;
- соображения объектной модели приложения. Для нормальной работы приложения могут потребоваться некоторые, не отраженные в логической модели данных таблицы;
- перенос данных из унаследованных систем баз данных;
- качество существующих данных;
- процедурные соображения, такие, как кластеризация и отказоустойчивость, резервное копирование и процессы обновления и развертывания.

Занятие 3. Проверка данных

Помимо обеспечения нормальной производительности базы данных очень важно качество хранимых данных. Это означает, что БД должна поддерживать целостность данных и хранить данные, удовлетворяющие бизнес-правилам.

На этом занятии рассказывается о требованиях к целостности данных решения, а также об определении бизнес-правил и реализации их в базе данных и на бизнес-уровне.

Изучив материал этого занятия, вы сможете:

- ✓ дать определение целостности данных;
- ✓ определить требования к целостности данных;
- ✓ определить бизнес-правила;
- ✓ реализовать бизнес-правила в базе данных;
- ✓ реализовать бизнес-правила в виде компонентов.

Продолжительность занятия — около 15 минут.

Целостность данных

Целостность данных означает их непротиворечивость и точность. Важный шаг планирования БД — принятие решения о способе обеспечения ее целостности. Существует три типа целостности данных: предметная, сущностная и ссылочная.

Предметная целостность

Этот вид предусматривает целостность с точки зрения предметной области, то есть определение набора разрешенных значений данных для столбца и допустимость значения NULL. Предметная целостность реализована с помощью проверок данных и ограничений, таких, как формат или диапазон возможных значений столбца. Например, в соответствии с бизнес-правилами скидка составляет 2—18%, кроме того она не может быть нулевой, поскольку это чревато ошибками в расчетах.

Целостность объекта-сущности

Целостность объекта-сущности требует, чтобы каждая строка в таблице имела уникальный идентификатор в поле основного ключа. Можно ли это значение менять и можно ли удалять строку целиком, зависит от уровня ссылочной целостности, которая задана между основным ключом одной таблицы и внешними ключами любых других таблиц. Например, у записи о клиенте должен быть основным ключ. Если обстоятельства потребуют изменить значение ключа (например, из-за слияния компаний) или сделать строку неактивной, не удаляя ее физически, решения о целостности сущности и ссылочной целостности реализуются в виде бизнес-правила.

В совокупности предметная и сущностная целостность помогают обеспечить *согласованность* и целостность всех объектов-сущностей логического дизайна. При планировании базы данных в проекте непротиворечивость объ-

- **Условия, которых необходимо избегать.** Например, цену товара нельзя уменьшать ниже определенного уровня или количество запасов товара на складе не может быть отрицательным.
 - **Порядок событий.** Например, необходимо решить, что делать, когда объем запасов товара упадет до отрицательной величины или когда пришел платеж за заказ.
- Бизнес-правила и требования к целостности данных реализуются:
- **непосредственно в базе данных.** Большинство СУБД поддерживают автоматические процедуры, которые привязываются к таблицам и столбцам и обеспечивают соблюдение бизнес-правил и требований по целостности данных;
 - **программно вне базы данных.** Бизнес-правила и требования по целостности данных часто реализуют в клиентском приложении, создавая компоненты, разворачиваемые на сервере приложений или используя языки программирования, поддерживаемые ядром БД.

Основное различие между этими методами в том, что в первом случае правила автоматически выполняются ядром БД, а во втором необходимо явно вызывать методы проверки в приложении.

Решение о месте реализации правила часто основывается на самой природе требуемой логики программирования. Большинство реляционных БД способны автоматически выполнять методы, содержащие простую и наиболее часто применяемую прикладную логику. Эта логика относится к определенным задачам бизнес-правил и сконцентрирована в одном месте. Ее проще обновлять, и работает она намного эффективнее. Если требуется более сложная логика, ее выводят в код приложения — на любой из трех уровней сервисов: пользовательский, бизнеса или данных.

Реализация бизнес-правил в базе данных

При реализации бизнес-правила непосредственно в базе данных СУБД берет на себя большую часть работы, используя автоматические и встроенные проверки.

Бизнес-правила и требования к целостности данных в ядре БД обеспечиваются применением набора критериев, которым данные должны или, наоборот, не должны удовлетворять. Обычно большинство требований к целостности данных реализуются с помощью автоматического контроля и свойств ядра БД. Автоматический контроль представляет собой спецификацию или объявление критерия в момент создания объекта (поля, таблицы, индекса или ключа).

Возможности базы данных по реализации бизнес-правил

Обеспечение целостности данных с использованием встроенных в базу данных функций имеет ряд четких преимуществ: целостность обеспечивается автоматически, а критерии не нуждаются в специальной поддержке или обновлении, исключением являются ситуации, когда тот или иной критерий необходимо обновить. Обычно применяют следующие свойства баз данных.

- **Типы данных.** Назначение полям в базе данных соответствующих типов данных гарантирует, что в таблицу не попадут значения «не того» типа. Например, при назначении полю типа Date (дата) БД не позволит записать в него любые строковые или числовые данные не в формате даты. Типы данных гарантируют корректность формата, но не в состоянии обеспечить правильность значений.

- **Значения по умолчанию.** Они записываются в поле, если не указаны явно в команде INSERT. Значения по умолчанию позволяют обеспечить заполнение поля корректным значением, даже если его явно и не указали.
- **Правила проверки данных** инкапсулируют логику проверки допустимости значений, разрешенных в том или ином поле таблицы. Они определяют длину поля, маску ввода или диапазон допустимых значений, а также применяются для простых сравнений.
- **Ключи.** Большинство баз данных автоматически отслеживают ссылочную целостность таблиц. Основные и внешние ключи из физической модели напрямую соответствуют конфигурации БД, которая автоматически обеспечивает ссылочную целостность между связанными таблицами.
- **Триггеры** — это совокупность программных конструкций, явно заданных для определенной таблицы. При выполнении определенного действия (например, при вставке, обновлении или удалении) триггер автоматически запускает соответствующий код на исполнение. Например, позволяет выполнить проверку данных в других таблицах или автоматически обновить другую таблицу.

Программная реализация бизнес-правил

Анализ обычно показывает, что одни бизнес-правила лучше всего реализовать на бизнес-уровне, а другие — на уровне данных. Для программной реализации бизнес-правил на уровне данных существует ряд средств.

- **Хранимые процедуры** — это именованные наборы SQL-выражений в СУБД. Они хранятся в скомпилированном виде, поэтому анализатору запросов не приходится при каждом вызове выполнять синтаксический разбор. Хранимые процедуры позволяют написать подпрограмму, выполняющую определенные действия и сохраняющую результаты, и вызывать ее по мере необходимости.

Хранимые процедуры удобно использовать для контроля внесения изменений в таблицу. Например, вместо того чтобы давать пользователю право доступа к определенной таблице, можно разрешить ему обновление данных только через хранимые процедуры, в которых предусмотреть проверку корректности изменений с последующей записью правильных данных и отклонением ошибочных.

Хранимые процедуры также позволяют управлять ходом процесса, например изменить или переупорядочить несколько таблиц. А также применяются для задач администрирования.

- **Сценарии (scripts).** как правило, создаются для автоматизации процессов, которые неудобно или неэффективно поручать ядру БД. Например, вы вправе создать сценарий для сопровождения базы данных, который импортирует определенный массив данных с мейнфрейма, выполняет индексацию БД и копирует отчет на файловый сервер.

Сценарии пишутся на самых различных языках программирования. Каждый сценарий выполняется в отдельном от СУБД адресном пространстве и кроме манипулирования данными способен выполнять другие задачи. Сценарии в основном популярны в средах с поддержкой командной строки или применяются в составе пакетных процессов, выполняющих другие задачи.

Реализация проверки данных в компонентах

В многоуровневом приложении сервис данных работает как посредник между бизнес-сервисами приложения и хранилищем данных, поэтому при внесении изменений в хранилище не обязательно изменять бизнес-сервисы. Сервисы данных выполняют базовые задачи: создание, выборку, обновление и удаление данных, а также используются для работы с данными в базе и обеспечения их целостности. Вы вправе спроектировать и разработать компоненты, реализующие сервисы данных в приложении.

Компоненты — это исполняемый код, располагающийся на сервере или внутри другой программы. Компоненты сродни сценариям, но они более тесно интегрированы со средой разработки, ядром баз данных и другим кодом.

Целостность данных иногда обеспечивают посредством логики, инкапсулированной в компоненте и вызываемой по мере необходимости. Критерии логики определяются **бизнес-правилами**, а также любыми дополнительными требованиями к целостности данных, определенными для решения.

Компоненты, как правило, разворачиваются на отдельном сервере приложений. Вот некоторые из преимуществ развертывания компонентов на сервере приложений:

- **простота сопровождения.** Код гораздо проще поддерживать, если он хранится в одном месте (или в небольшом количестве мест) и включен в состав самого приложения. Увеличение затрат на приобретение сервера приложений компенсируется улучшенными возможностями по поддержке системы. Любые необходимые изменения вносятся только на одном или небольшом количестве компьютеров, что очень мало по сравнению с числом клиентов, которых может быть несколько тысяч;
- **масштабируемость.** По мере возрастания нагрузки на систему достаточно просто добавить дополнительные серверы приложений и распределить нагрузку между ними.

Реализация бизнес-правил на основе компонентов подчас увеличивает совокупную стоимость решения. Дополнительные затраты состоят не только из дополнительного аппаратного и программного обеспечения для сервера приложений, но из стоимости разработки и поддержки кода обработки запросов данных.

Практикум. Создание схемы данных



При выполнении упражнений используйте знания, полученные на занятиях.

Упражнение. Создание схемы данных

Решение создается для упомянутой в предыдущих главах компании Adventure Works Cycles. Совершая покупку, клиенты компании заполняют форму «Заказ товара». Условия обработки заказов таковы:

- перед началом заполнения формы клиент вводит порядковый номер заказа;
- в заказ разрешается в дальнейшем вносить изменения;
- скидки применяются к каждой по отдельности позиции заказа.

Ваша задача — создать проект схемы данных для сбора всей информации в форме «Заказ товара». Схема данных должна содержать:

- одну или несколько таблиц для хранения данных о заказах;
- основные и внешние ключи;
- типы данных в каждом столбце;
- индексы;
- поля, необходимые в соответствии с требованиями бизнеса и данных.

В качестве отправной точки используйте схему данных, расположенную на вкладке P08 Schema Starter документа *C08Ex1.vsd* в папке *\SolutionDocuments\Chapter08* на прилагаемом к книге компакт-диске. Для просмотра формы «Заказ товара» откройте файл *C08Ex1.doc* в той же папке.

Одно из возможных решений вы найдете в файле *C08Ex1_ShortAnswer.vsd*. Вы также вправе спроектировать подробную схему данных, подобную показанной в файле *C08Ex1_LongAnswer.vsd*. Полную схему данных информационной системы компании Adventure Works Cycles вы найдете в файле *Adventure Works Cycles Data Schema.vsd*.

Резюме

- На этапе планирования проектная команда анализирует требования к данным и определяет их структуру, способ хранения, доступа и проверки.
- Схема базы данных отражает **объекты-сущности** предметной области, их атрибуты и связи между объектами.
- Физическая модель данных СУБД определяет внутреннюю структуру хранения данных.
- Наиболее популярны три типа физической модели данных: плоский файл, иерархическая и реляционная модели.
- Определенные в процессе логического дизайна **объекты-сущности** и атрибуты преобразуются в таблицы, поля и связи в базе данных.
- Таблица — это физическое представление объектов-сущностей в реляционной базе данных.
- Столбец — это физическое представление атрибута в реляционной базе данных.
- Типы данных определяют вид и отображение данных, хранящихся в поле БД.
- Ключи уникально идентифицируют экземпляры объекта-сущности в модели данных.
- Основные ключи уникально идентифицируют строки данных в **таблице**.
- Внешние ключи связывают две таблицы базы данных.
- Связи между объектами-сущностями реализуются путем создания ключей, **связывающих** таблицы между собой.
- Для повышения производительности базы данных необходимо:
 - оптимизировать базу данных;
 - выполнить нагрузочное тестирование приложения;
 - тщательно продумать использование транзакций;
 - сократить и **оптимизировать** передачу информации через границы процессов и приложений.
- Индекс — это **упорядоченный** список строк таблицы, используемый СУБД для ускорения операций поиска.
- Скорость обработки **очень** больших таблиц оптимизируют путем вертикального или горизонтального разбиения данных на разделы.
- При горизонтальном разбиении таблица, содержащая большое количество строк **делится** на несколько таблиц, содержащий одинаковые столбцы.
- При вертикальном разбиении таблица, содержащая большое число столбцов, делится на несколько таблиц, содержащих строки с эквивалентными уникальными идентификаторами.
- Нормализация — это процесс последовательного совершенствования логической модели для устранения дублирования данных и повышения эффективности использования дискового пространства.
- Нормализация позволяет снизить вероятность несогласованности **данных** и повышает быстродействие операций обновления данных.
- **Денормализация** — это обратный по отношению к нормализации процесс, в результате которого появляются таблицы с большим количеством полей. Для извлечения данных из них требуется меньше соединений (**join**), что положительно сказывается на скорости обработки запросов.
- Целостность данных означает их **непротиворечивость** и корректность.

- Предметная целостность определяет набор значений данных для столбца и допустимость хранения значения null.
- Целостность объекта-сущности требует, чтобы каждая строка в таблице имела уникальный идентификатор, или основной ключ.
- Ссылочная целостность обеспечивает сохранение гарантированной связи между основным (в таблице родительских объектов-сущностей) и внешним ключом (в таблице дочерних объектов-сущностей).
- Требования к целостности данных помогают обеспечить удовлетворение всех логических и физических требований к решению, а также соответствие физического дизайна спецификации.
- Определение бизнес-правил включает определение условий: которые необходимо удовлетворить, которых необходимо избежать, а также последовательность возникновения событий.
- Бизнес-правила реализуют в базе данных при помощи таких встроенных в ядро СУБД механизмов проверки, как типы данных, правила проверки данных, значения по умолчанию, ключи и триггеры.
- Программно бизнес-правила реализуют в хранимых процедурах или сценариях.
- Бизнес-правила также реализуют в виде компонентов, которые развертываются на серверах приложений.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Каким образом выполняется проектирование модели данных на этапе планирования?
2. Каково назначение схемы базы данных?
3. Каково назначение атрибутов?
4. Зачем определяют типы в базе данных?
5. Каким образом в большинстве СУБД поддерживаются связи вида «многие-ко-многим»?
6. Как оптимизировать транзакции, чтобы повысить производительность системы?
7. Какое влияние оказывает индексирование на доступ к данным?
8. Чем отличаются горизонтальное и вертикальное разбиение данных на разделы?
9. Каковы преимущества нормализации?
10. Что такое денормализация?
11. Какие три типа целостности данных обычно применяются в БД?
12. Как определяют требования к целостности данных?
13. Каковы критерии определения бизнес-правил?
14. Каким образом ключи применяются для реализации ссылочной целостности?
15. Каковы преимущества компонентов при реализации бизнес-правил?

ГЛАВА 9

Дизайн спецификаций безопасности

Занятие 1. Основные сведения о безопасности применительно к разработке приложений	270
Занятие 2. Планирование защиты приложения	273
Занятие 3. Использование функций безопасности .NET Framework	280
Занятие 4. Проектирование механизмов авторизации, аутентификации и аудита	289
Практикум. Моделирование опасностей и меры по их предотвращению	298
Резюме	298
Закрепление материала	300

В этой главе

В этой главе рассказывается о проектировании защиты приложения. Проектирование функций и политик безопасности — один из важнейших аспектов разработки приложений. Расходы на защиту корпоративных сетей постоянно растут, но компании все равно продолжают терпеть убытки из-за кражи интеллектуальной собственности, простоев системы, снижения производительности, потери репутации и доверия со стороны потребителей. Но даже в такой агрессивной среде можно обеспечить достаточный уровень защиты бизнес-приложений путем реализации механизмов аутентификации и авторизации, а также обеспечить целостность данных путем шифрования и проверки данных.

Прежде всего

Для изучения материалов этой главы необходимо:

- понимать технологии Microsoft;
- знать основные принципы обеспечения безопасности.

Занятие 1. Основные сведения о безопасности применительно к разработке приложений

Существует множество механизмов защиты приложений: брандмауэры, прокси-серверы, защищенные каналы связи и схемы аутентификации. Однако, чтобы обойти систему безопасности, злоумышленнику достаточно отыскать всего одну лазейку в системе. На этом занятии вы узнаете о наиболее часто встречающихся прорехах в защите приложений, о том, как ими пользуются злоумышленники, а также познакомитесь с важными принципами безопасного программирования.

Изучив материал этого занятия, вы сможете:

- ✓ обнаруживать типичные уязвимые места системы защиты приложений;
- ✓ определять недостатки традиционных моделей защиты;
- ✓ объяснять основные принципы проектирования защищенных приложений;
- ✓ давать определения наиболее важных терминов по безопасности,

Продолжительность занятия — около 15 минут.

Типичные бреши в системе безопасности

Для достижения своих неблагоприятных целей злоумышленники самым разнообразным образом «эксплуатируют» бреши в системе. Брешь — это слабое место в системе защиты, которую использует нападающий для получения доступа к корпоративной сети или отдельным ее ресурсам. Некоторые бреши (например, слабые пароли) не связаны с неудачным проектом или качеством разработки приложения. Однако их необходимо учитывать в процессе укрепления безопасности информационных систем предприятия. Вот наиболее типичные уязвимые места приложений.

- **Слабые пароли.** Они позволяют нападающему получить доступ не просто к отдельному компьютеру, а ко всей сети в целом.
- **Неправильная конфигурация ПО** — частая причина уязвимости системы. Если сервисы запускаются под учетной записью локальной системы (Local System) или другой учетной записью с высокими полномочиями, атакующий может использовать сервис для получения доступа к системе и осуществить злонамеренные действия.
- **Социальная инженерия** — метод получения у пользователей конфиденциальной информации посредством эксплуатации особенностей человеческой психологии или незнания пользователями принципов безопасности. Например, злоумышленник представляется администратором службы поддержки и под предлогом необходимости выполнения административной задачи узнает у пользователя его пароль.
- **Подключения к Интернету.** Параметры по умолчанию в Internet Information Services (IIS) 5.0 предусматривают активизацию ненужных сервисов и открытие неиспользуемых приложением портов. В результате злоумышлен-

ник получает дополнительные возможности для проведения успешной атаки. Например, модемные подключения выполняются мимо брандмауэров, защищающих сеть от непрошенных гостей. Узнав используемый модемом телефонный номер и пароль, хакер сможет подключиться к любому компьютеру сети.

- **Передача данных в незашифрованном виде.** Если обмен данными между сервером и пользователем осуществляется открытым текстом, велика опасность перехвата, прочтения или изменения сообщения хакером в процессе передачи.
- **Переполнение буфера.** Злоумышленники исследуют приложения на предмет возможности переполнения буфера, поскольку это позволяет вызвать нарушение в работе приложения или крах операционной системы. Кроме того, сообщения об ошибках облегчают хакеру обнаружение других брешей.
- **Внедрение SQL (SQL injection)** применяют, если SQL-выражение создается динамически, на основе данных, введенных пользователем. Атакующему иногда удается модифицировать SQL-выражение так, что оно выполняет не предусмотренные создателем программы действия.
- **Секретные данные, прописанные в коде.** Много проблем с безопасностью возникает из-за того, что секретные данные, например пароли или ключи шифрования, прописаны в коде приложения и злоумышленник в состоянии извлечь их.

Примечание Подробнее о проблемах с безопасностью рассказывается в Windows Security Resource Kit (Microsoft Press, 2003).

Недостатки традиционных моделей безопасности

Традиционные модели защиты не всегда защищают от опасностей, возникающих в сетевой среде. В большинстве таких моделей доступ к ресурсам предоставляется на основании идентификатора пользователя, от имени которого выполняется код. В этой модели отсутствует механизм ограничения доступа к ресурсам на основе идентификационных данных самого кода.

Эту модель защиты можно обойти, если обладающий достаточно широкими правами в системе пользователь по незнанию запустит на исполнение злонамеренный код:

- открыв вложенный в сообщение электронной почты файл;
- запустив внедренный в Web-страницу сценарий;
- открыв загруженный из Интернета файл.

Принципы создания стратегии защиты

Чтобы спроектировать защищенное приложение, вы должны знать следующие принципы безопасности и применять их при создании стратегии защиты.

- **Полагайтесь только на проверенные системы безопасности.** Необходимо применять проверенные промышленные системы безопасности всюду, где это возможно, а не создавать собственные «кустарные» решения. Используйте прошедшие проверку временем и протестированные специалистами алгоритмы и методы, предоставляемую платформой инфраструктуру и поддерживаемые поставщиками технологии. Если вы все-таки решили разработать собст-

венную инфраструктуру безопасности, то позаботьтесь, чтобы ваши алгоритмы и методы проверили эксперты компаний, занимающихся аудитом безопасности.

- **Никогда не доверяйте данным, полученным извне.** Необходимо в обязательном порядке проверять все данные, введенные пользователем или полученные из других сервисов.
- **Считайте все внешние системы незащищенными.** Если ваше приложение получает важные данные в незашифрованном виде, такая информация должна считаться скомпрометированной по умолчанию.
- **Придерживайтесь принципа наименьших привилегий.** В любых обстоятельствах предоставляйте учетной записи, от имени которой будет выполняться сервис, только минимально необходимые для выполнения задачи права. Осуществляйте доступ к ресурсам от имени учетных записей, обладающих минимально необходимыми разрешениями.
- **Минимизируйте количество доступных компонентов и данных.** Риск тем выше, чем больше число компонентов и объем данных, доступных из приложения, поэтому предоставляйте лишь те функции, которые действительно требуются конечным пользователям.
- **Используйте безопасные параметры по умолчанию.** Блокируйте сервисы, права учетных записей и технологии, в которых нет явной необходимости. Параметры конфигурации по умолчанию приложения, устанавливаемого на клиентском или серверном компьютере, должны быть безопасными.
- **Не полагайтесь на защиту, основанную на незнании или сокрытии определенной информации.** Шифрование данных следует строить на основе надежных ключей и проверенных алгоритмов. Безопасное хранение данных предотвратит доступ посторонних при любых обстоятельствах. Перестановка строк, хранение информации в неожиданных местах и тому подобные «фокусы» не обеспечивают должной защиты.
- **Устраняйте дыры в безопасности системы, классифицируя уязвимые места системы по схеме STRIDE.** Каждая буква в сокращении STRIDE обозначает определенную категорию опасности: *подмена сетевых объектов* (spoofing identity), *модификация данных* (tampering), *отказ от причастности* (repudiation), *раскрытие информации* (disclosure), *отказ в обслуживании* (denial of service) и *повышение привилегии* (elevation of privilege). Система должна противостоять всем подобным угрозам.

Примечание Подробнее об обеспечении безопасности рассказано в книге Майкла Ховарда и Дэвида Леблана «Защищенный код» («Русская редакция», М., 2001).

Занятие 2. Планирование защиты приложения

Приступая к планированию особенностей защиты приложения, необходимо четко представлять, что именно угрожает его безопасности. Лишь после этого определяют и разрабатывают адекватные меры защиты. Планирование безопасности приложения предусматривает ряд мероприятий.

- **Определение опасностей (моделирование угроз).** Это самый важный момент планирования безопасности, так как, не зная опасностей, невозможно определить политики безопасности. Оценивая угрозу приложению, собирают следующую информацию:
 - какие активы организации следует защищать;
 - что именно угрожает каждому из активов;
- **Создание политики безопасности,** позволяющей устранить или минимизировать угрозы. После определения большинства опасностей их следует разбить по категориям и определить стратегию защиты для каждой категории. О методах предотвращения угроз — далее на этом занятии.

Разработано много методов определения и систем классификации опасностей. Одна из наиболее популярных — STRIDE. На этом занятии рассказывается о моделировании угроз с помощью именно этой системы.

Изучив материал этого занятия, вы сможете:

- ✓ определить связанные с защитой операции, которые выполняются на различных стадиях процесса дизайна MSF;
- ✓ рассказать о моделировании опасностей;
- ✓ создать модель опасностей;
- ✓ использовать модели опасностей для принятия мер по предотвращению опасностей.

Продолжительность занятия — около 15 минут.

Роль безопасности в процессе разработки приложений

Основная причина слабой защиты приложения — отсутствие или недостаточное внимание к этой стороне проекта на ранних стадиях дизайна. Если решение вопросов безопасности откладывать «на потом», приложение окажется сильно уязвимым, а стоимость разработки и поддержки значительно возрастет.

Планирование и реализацию функций безопасности следует проводить на всех этапах разработки приложения. В таблице 9-1 перечислены этапы модели процессов MSF и выполняемые на каждом этапе мероприятия по обеспечению безопасности.

Таблица 9-1. Мероприятия по обеспечению безопасности на различных этапах модели процессов MSF

Этап	Действия
Создание общей картины	Сбор требований по безопасности. Команда общается с заказчиками и всеми заинтересованными в проекте лицами, чтобы выяснить, какие данные и операции являются конфиденциальными, какие привилегии необходимы пользователям и как существующие приложения удовлетворяют этим требованиям. На основании этой информации создается документ, в котором перечисляются все требования по безопасности
Планирование	Создание моделей опасностей, которые позволят предвидеть возможные угрозы (см. раздел «Модель опасностей STRIDE»). В функциональной спецификации предусматривают функции безопасности, позволяющие устранить или минимизировать все обнаруженные риски
Разработка	Реализация функций безопасности в соответствии с функциональной спецификацией
Стабилизация	Тестирование защищенности приложения. Модель опасностей дополняется новой информацией, выявленной в процессе исследования, тестирования и взаимодействия с пользователями
Развертывание	Мониторинг опасностей, грозящих защите приложения

После поставки готового продукта заказчику следует постоянно следить за безопасностью приложения и атакам на него, проверяя, достаточно ли защищено решение. При обнаружении брешей в защите их необходимо «латать» и документировать для устранения в следующих версиях.

Модель опасностей STRIDE

STRIDE — это метод, применяемый для определения и классификации опасностей, грозящих приложению. Каждая буква в сокращении STRIDE обозначает определенную категорию опасности: *подмена сетевых объектов* (Spoofing identity), *модификация данных* (Tampering), *отказ от причастности* (Repudiation), *раскрытие информации* (disclosure), *отказ в обслуживании* (Denial of service) и *повышение привилегий* (Elevation of privilege). Система должна противостоять угрозам всех подобных видов. Большинство опасностей попадают сразу в несколько категорий STRIDE.

- Подмена сетевых объектов предусматривает ситуацию, в которой злоумышленник прикидывается доверенным объектом. Например, узнает пароль доверенного пользователя, а затем получает доступ к конфиденциальной информации или отправляет по электронной почте письмо от имени доверенного источника.

Примечание Особым случаем подмены сетевых объектов является подмена IP-адреса. Это происходит, когда злоумышленник подставляет ложный IP-адрес при пересылке через Интернет, при этом у адресата создается впечатление, что он взаимодействует с доверенным источником. В такой ситуации злоумышленник может получить незаконный доступ к компьютерной системе.

- **Модификация данных.** Пользователь получает неадекватный доступ к компьютеру и оказывает влияние на его работу, изменяя конфигурацию или данные. Модификация может быть как злонамеренной, так и случайной. Например, случайная модификация происходит, когда пользователь удаляет сетевые файлы или изменяет данные в базе данных, не должен иметь доступ.
- **Отказ от причастности** возможен, когда системный администратор или агент безопасности не в состоянии доказать, что пользователь выполнил (со злым умыслом или без него) определенную операцию. Например, как злоумышленник может безнаказанно утверждать, что не атаковал систему, так администратор не может подтвердить наличие записей.
- **Раскрытие информации** — получение пользователем доступа к информации, которую ему просматривать не разрешено, например файл с номерами кредитных карт и сроками их действия.
- **Отказ в обслуживании** — любая атака, целью которой заключается в отключении или предотвращении доступа к вычислительному ресурсу. Атаки вида «отказ в обслуживании» (Denial-of-Service, DoS) обычно вызывают:
 - нарушение работы приложения или операционной системы;
 - выполнение процессором бессмысленных длительных операций;
 - перегрузку памяти, замедляющую работу приложений и ОС;
 - сужение полосы пропускания сети.
- **Повышение привилегий.** Пользователь получает более высокие, чем изначально назначенные администратором привилегии, что дает злоумышленникам возможность выполнять атаки, относящиеся к любой из перечисленных категорий.

Создание модели опасностей

Создание модели опасностей — первый шаг в построении защищенного приложения. Этот процесс состоит из нескольких задач.

1. **Проведите мозговые штурмы.** Пригласите опытных разработчиков и членов каждой из ролей MSF на сеанс мозгового штурма, на котором попытайтесь выявить всевозможные угрозы.
2. **Перечислите все возможные опасности.** В процессе мозгового штурма снабдите всех участников списком предполагаемых функций и опишите архитектуру продукта. Попросите участников представить опасности, которые могут возникнуть при взаимодействии компонентов приложения и при связи с другими системами.
3. **Определите категории по STRIDE.** После создания начального списка угроз безопасности разнесите их по категориям в соответствии с моделью STRIDE.
4. **Создайте описания опасностей.** Для каждой угрозы безопасности следует задокументировать следующую информацию:
 - тип опасности;
 - влияние атаки на организацию в плане возможного материального урона и усилий, необходимых на восстановление;
 - конкретный метод, позволяющий хакеру воспользоваться уязвимостью системы;

- вероятность атаки
 - возможные спосо
5. Проведите исследование, чтобы противодействовать атаке. Другие и технические вопросы. В процессе мозгового штурма неизбежно возникнут вопросы. Если во время встречи доступен Интернет, вы можете найти ответы немедленно, не прерывая встречи.
6. Распределите риски, связанные с каждой угрозой, по ранжиру, то есть назначьте каждой категории уровень риска, который измеряется как частное от деления критичности опасности на вероятность ее возникновения.

рис. 9-1 показано Web-приложение учета расходов. Необходимо создать модель опасностей этого приложения.

Web-приложение учета расходов сотрудника

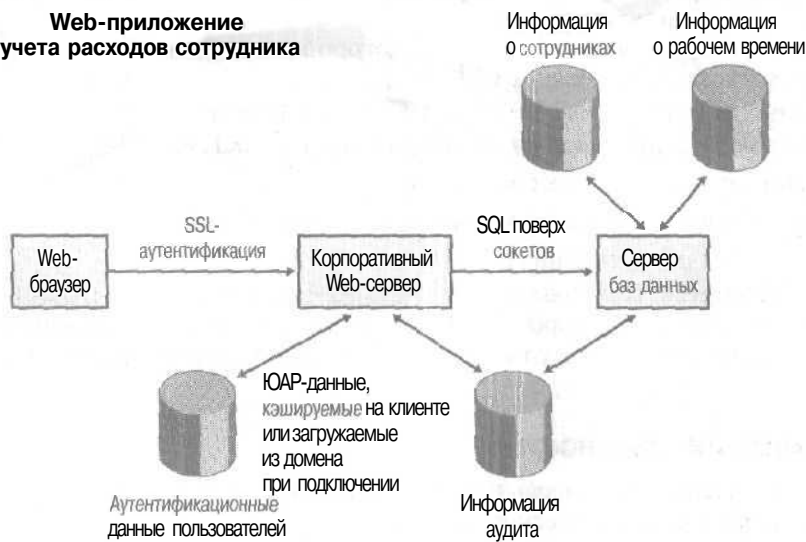


Рис. 9-1. Приложение для учета расходов сотрудника

Опасности и их категории

В таблице 9-2 перечислены возможные опасности, грозящие приложению для учета расходов и определенные в процессе мозгового штурма. Они классифицированы в соответствии с моделью STRIDE.

Таблица 9-2. Опасности, грозящие приложению учета расходов

Угроза	Категория*
В базах данных рабочего времени и информации о сотрудниках существует опасность прямого доступа, при наличии которого не уполномоченные для этого пользователи получают возможность просматривать и изменять конфиденциальные данные (например, статистику)	T, I
Сервер баз данных может стать недоступным, если на него обрушится лавина пакетов TCP/IP	D

* Расшифровку обозначений см. в тексте на стр. 274. — Прим. ред.

Таблица 9-2. (окончание)

Угроза	Категория*
Посторонние пользователи или приложения получают возможность просматривать журналы аудита, а также изменять или удалять содержащиеся в них данные	T, R
Посторонние лица перехватывают и перлюстрируют пакеты аутентификационного протокола LDAP (Lightweight Directory Access Protocol). На основании информации пакетов злоумышленник может понять, как реализовать олицетворение (impersonate) другого пользователя. В частности, простой оператор сможет действовать от имени финансового директора	S, I, E
Данные, которыми обмениваются Web-браузер и Web-сервер, уязвимы — они доступны для просмотра посторонним пользователям. Это чревато компрометацией важной информации, в том числе паролей, личной информации сотрудников и сведений о зарплате и рабочем времени	T, I
Web-сервер может стать недоступным. Более того, злоумышленник может развернуть вместо него подставной Web-сервер. Ясно, что это вызовет компрометацию всей информации, поступающей от браузера	S, T, I, D, E
В обрабатываемые браузером Web-страницы можно внедрять вирусы и черви. При отображении Web-страницы этот зловредный код исполняется и способен нанести значительный урон компьютеру пользователя	T, D

* Расшифровку обозначений см. в тексте на стр. 274. — Прим. ред.

Использование модели опасностей

После определения угроз следует решить, как противостоять каждой из них. Существует несколько способов решения этой задачи.

- **Проинформировать пользователей об опасности.** Если брешь возникает при выполнении пользователем определенных действий, чаще всего достаточно просто уведомить его. Например, пользователям Microsoft Windows XP, устанавливающим флажок, разрешающий отображение защищенных файлов ОС, выдается предупреждение о значительной опасности, которой подвергаются файлы. В этом случае угроза порчи данных в основном заключается в возможности доступа и повреждения пользователем файлов операционной системы.
- **Удалить функции.** Если вам известна функция, представляющая серьезную угрозу безопасности, и нет способа эффективно защитить продукт от опасности, хорошенько подумайте, стоит ли включать эту функцию в финальную версию продукта.
- **Определить способ противодействия.** Если без опасной функции не обойтись, необходимо выбрать «противоядие» — метод защиты. Существует много таких методов.

Методы противодействия опасностям

Они делятся на две категории: общие методы и методы модели STRIDE. В таблице 9-3 описаны некоторые из общих методов.

Таблица 9-3. Методы противодействия опасностям

Метод	Описание
Аутентификация и авторизация	Аутентификация позволяет выяснить, действительно ли объект является тем, за кого себя выдает. Авторизация — это процесс определения прав доступа к ресурсам
Защищенная связь	Необходимо обеспечить безопасность связи между уровнями приложения, чтобы предотвратить изменение данных в процессе передачи или при их нахождении в очереди. Для безопасной связи применяются: <ul style="list-style-type: none"> • <i>SSL (Secure Sockets Layer)</i>. Этот протокол позволяет построить зашифрованный канал связи между клиентом и сервером; • <i>IPSec</i> применяется для защиты данных, которыми обмениваются пары компьютеров, например серверы приложений и базы данных; • <i>Виртуальные частные сети (Virtual Private Network, VPN)</i> позволяют организовать связь типа «точка — точка» (поверх IP) через Интернет (или другие сети)
Качество сервиса (Quality of Service, QoS)	Обеспечение профилирования сообщений, проходящих в систему
Ограничение числа входящих запросов (throttling)	Ограничение количества сообщений, проходящих в систему. Если не контролировать число поступающих в систему сообщений, она может «потонуть» в потоке сообщений
Аудит	Процесс сбора и сохранения информации о действиях пользователей и важных событиях с целью последующего анализа. Его также называют журналированием (logging). Приложения часто регистрируют информацию о важных событиях в журналах событий Windows. Вы можете использовать эти записи для аудита доступа к системе и устранения неполадок
Фильтрация	Перехват, проверка и контроль поступающих в систему сообщений
Наименьшие привилегии	Пользователям предоставляется минимальный уровень привилегий, достаточный лишь для выполнения задач, и не более того

В таблице 9-4 приводится сокращенный список методов предотвращения опасностей, применяемых к различным категориям STRIDE.

Таблица 9-4. Методы предотвращения опасностей, относящихся к различным категориям STRIDE

Метод	Тип угрозы*
Аутентификация	S, D
Защита секретной информации	S, I
Записи аудита	R
Отказ от хранения секретной информации	S, I
Конфиденциальные протоколы	I
Авторизация	T, I, D

* Расшифровку обозначений см. в тексте на стр. 274. — Прим. ред.

Таблица 9-4. (окончание)

Метод	Тип угрозы*
Хеши	T
Коды аутентификации сообщений	T
Цифровые подписи	T, R
Безопасные протоколы	T, R
Временные метки	R
Фильтрация	D
Ограничение числа входящих запросов	D
Качество сервиса	D
Минимизация привилегий	E

* Расшифровку обозначений см. в тексте на стр. 274. — *Прим. ред.*

Примечание Подробнее о моделировании опасностей и способах противодействия вы можете получить в книге Майкла Ховарда и Дэвида Леблана «Защищенный код» («Русская Редакция», М., 2001).

Занятие 3. Использование функций безопасности .NET Framework

На этом занятии описываются важные функции безопасности, предусмотренные в каркасе .NET Framework.

Примечание Подробнее о создании защищенных приложений Microsoft ASP.NET рассказано в статье «Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication» в разделе .NET Security сайта MSDN (<http://msdn.microsoft.com>).

Изучив материал этого занятия, вы сможете:

- ✓ дать определение:
 - проверки безопасности типа;
 - подписания кода;
 - шифрования и подписания данных;
 - безопасности доступа из кода;
 - безопасности, основанной на ролях;
 - изолированного хранилища;
- ✓ объяснить функции защиты в Web-технологиях каркаса .NET.

Продолжительность занятия — около 20 минут.

Контроль типов

Контроль безопасности типов играет ключевую роль в изоляции сборок и обеспечении безопасности среды исполнения.

Определение кода с контролем типов

Любой код, имеющий доступ только к той памяти, на которую у него есть права, называется *кодом с контролем типов* или *безопасным с точки зрения типов* (type-safe). Например, такой код не в состоянии получить доступ к закрытым (private) полям другого объекта.

Код без контроля типов может представлять угрозу для приложения, например получить доступ к неуправляемому (unmanaged) коду и изменить его в соответствии со злым умыслом хакера. Поскольку такой код не является защищенным, исполняющая среда не в состоянии запретить ему обращаться к машинному (native) коду напрямую. В коде с контролем типов механизм безопасности исполняющей среды надежно заботится о защите и предоставляет коду доступ строго в отведенных для него рамках. Для исполнения небезопасного кода необходимо разрешение *SkipVerification*, которое предоставляется только пользующемуся высоким доверием коду.

Определение проверки контроля типов

Во время JIT-компиляции (Just-in-time compilation) MSIL-код (Microsoft Intermediate Language) компилируется в машинный код. На этом этапе стандарт-

ный процесс проверяет метаданные и MSIL-код метода на предмет корректности (well-formed) и контроля типов.

Компоненты с контролем типов могут работать в одном процессе, даже если их уровни безопасности различаются. Следующее справедливо для кода с подтвержденной поддержкой контроля типов:

- ссылка на тип соответствует типу, на который ссылается;
- у объекта вызываются только должным образом определенные операции;
- методы вызываются только через определенные интерфейсы, гарантируя выполнение всех проверок безопасности.

Такие ограничения гарантируют проверку ограничений безопасности. Кроме того, экземпляры различных сборок с контролем типов могут безопасно выполняться в одном процессе, поскольку исключается возможность пересечения занятых ими участков памяти.

Подписание кода

Чтобы гарантировать надежность ресурса, загружаемого по сети, необходимо обеспечить его:

- **аутентичность**, которая позволяет пользователям однозначно определять происхождение кода и не дает злоумышленникам возможности подменить идентификационные данные компании, производителя кода;
- **целостность**, которая гарантирует неизменность кода посторонними после его публикации.

Определение подписания кода

Подписание кода путем присвоения ему *строгого имени* (strong name) позволяет подтвердить происхождение кода и защищает от компрометации. Подписание кода — это процесс присвоения фрагменту кода реквизитов, которые удостоверяют его создателя. Реквизиты кода обычно проверяются перед его установкой и запуском на исполнение.

Подписание кода в .NET Framework

Подписание кода в .NET Framework основывается на строгих именах (strong name), кроме того, каркас .NET Framework поддерживает цифровые сертификаты и подпись в стандарте Authenticode.

Шифрование и подписание данных

Подписание данных и шифрование — это модификация данных, цель которой заключается в защите информации от посторонних и возможности проверить, не скомпрометированы ли данные, то есть не подверглись ли они изменению.

Определение шифрования и расшифровывания

Шифрование — это процесс сокрытия данных перед отсылкой или сохранением. Незашифрованные данные называют *открытым текстом* (plaintext), а зашифрованные — *зашифрованным текстом* (ciphertext), или шифротекстом. *Расшифровка* — процесс восстановления зашифрованного текста в читабельный, открытый текст. Процессы шифрования и расшифровки лежат в основе методов хэширования и подписания данных.

Определение криптографического хеширования

В процессе *хеширования* на основании исходного текста создается последовательности байт фиксированной длины, которая называется *хешем* и практически однозначно идентифицирует исходный текст. Хеш вычисляется путем применения математической функции хеширования к произвольному объему данных. Криптографические хеши, созданные с применением криптографических функций .NET Framework, статистически уникальны, то есть хеши двух различных двухбайтовых последовательностей отличаются.

Определение подписанных данных

Подписанные данные — это данные, оформленные в соответствии с определенными стандартами. Кроме самого информационного наполнения они содержат зашифрованные хеши, выполняющие роль подписи данных. Хеши используются для подтверждения личности подписавшего данные и гарантии неизменности сообщения с момента подписания.

Безопасность доступа из кода

Система безопасности Windows предотвращает доступ к компьютерной системе не уполномоченных для этого пользователей. Однако сама по себе она не в состоянии предотвратить загрузку и запуск вредоносного кода полномочными пользователями.

Определение безопасности доступа из кода

Безопасность доступа из кода, обеспечиваемая .NET Framework, защищает компьютерные системы от вредоносного или просто ошибочного кода и позволяет безопасно запускать мобильный код.

Безопасность доступа из кода позволяет регулировать уровень доверия коду в зависимости от его происхождения (*origin*) и признаков (*evidence*), например подписи строгим именем и прочих атрибутов идентификационных данных кода. В частности, загруженному из корпоративной сети коду, опубликованному вашей компанией, можно доверять значительно больше, чем коду из Интернета, тем более если автор программы не подписал его.

Запрос требуемого уровня привилегий

.NET Framework позволяет включать в приложение функции, которые требуют от операционной системы определенного уровня привилегий. Этот запрос определяет уровень привилегий приложения, который:

- требуется для запуска;
- может использоваться, но не требуется для запуска;
- не нужен и должен явно исключаться.

Безопасность, основанная на ролях

Этот механизм защиты в первую очередь ориентирован на устранение угрозы подмены сетевых объектов путем запрещения не прошедшим авторизацию пользователям выполнять операции, для которых такая процедура обязательна. Она предусматривает проверку идентификационных данных и принадлежность пользователя к определенной роли.

Совет Помимо классов, предназначенных для реализации основанной на ролях защиты в других механизмах аутентификации, в .NET Framework есть классы для определения пользователей и групп Windows.

Рис. 9-2 иллюстрирует основанную на ролях модель безопасности.

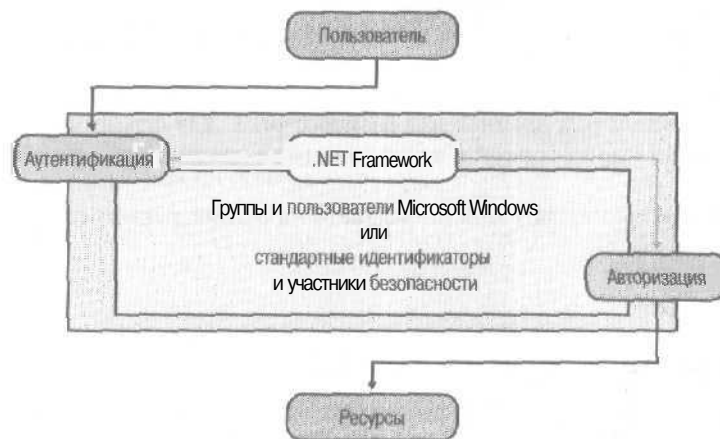


Рис. 9-2. Основанная на ролях безопасность

Аутентификация

Аутентификация — это процесс выяснения и подтверждения личности пользователя путем *анализа реквизитов (credentials)* и сверки с данными из полномочного источника. Механизмов аутентификации множество, причем некоторые из них применяются вместе с основанной на ролях безопасностью .NET Framework.

В качестве механизмов аутентификации часто используются стандартные функции ОС, Passport, а также реализованные в приложении механизмы, такие, как NTLM и Kerberos версии 5.

Авторизация

Авторизация — процесс проверки прав и разрешений пользователей на доступ к ресурсам. Она выполняется после аутентификации, и для нее требуется информация об идентификационных данных пользователя и ролях для определения ресурсов, к которым пользователю разрешается доступ. Для реализации авторизации вы можете применить основанную на ролях систему безопасности, встроенную в .NET Framework.

Изолированное хранилище

Очень часто от приложения требуется хранить на клиенте такие персональные данные, как предпочтения пользователя или состояние приложения.

Это опасно, так как злоумышленник может получить доступ к хранилищу и повредить или изменить информацию, поэтому требуется стандартная система хранения. Разработка «доморощенной» системы хранения данных оказывается сложной, а результат чаще всего оказывается неудовлетворительным. Вот поче-

му так важно защитить данные, организовав *изолированное хранилище* (isolated storage).

В этом случае для хранения данных используется изолированная виртуальная файловая система на клиенте. Приложение сохраняет данные в уникальном отделении (compartment), ассоциированном с некоторыми реквизитами кода, например Web-сайтом, издателем или подписью.

Отделение данных — это абстракция, а не какое-то физическое место хранения. Она состоит из одного или более файлов изолированного хранилища, называемых *ячейками* (store). Физическое местоположение данных указано в ячейке, а доступ к ней выполняется «прозрачно» для разработчика. Для ограничения места, занимаемого той или иной сборкой в изолированном хранилище, применяются квоты.

Разрешения на доступ к хранилищу можно задавать на основе реквизитов:

- пользователя;
- сборки;
- приложения.

Функции безопасности в технологиях .NET

Web-приложения, созданные на базе .NET, реализуют один или более логических сервисов, которые могут базироваться на разных технологиях, например: Microsoft ASP.NET, Enterprise Services, XML Web Services, удаленном взаимодействии (remoting), Microsoft ADO.NET и Microsoft SQL Server. Для создания эффективных стратегий защиты необходимо понимать тонкости различных функций безопасности отдельных продуктов и конкретной технологии и уметь организовать их взаимодействие.

Безопасность в ASP.NET

ASP.NET — это эффективный инструмент для создания Web-страниц. Когда информация о кредитной карте пользователя сохраняется на Web-сайте, файл или базу данных, в которых хранится эта информация, следует защитить от доступа посторонних. ASP.NET в совокупности с IIS позволяют проверять реквизиты пользователя (такие, как имя и пароль) посредством любой из следующих технологий:

- стандартных механизмов аутентификации в Windows: базовой (basic), на основе хеша (digest) или интегрированной (NTLM или Kerberos);
- аутентификации с использованием паспорта (Passport);
- форм;
- клиентских сертификатов.

В ASP.NET аутентификация реализована посредством провайдеров аутентификации.

- **Аутентификация на основе форм.** Не прошедшие аутентификацию запросы перенаправляются в HTML-форму на стороне клиента. Пользователь вводит свои реквизиты и отправляет данные формы на сервер. В случае успешной аутентификации система создает cookie-файл, содержащий учетные данные пользователя или ключ для дальнейшей работы пользователя.
- **Аутентификация с использованием Passport** — это предоставляемый Microsoft централизованный сервис аутентификации, который позволяет сайтам-участникам поддерживать единый вход в систему и ключевые сервисы по работе с профилями.

- **Windows-аутентификация** поддерживается в ASP.NET при совместной работе с IIS. Аутентификацию выполняет IIS по одному из трех механизмов: базовая аутентификация, хеш-аутентификация и интегрированная Windows-аутентификация. В случае успешной аутентификации на IIS среда ASP.NET использует полученные идентификационные данные для авторизации доступа.

Примечание Подробнее о системе безопасности в ASP.NET рассказывается в статье «ASP.NET Security» из раздела «Building Secure ASP.NET Applications» на Web-сайте MSDN (<http://msdn.microsoft.com>).

Защита в Enterprise Services

Традиционные сервисы COM+ (распределенные транзакции, JIT-активизация, создание пулов объектов и управление параллельными операциями) доступны и для компонентов .NET. В среде .NET эти сервисы называются Enterprise Services.

На рис. 9-3 на примере клиентского Web-приложения ASP.NET показаны поддерживаемые Enterprise Services функции аутентификации, авторизации и защищенного обмена данными.

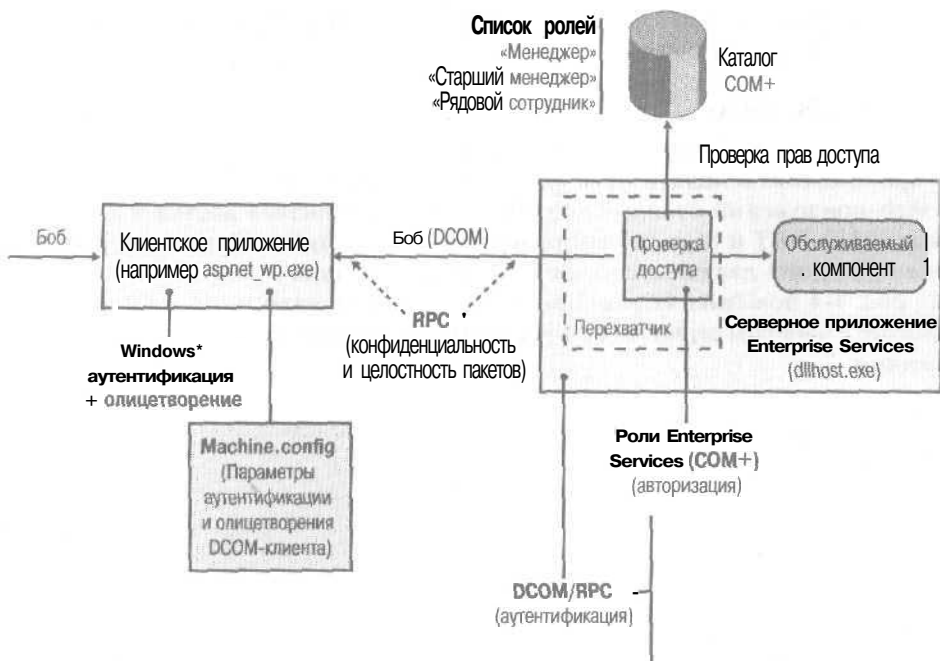


Рис. 9-3. Архитектура основанной на ролях защиты, интегрированной в Enterprise Services

Примечание Дополнительную информацию о защите в Enterprise Services вы найдете в статье «Enterprise Services Security» в разделе «Building Secure ASP.NET Applications» на Web-сайте MSDN (<http://msdn.microsoft.com>).

Безопасность Web-сервисов

Web-сервисы обеспечивают обмен данными и удаленный вызов логики приложения с использованием обмена сообщениями в формате SOAP и позволяют пересылать данные через брандмауэры и между гетерогенными системами. Защита Web-сервисов обеспечивается на трех уровнях:

- на уровне платформы или на транспортном уровне («точка-точка»);
- на уровне приложения (пользовательский механизм аутентификации);
- на уровне сообщения (сквозная).

Примечание Дополнительную информацию о защите Web-сервисов вы найдете в статье «Web Services Security» в разделе «Building Secure ASP.NET Applications» на Web-сайте MSDN (<http://msdn.microsoft.com>).

Защита в .NET при удаленном взаимодействии

Механизм удаленного взаимодействия .NET позволяет получить доступ к удаленным и распределенным объектам, находящимся в других процессах и даже на других машинах. У этого механизма нет собственной модели безопасности. Для аутентификации и авторизации между клиентом и сервером используются каналы и процессы на узлах.

ADO.NET и SQL Server

Библиотека ADO.NET предоставляет сервисы доступа к данным, используется в Web-приложениях и поддерживает отсоединенные наборы записей. При создании Web-приложений очень важно обеспечить безопасный доступ и хранение данных. ADO.NET и SQL Server поддерживают ряд функций защиты, которые можно применять для обеспечения безопасного доступа к данным.

На рис. 9-4 показана модель прикладного уровня удаленного взаимодействия вместе с набором сервисов безопасности, предоставляемых различными технологиями.

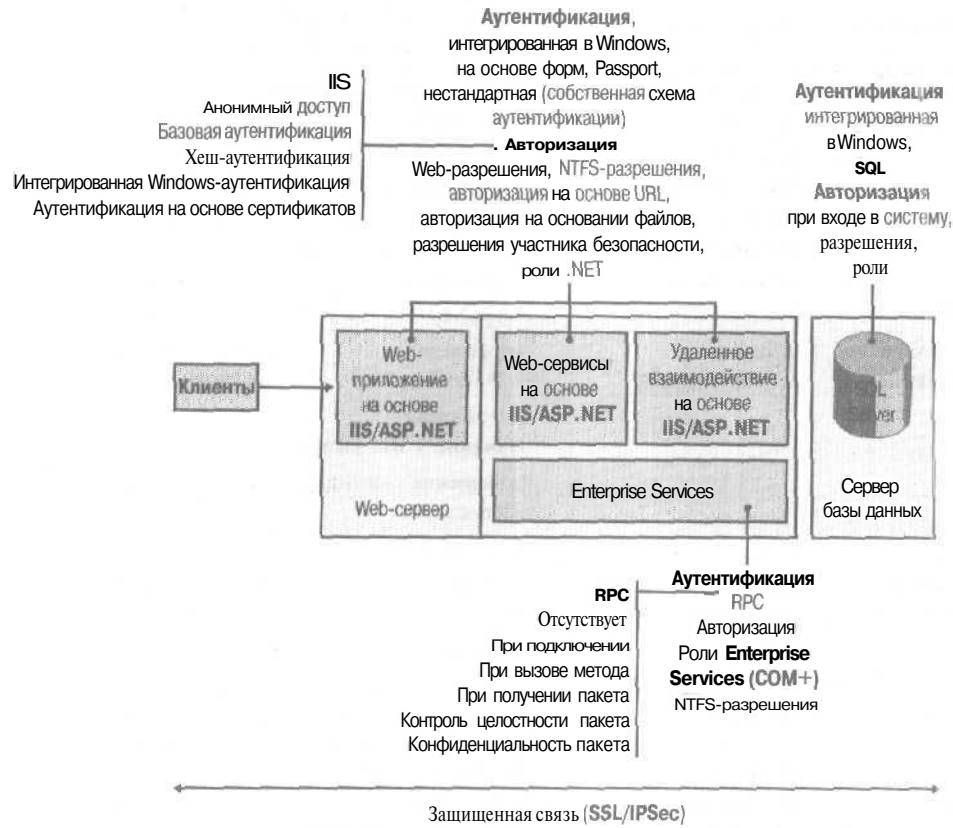


Рис. 9-4, Архитектура системы безопасности

В таблице 9-5 собраны воедино и систематизированы функции аутентификации, авторизации и защищенной связи, поддерживаемые технологиями .NET.

Таблица 9-5. Функции защиты и технологии .NET

Технология	Аутентификация	Авторизация	Защищенная связь
ASP.NET	Нет (собственная схема аутентификации)	Авторизация на основе файлов	
	Windows-аутентификация	Авторизация на основе URL-адреса	
	Аутентификация на основе форм	Разрешения участника безопасности	
	Аутентификация на основе Passport	Роли .NET	

(см. следующую страницу)

Таблица 9-5. Функции защиты и технологии .NET (окончание)

Технология	Аутентификация	Авторизация	Защищенная связь
Web-сервисы	Windows-аутентификация	Авторизация на основе файлов	SSL и шифрование на уровне сообщений
	Нет (собственная схема)	Авторизация на основе URL-адреса	
	Аутентификация на уровне сообщения	Разрешения участника безопасности Роли .NET	
Удаленное взаимодействие	Windows-аутентификация	Авторизация на основе файлов	SSL и шифрование на уровне сообщений
		Авторизация на основе URL-адреса	
		Разрешения участника безопасности Роли .NET	
Enterprise Services	Windows-аутентификация	Роли Enterprise Services (COM+) Разрешения NTFS	Шифрование в RPC (Remote Procedure Call)
SQL Server	Windows-аутентификация (Kerberos/NTLM) Встроенная аутентификация SQL Server	Вход в систему сервера Вход в базу данных Встроенные роли базы данных Пользовательские роли Роли приложений Разрешения на объекты	SSL

Занятие 4. Проектирование механизмов авторизации, аутентификации и аудита

Создание проекта систем авторизации и аутентификации в приложении — задача не из легких. Причем чем раньше вы займетесь этой стороной дизайна продукта, тем успешнее удастся защитить систему. В большинстве случаев для повышения безопасности решения дополнительно реализуют систему аудита действий пользователей и различных бизнес-операций.

На этом занятии рассказывается об основных особенностях процесса проектирования стратегий авторизации и аутентификации, а также о том, как проектируются авторизация, аутентификация и аудит для различных уровней приложения,

Изучив материал этого занятия, вы сможете:

- ✓ определить процесс проектирования стратегий авторизации и аутентификации;
- ✓ разработать стратегии авторизации для компонентов пользовательского интерфейса и компонентов пользовательских процессов;
- ✓ разработать стратегии авторизации для бизнес-компонентов и компонентов доступа к данным;
- ✓ разработать стратегии аутентификации для компонентов пользовательского интерфейса и компонентов пользовательских процессов;
- ✓ разработать стратегии аутентификации для компонентов доступа к данным;
- ✓ разработать стратегии аудита.

Продолжительность занятия — около 25 минут.

Разработка стратегий авторизации и аутентификации

Разработка общей стратегии аутентификации и авторизации выполняется в несколько этапов.

1. Определение ресурсов.
2. Выбор стратегии авторизации.
3. Выбор объектов безопасности, используемых для контроля доступа к ресурсам.
4. Определение типа объектов безопасности — перемещаемые или нет.
5. Выбор метода аутентификации.
6. Определение механизмов перемещения объектов безопасности в системе,

Определение ресурсов

Определите ресурсы, которые приложение предоставляет клиентам, на пример:

- ресурсы Web-сервера: Web-страницы, Web-сервисы и статические ресурсы (HTML-страницы и изображения);
- * ресурсы базы данных, например специфические данные отдельных пользователей или информация, общая для всего приложения;

- сетевые ресурсы, **такие**, как ресурсы удаленной файловой системы и данные каталогов, например Active Directory.

Выбор стратегии авторизации

Существует две ключевые стратегии авторизации:

- **на основе ролей.** Система различает пользователей по их ролям;
- **на основе ресурсов.** Каждый ресурс **защищается** списком управления доступом (ACL) Windows. При доступе к ресурсам приложение олицетворяет клиента, что позволяет операционной системе выполнить стандартную проверку прав на доступ.

Выбор объекта безопасности для доступа к ресурсам

Необходимо выбрать объекты безопасности и их реквизиты, которые будут применяться для доступа к ресурсам на всех уровнях приложения.

- **Инициатор цепочки вызовов.** Модель олицетворения или делегирования, в которой реквизиты инициатора цепочки вызовов получаются и передаются на каждый из уровней системы. Делегирование — ключевой критерий для определения механизма аутентификации.
- **Процесс.** Этот вариант используется по умолчанию. Доступ к локальным ресурсам и вызовы вниз по иерархии выполняются от имени текущего процесса. Возможность выбора данного варианта определяется тем, какие границы разделяют **вызывающую** и вызываемую систему, поскольку принимающая система должна получить и распознать идентификационные данные процесса.
- **Учетная запись сервиса.** В этом механизме используется специально созданная учетная запись сервиса, например для доступа к базе данных компонент может использовать фиксированные имя пользователя и **пароль**. Когда требуются **идентификационные** данные учетной записи, Windows следует использовать серверное приложение Enterprise Services.
- **Пользовательские реквизиты.** Если нет возможности обратиться к учетным записям Windows, вы вправе создать собственные реквизиты, содержащие информацию конкретного контекста безопасности. Это могут быть списки ролей, уникальные идентификаторы или любая другая информация.

Потоки идентификационных данных

Чтобы обеспечить для каждого отдельно взятого пользователя авторизацию, аудит и выборку данных, часто приходится передавать реквизиты инициатора вызова через разные уровни приложения и даже между разными компьютерами. Например, если серверный диспетчер ресурсов должен авторизовать каждого **обращающегося** к нему клиента, ему придется в обязательном порядке получать реквизиты клиента.

На основании требований менеджера ресурсов к авторизации и требований по аудиту определите, какие реквизиты должны пересылаться через приложение.

Выбор способа аутентификации

Выбор способа аутентификации определяется двумя ключевыми факторами: пользователями приложения (браузеры и наличие учетных записей Windows) и требованиями приложения по олицетворению, делегированию и аудиту.

При разработке Web-приложений в среде .NET вам доступны несколько вариантов аутентификации. Вы вправе применить один из нескольких механизмов аутентификации, поддерживаемых IIS, или реализовать в приложении собственную схему. При выборе способа аутентификации принимайте во внимание следующие обстоятельства:

- операционные системы клиента и сервера;
- тип клиентского браузера;
- количество пользователей, а также местоположение и тип базы данных, в которой хранятся имена пользователей и пароли;
- требования по развертыванию, например, должно ли приложение поддерживать работу в Интернете, интрасети или в сети, защищенной брандмауэром;
- тип приложения, например интерактивный Web-сайт или неинтерактивный Web-сервис;
- уровень конфиденциальности защищаемых данных;
- производительность и масштабируемость;
- требования приложения по авторизации: должно ли приложение быть доступным всем пользователям или лишь определенная его часть, причем только зарегистрированным пользователям, а остальные области — только администраторам.

Вот некоторые из поддерживаемых в ASP.NET и IIS типов аутентификации:

- аутентификация на основе форм;
- аутентификация на основе Passport;
- интегрированная Windows-аутентификация;
- базовая аутентификация;
- аутентификация на основе хешей, или дайджестов.

Организация обмена реквизитами

Для поддержки контекста безопасности вы вправе передавать реквизиты как на уровне приложения, так и на уровне ОС. На уровне приложения реквизиты передаются в параметрах методов и хранимых процедур. Обмен реквизитами на уровне приложения позволяет:

- осуществлять выборку предназначенных только для конкретных пользователей данные, для чего применяются доверенные параметры запроса:

```
SELECT x, y FROM SomeTable WHERE username='Jane'
```

- реализовать собственный механизм аудита на любом из уровней приложения.

Обмен реквизитами на уровне ОС позволяет:

- использовать встроенные в ОС возможности аудита (например, аудит Windows и SQL Server);
- выполнять авторизацию каждого пользователя на основе Windows-реквизитов.

Чтобы передавать реквизиты на уровне операционной системы, можно использовать модель олицетворения и делегирования. В некоторых случаях удается реализовать делегирование Kerberos, но если платформа его не поддерживает, подойдут и другие методы, например базовая аутентификация. При использовании базовой аутентификации реквизиты пользователя доступны серверному приложению и могут использоваться для организации доступа к сетевым ресурсам, расположенным ниже в иерархии.

Разработка стратегии авторизации в компонентах пользовательского интерфейса

Основная функция компонентов пользовательского интерфейса — отображать информацию на экране и принимать вводимые пользователем данные. Авторизация на этом уровне выполняется, если необходимо скрыть или, наоборот, показать пользователю определенные поля, включить или отключить элементы управления.

Авторизация в компонентах пользовательского интерфейса

Если пользователь не должен видеть некую информацию, то лучше всего вообще не передавать ее компонентам презентационного уровня.

Практикуется выполнение определенной персонификации пользовательского интерфейса или меню с тем, чтобы пользователь видел только те панели, Web-элементы или пункты меню, которые доступны для его роли. Приложение, как правило, активизируется при запуске исполняемого файла пользовательского интерфейса. Необходимо установить разрешения на доступ для кода сборки пользовательского интерфейса, если требуется предотвратить доступ к важным ресурсам, таким, как файлы, из пользовательского интерфейса.

Совет по планированию Проанализируйте контекст безопасности, в котором работают презентационные компоненты приложения и протестируйте их работу в условиях соответствующих ограничений.

Авторизация в компонентах пользовательских процессов

Компоненты пользовательских процессов управляют данными и потоками управления между пользовательскими процессами. Выполняйте авторизацию на этом уровне, если необходимо:

- * контролировать способность пользователя инициировать процесс взаимодействия с пользовательским интерфейсом;
- добавлять или удалять отдельные фрагменты или целые компоненты пользовательского интерфейса в зависимости от пользователя. Например, каждый торговый представитель должен иметь доступ только к данным по своему региону, поэтому следует исключить для них стадию выбора региона.

Компоненты пользовательских процессов, как правило, вызываются только из компонентов пользовательского интерфейса. Вы можете реализовать механизм управления доступом из кода, чтобы ограничить круг компонентов, имеющих право обращаться к таким компонентам. Кроме того, он годится для ограничения взаимодействия компонентов пользовательских процессов друг с другом. Это особенно актуально в порталах, где очень важно, чтобы пользовательский процесс, реализованный в виде подключаемого модуля (add-in), не мог получать не предназначенную для него информацию из других пользовательских процессов и элементов управления.

Разработка стратегии авторизации в бизнес-компонентах

Вы можете реализовать авторизацию в бизнес-компонентах. При этом следует руководствоваться определенными принципами.

- Постарайтесь сделать авторизацию бизнес-процессов независимой от контекста пользователя, особенно если планируются несколько механизмов связи, например очереди и Web-сервисы.
- Где только возможно применяйте защиту, основанную на ролях, и старайтесь не полагаться на реквизиты пользователя. Так вы обеспечите большую масштабируемость, упростите администрирование и избежите проблем с множественностью канонических представлений имен пользователей. Для компонентов, обращающихся к сервисам, можно определить роли в соответствующем приложении Enterprise Services, а для компонентов .NET, работающих не под управлением Enterprise Services, допускается применять группы Windows или нестандартные (пользовательские) роли.

Разработка стратегии авторизации в компонентах доступа к данным

Компоненты доступа к данным — это последний предоставляющий бизнес-функции уровень перед уровнем данных приложения. Вам следует выполнять авторизацию на этом уровне в случаях, когда, к примеру, приходится совместно использовать компоненты доступа к данным с разработчиками бизнес-процессов, которым вы не полностью доверяете и если предпочитаете защитить доступ к мощным (и, поэтому, небезопасным) функциям источников данных.

Если вы выбрали аутентификацию Windows, то для поддержки авторизации рекомендуется применять роли Enterprise Services и атрибуты *Principal Permissions* в .NET. А роли и атрибуты .NET предпочтительнее, когда контекст безопасности Windows не используется.

Если исходный контекст клиента распространяется и на источник данных, можно применить функции авторизации базы данных, например разрешить или запретить доступ к хранимым процедурам.

Поскольку компоненты доступа к данным, как правило, вызываются только другими компонентами приложения (не напрямую), разумно ограничить круг сборок, которым разрешено вызывать эти компоненты — в основном это совокупность сборок, содержащих компоненты пользовательского интерфейса, бизнес-процессы и бизнес-объекты (если таковые есть в системе).

Разработка стратегии аутентификации в компонентах пользовательского интерфейса

Аутентификация пользователя в компонентах пользовательского интерфейса необходима, если приложение поддерживает авторизацию, аудит или персонификацию.

Аутентификация в пользовательском Web-интерфейсе

Для Web-интерфейсов доступен широкий спектр механизмов аутентификации. При выборе рекомендуем руководствоваться информацией на рис. 9-5.

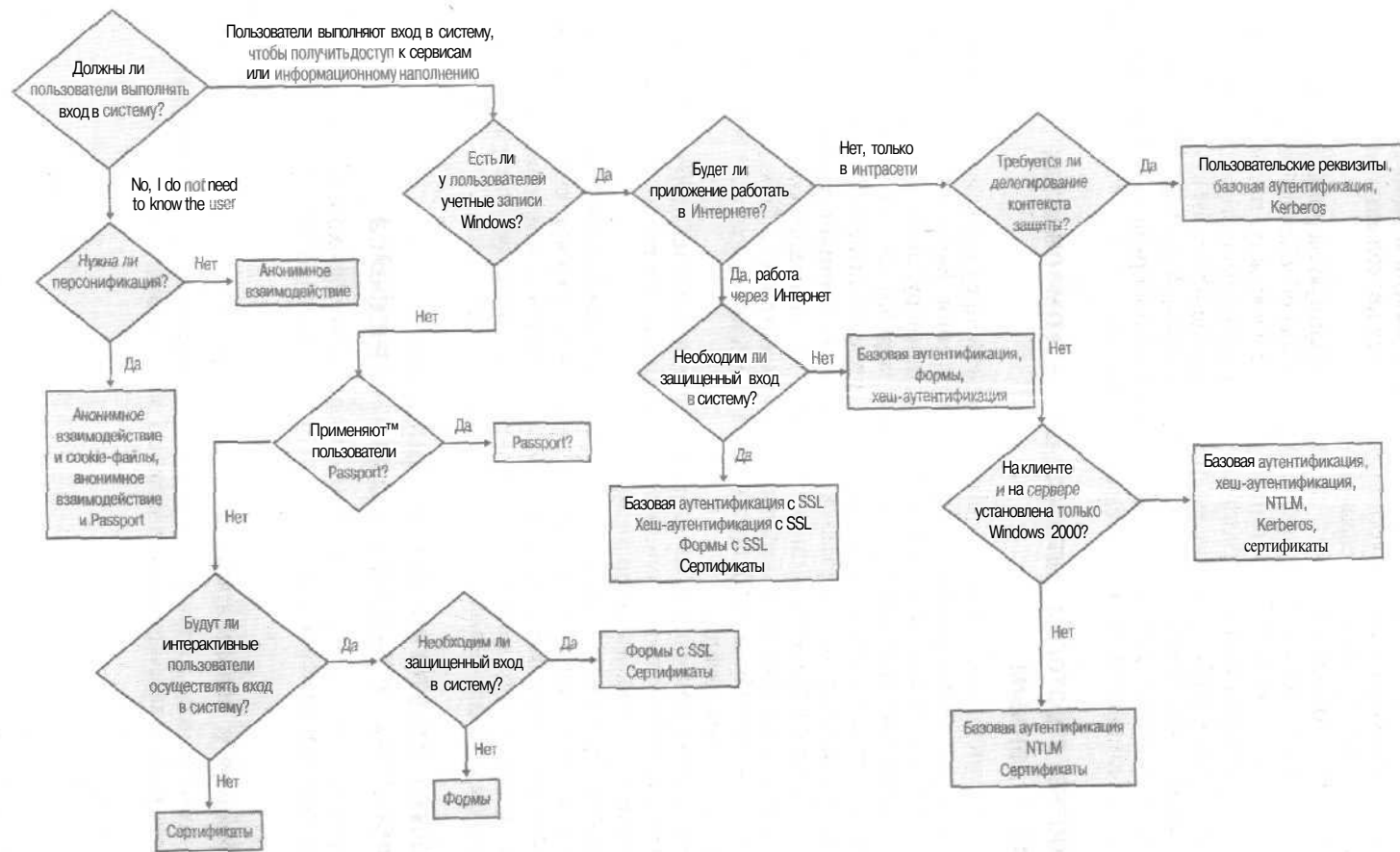


Рис. 9-5. Аутентификация в ASP.NET

Аутентификация в приложении со стандартным пользовательским Windows-интерфейсом

Такой интерфейс, как правило, предполагает собственный механизм аутентификации (приложение запрашивает имя пользователя и пароль) или в нем применяется системная аутентификация пользователей.

Аутентификация в компонентах пользовательских процессов

Компоненты пользовательских процессов не выполняют аутентификацию, а полагаются на контекст защиты, созданный при запуске приложения.

Компоненты пользовательских процессов должны запускаться в том же контексте, что и пользовательский интерфейс, поэтому все связанные с аутентификацией задачи делегируются пользовательскому интерфейсу или даже инфраструктуре, осуществляющей визуализацию интерфейса. В частности, в ASP.NET любой запрос к ASPX-странице приводит к запросу сервером ИС данных для аутентификации или перенаправлению пользователя на страницу с формой аутентификации. Эта операция выполняется «прозрачно» на любом уровне пользовательских процессов и не нарушает поддержку состояния, даже если аутентифицированный сеанс истек и нуждается в переустановке.

Разработка стратегий аутентификации в компонентах доступа к данным

Компоненты доступа к данным предназначаются для использования другими компонентами приложения или сервисами, а также для вызова из сценариев или других приложений. Поэтому при проектировании этих компонентов рекомендуется полагаться на контекст защиты, созданный вызывающей стороной, или на механизм аутентификации, применяемый при удаленном взаимодействии. Компоненты доступа к данным проходят аутентификацию при доступе к базе данных по одной из двух возможных схем: на основе учетных записей сервисов или через олицетворение вызывающей стороны.

Учетные записи сервисов

Обычно предусматривают одну или несколько учетных записей сервисов — набор, представляющий роли или типы пользователей. В большинстве случаев достаточно одной учетной записи, большее их количество необходимо, только если требуется дополнительный контроль над авторизацией. Например, в приложении обработки заказов доступ к базе данных возможен от имени учетной записи *TheOrderApplication* или отдельно; от *OrderProcessingManager* или *OrderProcessingClerk* — в зависимости от роли пользователя.

Учетные записи сервисов следует применять, если:

- необходимо подключение к источнику данных из среды, в которой невозможно олицетворение инициатора цепочки вызовов (например, Microsoft BizTalk Server);
- весьма ограничены возможности контроля изменения учетных записей, способных выполнять вход в другие системы. Пример: вход в реляционную СУБД, которая жестко контролируется администратором баз данных;
- хранилище данных, к которому осуществляется доступ, имеет механизм аутентификации, отличный от применяемого в приложении. Пример: вход в Web-сервер через Интернет.

Не используйте сервисные учетные записи, если:

- нет безопасного способа хранения и поддержки реквизитов сервисов;
- действующие политики безопасности требуют отдельного доступа к пользовательским данным в хранилище. Пример: необходим доступ к данным или объектам в SQL Server от имени пользователей;
- поддерживается аудит хранилища данных с привязкой к отдельным пользователям.

Олицетворение вызывающего

Олицетворение вызывающего выполняется, если осуществляется доступ к хранилищу данных от имени учетных записей, которые «один-к-одному» соответствуют пользователям приложения. Например, если Джейн выполнила вход в приложение, то при обращении компонентов доступа к данным они олицетворяют Джейн, если вход осуществляется с ее реквизитами.

Олицетворение вызывающего необходимо, если:

- хранилище данных выполняет авторизацию на основе личности вошедшего в систему пользователя;
- хранилище данных должно выполнять отдельный аудит действий пользователей.

Разработка стратегии аудита

Для обеспечения поддержки аудита бизнес-операций необходимо наличие защищенного хранилища. В сущности, аудит можно рассматривать как защищенное ведение журнала событий. Создавая собственную реализацию системы аудита, вы должны позаботиться о том, чтобы записи журнала оказались недоступными для изменения посторонними или по крайней мере чтобы такие изменения легко обнаруживались (для этого годится электронная цифровая подпись), и о защите места хранения журнала (например, обеспечить невозможность изменения строк подключения и/или файлов-хранилищ). В частном механизме аудита вы вправе использовать подписание документов, встроенную в ОС аутентификацию и защиту доступа из кода, что предотвратит внесение подложных данных в журнал.

Аудит в компонентах пользовательского интерфейса и пользовательских процессах

Операции компонентов пользовательского интерфейса, как правило, не подвергаются аудиту. Исключение составляют глобальные события, такие, как вход и выход из системы, изменение пароля и все связанные с безопасностью исключения.

Поскольку компоненты пользовательских процессов представляют выполняемые пользователями операции, которые часто останавливаются, прерываются, не получают продолжения и т.д., они редко подвергаются аудиту. Однако вы вправе регистрировать все исключения, связанные с безопасностью.

Аудит в компонентах бизнес-процессов

Наиболее часто выполняется аудит бизнес-процессов, так как очень важно знать, кто и когда выполнял ключевые бизнес-операции.

Если события регистрируются в контексте транзакции в базе данных, поддерживающей транзакции (например, SQL Server), компонент аудита должен создавать новую транзакцию, чтобы сбой в дереве исходной транзакции не вызвал откат записи в журнал аудита.

Аудит в компонентах доступа к данным

Компоненты доступа к данным — уровень бизнес-логики, наиболее близко расположенный к хранилищу данных, поэтому это хорошее место для реализации аудита.

Компоненты доступа к данным, как правило, вызывают хранимые процедуры, которые собственно и выполняют всю связанную с данными работу. Вот почему многие предпочитают выполнять аудит прямо в СУБД.

Примечание Дополнительную информацию о реализации аудита в SQL Server вы найдете в статье «Auditing SQL Server Activity» в комплекте ресурсов SQL Server 2000 SDK на Web-сайте MSDN (<http://msdn.microsoft.com>).

Практикум. Моделирование опасностей и меры по их предотвращению



При выполнении упражнений используйте знания, полученные на занятиях.

Упражнение 1. Определение потенциальных опасностей

В компании Adventure Works Cycles клиенты создают заказы двумя способами. Во-первых, через открытый Web-сайт, действуя точно так же, как и покупатели при размещении заказов. Однако торговые представители должны вводить свои реквизиты, чтобы предоставлять клиентам скидки. Второй метод — посредством портативных компьютеров (наладонников) и ноутбуков с приложениями Windows Forms. Торговые представители по продажам создают заказ на наладоннике либо на ноутбуке, но подпись клиента получают с помощью наладонника. Затем торговые представители подключаются к сети и загружают информацию в приложение ввода заказов.

В обоих случаях безопасность заказов подвергается различным опасностям. Откройте файл *C09Ex1.ppt* из папки `\SolutionDocuments\Chapter09` на прилагаемом к книге компакт-диске и изучите высокоуровневые диаграммы архитектуры приложения Windows Forms. Используя модель STRIDE, определите, что грозит приложению.

Одно из возможных решений вы найдете в файле *C09Ex1_Answer.ppt* из той же папки.

Упражнение 2. Применение технологий для предотвращения опасности

Используя обнаруженные при выполнении первого упражнения опасности, перечислите технологии, применение которых позволит им предотвратить или смягчить возможные отрицательные последствия.

Одно из возможных решений вы найдете в файле *C09Ex2_Answer.ppt* в папке `\SolutionDocuments\Chapter09`.

Резюме

- Наиболее типичными уязвимыми местами приложений считаются:
 - слабые пароли;
 - неправильно сконфигурированное ПО;
 - уязвимость по отношению к методам социальной инженерии;
 - подключения к Интернету;
 - пересылка данных открытым текстом;
 - переполнение буфера;
 - внедрение SQL;
 - секретные данные, жестко прописанные в коде.
- Некоторые принципы стратегии безопасности:
 - полагайтесь только на оттестированные и проверенные системы безопасности;
 - никогда не доверяйте данным, полученным извне;

- по умолчанию считайте внешние системы незащищенными;
- придерживайтесь принципа наименьших привилегий;
- минимизируйте количество доступных компонентов и данных;
- по умолчанию используйте безопасные параметры;
- не полагайтесь на защиту, основанную на умалчивании информации;
- следуйте принципам STRIDE.
- Модель STRIDE — это модель опасностей. Каждая буква в сокращении STRIDE обозначает определенную категорию угрозы безопасности: *подмена сетевых объектов* (spoofing identity), *модификация данных* (tampering), *отказ от причастности* (repudiation), *раскрытие информации* (information disclosure), *отказ в обслуживании* (denial of service) и *повышение привилегий* (elevation of privilege).
- Последовательность действий при создании модели опасностей:
 - проведите мозговые штурмы;
 - классифицируйте опасности по категориям STRIDE;
 - перечислите все возможные опасности;
 - создайте комментарии;
 - проведите исследование;
 - распределите риски, связанные с каждой опасностью, по ранжиру.
- .NET Framework поддерживает следующие функции защиты:
 - контроль типов;
 - подписание кода;
 - шифрование и подписание данных;
 - безопасность доступа из кода;
 - основанная на ролях защита;
 - изолированное хранилище.
- Разработка стратегии аутентификации и авторизации в приложении выполняется в несколько этапов:
 - определение ресурсов;
 - выбор стратегии авторизации;
 - выбор объектов и реквизитов, на основании которых определяется доступ к ресурсам;
 - организация пересылки реквизитов;
 - выбор способа аутентификации;
 - определение потоков идентификационных данных в системе.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Каковы недостатки традиционных моделей безопасности?
2. Каковы принципы создания защищенного кода?
3. Из перечисленных ниже высказываний выберите те, что применимы к переполнению буфера:
 - контроль типов предназначен для предотвращения переполнения буфера;
 - из-за переполнения буфера приложение зачастую перестает отвечать на запросы или неправильно работает;
 - злоумышленники могут использовать переполнение буфера для запуска произвольного кода;
 - сообщение об ошибке из-за переполнения буфера может представлять угрозу безопасности приложения.
4. На каком из этапов MSF необходимо создавать модель опасностей:
 - при планировании;
 - при разработке;
 - при стабилизации?
5. Что такое модель STRIDE?
6. Ознакомьтесь с приведенным сценарием атаки и определите, по каким категориям STRIDE следует классифицировать возникающие в данной ситуации опасности.

Карл заметил, что Боб оставил свой компьютер без присмотра и не заблокировал его. Карл сел за компьютер Боба и открыл приложение для работы с электронной почтой. Он отправил Алисе письмо от имени Боба. Затем он закрыл почтовый клиент и покинул рабочее место, причем и его действия, и уход остались незамеченными.
7. Что такое защита доступа из кода?
8. Что такое основанная на ролях защита?
9. Какие провайдеры аутентификации поддерживает ASP.NET?
10. Какие три типа защиты обеспечивают Web-сервисы?
11. Перечислите стадии разработки стратегии авторизации и аутентификации в приложении.

ГЛАВА 10

Завершение этапа планирования

Занятие 1. Реализация требований к проекту	302
Занятие 2. Планирование функций администрирования	316
Занятие 3. Планирование будущих этапов	319
Занятие 4. Создание технической спецификации	324
Практикум. Анализ плана тестирования и технической спецификации	325
Резюме	326
Закрепление материала	328

В этой главе

На этапе планирования создается львиная доля архитектуры и дизайна решения. Результаты этого этапа — планы разработки и развертывания решения, а также графики использования ресурсов и выполнения задач. В этой главе рассказывается о планах и задачах, над которыми приходится работать проектной команде на этапе планирования проекта.

Прежде всего

Для изучения материалов этой главы необходимо:

- знать модели процессов MSF;
- уметь создавать варианты и сценарии использования системы и диаграммы ВИС;
- знать результаты этапа создания общей картины решения, такие, как цели и область применения, структура проекта и анализ рисков;
- знать задачи и результаты трех стадий этапа планирования — концептуального, логического и физического дизайна.

Занятие 1. Реализация требований к проекту

На этапе планирования на дизайн приложения влияет масса обстоятельств. Некоторые из них неизменны и ограничены, в том числе время, бюджет и рабочая сила. Другие динамично меняются в течение всего цикла разработки — технологии, знания и навыки. Эти обстоятельства, несомненно, оказывают влияние на дизайн решения, но главный критерий — решение бизнес-задачи, и именно оно определяет функциональное наполнение и успех или провал готового решения.

На этом занятии вы узнаете о функциональных возможностях и задачах, которые должна решать создаваемая система: о факторах, которые учитываются при проектировании — о масштабируемости, доступности, надежности, производительности, совместимости, поддержке многих языков и локализации. Здесь также рассказывается о методах реализации этих требований.

Изучив материал этого занятия вы сможете:

- ✓ модернизировать дизайн для обеспечения:
 - масштабирования;
 - доступности;
 - надежности;
 - производительности;
 - способности к взаимодействию;
- ✓ создать спецификации для локализации и поддержки многих языков в создаваемой системе.

Продолжительность занятия — около 25 минут.

Как спроектировать масштабируемое приложение

Масштабируемость определяется как возможность расширять сервисы системы за счет увеличения системных ресурсов. При этом масштабируемое приложение обслуживает больше запросов, а никаких изменений самого приложения не требуется.

В масштабируемом приложении необходим определенный баланс между программным и аппаратным обеспечением. Для повышения масштабируемости приложения увеличивают программные и/или аппаратные ресурсы. Расширение ресурсов обычно дает эффект, однако иногда результат оказывается нулевым или даже отрицательным: возможности приложения по предоставлению сервиса не увеличиваются или даже падают. Допустим, вы решили использовать балансировку нагрузки. Это вряд ли принесет ожидаемый эффект, если приложение основано на синхронных вызовах методов или при обслуживании пользовательских запросов выполняет выборку больших наборов данных.

Стандартные методы масштабирования

- **Вертикальное масштабирование (scaling up)** предусматривает совершенствование оборудования сервера, например увеличение объема оперативной памяти, установку более быстрых процессоров или увеличение их количества, а также перенос приложения на более производительный компьютер. Основная цель вертикального масштабирования — предоставить в распоряжение

приложения больше аппаратных ресурсов. Как правило, при этом не требуется менять исходный код приложения. Процедуры администрирования также практически не меняются. Однако следует иметь в виду, что эффект от вертикального масштабирования сходит на нет при достижении предела загруженности имеющихся (хотя и значительных) ресурсов.

На рис. 10-1 показана способность приложения обслуживать клиентские запросы при вертикальном масштабировании.

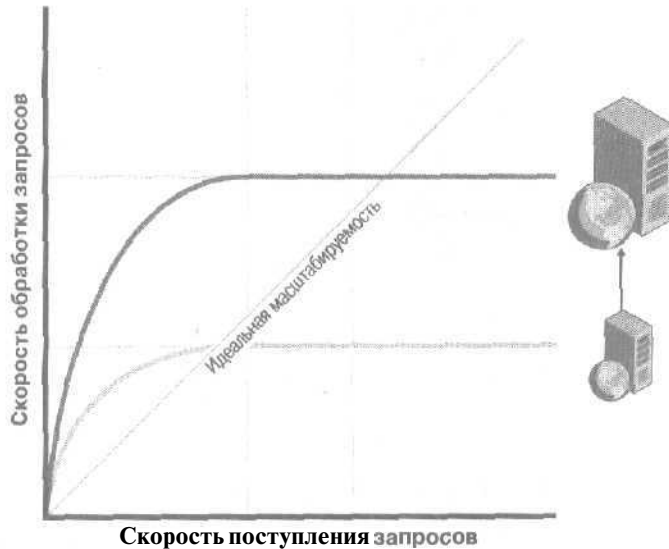


Рис. 10-1. Вертикальное масштабирование приложения

- **Горизонтальное масштабирование** (scaling out) предусматривает распределение нагрузки между несколькими серверами. Хотя при этом задействовано несколько компьютеров, для конечного пользователя они выглядят как один. Как и в предыдущем случае, очень важен баланс между программным и аппаратным обеспечением. Приложение должно выполняться, не требуя информации о сервере, на котором оно установлено. Эта концепция получила название *прозрачность размещения*. Горизонтальное масштабирование также позволяет повысить отказоустойчивость приложения.

Рис. 10-2 иллюстрирует влияние эффекта горизонтального масштабирования на способность приложения обслуживать растущие объемы запросов.

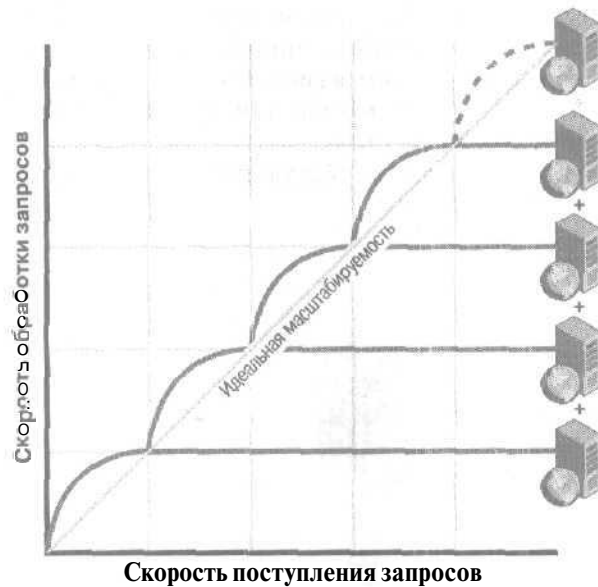


Рис. 10-2. Горизонтальное масштабирование приложения

Проектирование масштабируемости

Качественный дизайн — основа высокой масштабируемости приложения. Именно на этапе планирования закладываются основы масштабируемости приложения.

На рис. 10-3 показана роль дизайна, оптимизации кода, самого приложения и настройки аппаратного обеспечения в масштабируемости приложения. Наибольшее влияние на масштабируемость оказывает дизайн. По мере продвижения к вершине пирамиды степень влияния указанных факторов снижается, но на иллюстрации видно, что качественный дизайн влияет на масштабируемость гораздо сильнее, чем расширение аппаратных ресурсов.

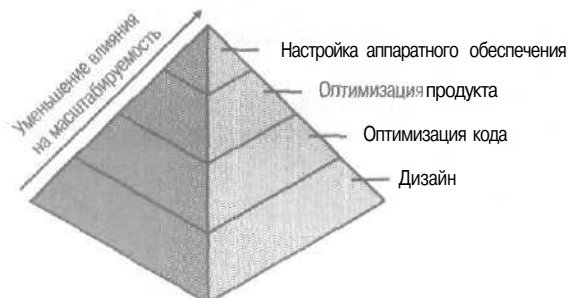


Рис. 10-3. Пирамида масштабируемости

Для обеспечения масштабируемости необходимо следовать нескольким рекомендациям.

- **Процессы не должны ожидать отклика дольше, чем необходимо.** Процессы делятся на синхронные и асинхронные. Чтобы продолжить работу после обращения к другому процессу, синхронный процесс должен дожидаться отклика,

то есть успешного или неудачного окончания другого процесса или процессов. В приложении на основе синхронных процессов идет борьба за ресурсы, которые часто и становятся узким местом. Это ограничивает как производительность, так и масштабируемость приложения. Значительно легче добиться масштабируемости, используя асинхронные процессы. В приложении на основе асинхронных процессов «долгоиграющие» операции откладываются или выносятся в отдельный процесс.

- **Процессы не должны конкурировать за ресурсы.** Один из самых серьезных «врагов» масштабируемости — борьба за ресурсы: память, процессорное время, полосу пропускания сети и подключения к базе данных. Планируйте использование ресурсов так, чтобы предотвратить подобные накладки:
 - преимущество отдавайте наиболее доступным и емким ресурсам, а дефицитные ресурсы используйте во вторую очередь;
 - обращайтесь к ресурсу как можно позже, а освобождайте — как можно раньше. Чем меньше времени процесс «владеет» ресурсом, тем он доступнее для других процессов.
- **Следует добиваться коммутативности процессов.** Две или более операции считаются коммутативными, если от изменения порядка их выполнения результат не меняется. Как правило, легко коммутативны операции, не задействованные в транзакциях. Если на загруженном сайте электронной коммерции постоянно обновлять информацию о доступных на складе товарах, возможна блокировка записей в базе данных из-за конкуренции запросов. Чтобы избежать этого, информацию об изменениях на складе следует заносить в отдельную таблицу транзакций склада. База данных должна периодически добавлять строки в эту таблицу, а затем сводные сведения о товарах со всеми изменениями вносятся в базу данных склада.
- **Следует создавать взаимозаменяемые компоненты.** Такие компоненты освобождают ресурсы, перемещаются в пул диспетчером ресурсов и инициализируются снова для использования новым клиентом. Разрабатывать компоненты следует так, чтобы никакая клиентская информация не попадала к другому клиенту. Кроме того, компонент должен поддерживать агрегирование и не привязываться к конкретному потоку. В схемах с созданием пула ресурсов, например пулов компонентов COM+ и ODBC-подключений, применяются взаимозаменяемые ресурсы. Вы можете использовать оснастку Component Services (Службы компонентов) для создания пула объектов, определения минимального и максимального размеров пула, а также для настройки таймаута. Более подробно эта тема обсуждается в статье «Improving Performance with Object Pooling» из документации к комплекту ресурсов Platform SDK MSDN.
- **Необходимо разделять ресурсы и сервисы (службы).** Необходимо минимизировать связь между ресурсами и сервисами путем их разделения — это позволяет предотвратить борьбу за ресурсы. Разделение сервисов также снижает нагрузку на критические ресурсы — процессор и полосу пропускания сети. Например, использование SSL для обеспечения безопасной связи создает заметную нагрузку на систему. Поэтому применять SSL рекомендуется на тех страницах, где действительно необходим высокий уровень безопасности, а на остальных для обслуживания SSL-сеансов используйте специальные Web-серверы. Для разделения ресурсов и сервисов можно создать множество маленьких модулей вместо нескольких больших и ограничить межмодульное взаимодействие. Од-

нако деление иногда усложняет систему. При излишнем разделении зависимых ресурсов увеличивается дополнительная нагрузка на систему.

Создание приложения высокой доступности

Хотя большинство приложений доступны не всегда, Web-приложения и критически важные корпоративные приложения должны работать (предоставлять свои сервисы) непрерывно. Если предполагается, что приложение будет работать 24 часа в сутки и 7 дней в неделю, высокую степень доступности следует заложить в дизайн. Благодаря совершенствованию аппаратного и программного обеспечения создание высокодоступных приложений значительно упростилось.

Определение доступности

Доступность — это мера отказоустойчивости приложения, то есть соотношение времени, когда приложение обслуживает запросы, к запланированному времени его работы. Доступность также учитывает время технологических простоев, поскольку в это время приложение не доступно.

Примечание Доступность не имеет отношения к обеспечению непрерывности бизнес-процессов, в частности к созданию резервных копий или альтернативных сайтов.

В таблице 10-1 перечислены исходные данные, которые используются для расчета степени доступности приложения.

Таблица 10-1. Данные, используемые для расчета доступности

Исходная информация	Единицы измерения	Определение
Средняя наработка на отказ (Mean Time Between Failures, MTBF)	Часы/число сбоев	Средний промежуток бесперебойной работы
Среднее время восстановления (Mean Time To Recovery, MTTR)	Часы/число сбоев	Средний промежуток времени, необходимый для ремонта и восстановления службы после сбоя

Формула для расчета степени доступности:

$$\text{доступность} \approx (\text{MTBF} / (\text{MTBF} + \text{MTTR})) \times 100$$

Например, стандартные требования по степени доступности для сайта компании Adventure Works Cycles — 24 часа в день и 7 дней в неделю. Если принять 1000 часов непрерывной работы за исходную точку, то при двух одночасовых сбоях в течение этого промежутка времени доступность составит:

$$((1000 / 2) / ((1000 / 2) + 1)) \times 100 = (500 / 501) \times 100 = 0,998 \times 100 = 99,8\%.$$

Обычно доступность измеряют *девятками*, например три девятки соответствуют доступности 99,9%. Однако, применять такой способ следует очень аккуратно, так как легко ошибиться и недооценить жесткость требований. Подсчет показывает, что три девятки (99,9%) составляют примерно 8,5 часов простоя за год. Следующий уровень, четыре девятки (99,99%) — это всего час, а пять девяток (99,999%) — 5 минут простоя в году.

Планирование уровней доступности

Чтобы определить необходимый уровень доступности, нужно ответить на несколько вопросов.

- Кто пользователи приложения и что они ожидают от приложения?
- Какое время простоя допустимо?
- Зависят ли от сервиса какие-либо внутрикорпоративные процессы?
- Каковы сроки и бюджет, выделенный для разработки приложения?

Реализация требований по доступности

Доступность — это упреждение, выявление и разрешение программных и аппаратных неполадок до того, как они приведут к сбою в работе сервисов или повреждению данных. Для обеспечения высокой доступности необходимо резервирование прикладных сервисов и данных, а процедуры поддержки и сопровождения приложения (как автоматизированные, так и ручные) должны быть проверенными и надежными.

Необходимо сократить не только аварийные, но и плановые простои. Последние обычно **требуются** для внесения изменений, обновления операционной системы, создания резервных копий или любых других жизненно важных операций, для которых необходима остановка приложения.

Доступность приложения также зависит от его надежности. Надежными и доступными бывают приложения, имеющие качественный фундамент: корректный дизайн и прошедшие всестороннее тестирование и сертификацию. Вот некоторые из методик, применяемых для обеспечения доступности.

- **Сокращение времени простоя.** Для этого применяется *«горячее» обновление* (rolling upgrade), при котором перед обновлением компонента кластерного сервера группа серверных ресурсов переносится на другой сервер, обновляемый сервер отключается, выполняется обновление компонента, а затем он снова возвращается в сеть. В это время нагрузка распределяется между другими серверами, и работа приложения не прерывается. Подобный метод обычно применяется в приложениях, поддающихся вертикальному масштабированию.
- **Сокращение внеплановых простоев за счет кластеризации** — технологии, применяемой для создания высокодоступных приложений. **Кластер** — это несколько компьютеров, соединенных сетью и логически объединенных кластерным ПО. Благодаря кластеризации многосерверный Web-сайт противостоит сбоям и работает непрерывно. При сбое активного сервера, его нагрузка автоматически переносится на пассивный сервер, текущие клиентские процессы переключаются, а потерпевший сбой сервис автоматически перезапускается. Даже в случае недоступности ресурса пользователи, подключенные к серверному кластеру, испытывают определенную задержку, **однако** обслуживание не прекращается. Кластерное ПО обеспечивает *переход при сбое* (failover) для приложений, файловых служб и служб печати, баз данных и систем обмена сообщениями, которые поддерживают работу в кластерных системах.
- **Балансировка сетевой нагрузки** применяется для равномерного распределения трафика между доступными серверами. Она также помогает улучшить доступность приложения: если сервер выходит из строя, сервис балансировки нагрузки автоматически перегруппирует кластер и направит трафик на другие серверы. Этот сервис особенно полезен в приложениях электронной коммерции,

которые поддерживают транзакционное взаимодействие внешних клиентов с серверами баз данных. При росте клиентского трафика выполняется вертикальное масштабирование Web-сервера, причем в кластере может быть до 32-х серверов. Сервис балансировки сетевой нагрузки автоматически выявляет сбойные серверы и перенаправляет клиентский трафик на действующие серверы, обеспечивая постоянную работоспособность клиентских сервисов.

- **RAID-массивы для хранения данных.** В них данные многократно дублируются, а при отказе одного из дисков приложение автоматически переадресуется на зеркальный диск и продолжает работу. Сбойный диск заменяется без остановки приложения.
- **Изоляция критически важных приложений.** В процессе своей работы приложение обращается к ресурсам — сетевым подключениям, данным или процессам. Отсутствие или перегрузка любого из этих ресурсов отрицательно сказывается на производительности и доступности приложений, которым он необходим. Если много приложений одновременно используют сервисы одних серверов, загруженность и пропускная способность этих серверов меняется не в лучшую сторону. Поэтому критически важные приложения рекомендуются размещать в выделенной инфраструктуре и в закрытых частных сетях.
- **Очереди** позволяют приложениям взаимодействовать с использованием асинхронных сообщений. Очереди гарантируют доставку сообщений независимо от наличия связи на тот момент (например, в мобильных приложениях), а само приложение перестает быть точкой возможного сбоя. Очереди сообщений также помогают справляться с пиковыми нагрузками и, таким образом, избавляют от, казалось бы, неизбежных затрат на приобретение дополнительного оборудования. Кроме того, увеличив число возможных маршрутов доставки сообщений, вы повысите шансы их удачной доставки и обработки.

Создание надежного приложения

Надежность приложения означает его способность обеспечивать адекватный результат и тесно связана с доступностью. Отличие в том, что доступность — это способность системы успешно обрабатывать все запросы и «без потерь» восстановиться после сбоя, а надежность — это мера того, как долго приложение способно работать и давать ожидаемые результаты без сбоя. Пользователи не жалуют ненадежный Web-сайт, что немедленно сказывается на его доходности и объеме будущих продаж. Кроме того, большие затраты на восстановление поврежденных данных лишь усугубляют убытки, возникающие из-за сбоев приложения. Ненадежные системы трудно сопровождать и развивать, поскольку точки сбоев, как правило, трудно обнаружить.

Причины сбоев в приложениях

Система — это совокупность аппаратных ресурсов, сервисов операционной системы, программных компонентов и процессов реального мира, которые все вместе обеспечивают функционирование бизнес-сервисов. Надежность системы в целом зависит от надежности каждого из компонентов, а сбой в одном компоненте немедленно сказывается на надежности остальных компонентов.

Есть масса причин сбоев в приложении:

- несовершенный, полный ошибок код;
- недостаточно тщательное тестирование;
- недостатки системы управления изменениями;

- ошибки эксплуатации;
- отсутствие постоянного мониторинга и анализа его результатов;
- отсутствие контроля качества в процессе производства ПО;
- взаимодействие с внешними службами или приложениями;
- изменение условий работы, например уровень загруженности и нагрузка;
- непредвиденные события, такие, как сбои в системе защиты и атаки по Широковещательным каналам;
- аппаратные сбои (отказы дисков, контроллеров, сетевых устройств, серверов, источников питания, памяти и/или процессоров);
- влияние окружающей среды (отключение питания и/или охлаждения, высокая запыленность, пожары, наводнения и другие стихийные бедствия).

Обеспечение надежности проекта

Для обеспечения надежности необходимо определить предполагаемый уровень использования приложения, создать профиль надежности и разработать решение, удовлетворяющее этому профилю. Необходимо понять, как обеспечиваются те или иные сервисы, описать и оценить сценарии возможных сбоев и разработать предпочтительные альтернативы. Помимо этого следует учесть взаимодействие продукта с другими приложениями.

Трудно определить проблемы с надежностью и способы их решения в еще не созданной системе. Можно начать с анализа уже работающих приложений. Это позволит определить частоту и места возникновения сбоев, их основные причины и возможные пути улучшения существующих систем. Такая информация пригодится при разработке более надежного решения.

Надежное решение должно предотвращать ошибки при вводе и преобразовании данных, управлении состояниями, а также обеспечивать восстановление без потерь после любого сбоя. Надежность приложения напрямую зависит от всех стадий цикла разработки ПО — от этапа планирования, разработки и тестирования до развертывания и стабилизации. Ряд мероприятий поможет создать надежное приложение:

- отражение в спецификации требований к надежности;
- использование качественной архитектурной инфраструктуры;
- включение в приложение информации, необходимой для управления;
- реализация избыточности и дублирования;
- применение инструментов контроля качества;
- проверка надежности приложения;
- реализация обработки ошибок;
- создание архитектуры приложения, которая предусматривает в случае сбоя отключение лишь некоторых функций, а не крах всего приложения.

Проектирование производительности

Производительность приложения измеряется определенными метриками, например количеством операций в единицу времени или степенью использования ресурсов. С точки же зрения пользователя это время отклика системы.

Цели и критерии производительности

До реализации требований по производительности в проекте приложения необходимо разобраться с целями и метриками оценки производительности. Чтобы определить цели, следует ответить на ряд вопросов.

- **Какова цель с точки зрения бизнеса.** Например, если бизнес-цель заключается в обработке заказов, количество которых увеличивается каждую неделю, начните с ожидаемого увеличения дохода и на основании этого определите цели по производительности в каждой из функциональных областей.
- **Какие функции являются критически важными.** Определение критически важных функций позволяет расставить приоритеты при проектировании решения. Например, пойти на снижение производительности менее важной функции ради сохранения или повышения производительности функции исключительной важности.
- **Какие функциональные возможности нужны разным группам пользователей.** Вы можете создать профили в соответствии с различными требованиями организации и конечных пользователей. Требования к производительности приложения будут различаться, поэтому важно определить связи между всеми функциональными областями и целями производительности. Допустим, вся информация о размещенных клиентами заказах хранится в базе данных. С точки зрения клиента приложение должно быстро обновлять информацию в базе данных. Компании же необходимо, чтобы система быстро сохраняла корректные данные. Следовательно, цель по производительности — обеспечить быструю вставку и обновление данных в базе. Профилирование облегчает деление БД на разделы и разработку адекватных тестов.

Примечание С точки зрения тестирования цели, касающиеся производительности, должны выражаться в единицах, поддающихся измерению в процедурах тестирования. Например, быструю вставку и обновление данных можно измерять в транзакциях в секунду.

Реализация требований к производительности в проекте

Требования к производительности необходимо определить до того, как команда займется разработкой. Чтобы точнее определить эти требования, необходимо выяснить ограничения проекта, сервисы, которые будет предоставлять приложение, и степень загрузки приложения.

- **Определение ограничений.** Проект ограничивается бюджетом, календарным планом, инфраструктурой и выбранными средствами разработки или технологии. Например, вы должны развернуть приложение не позднее определенной даты. Возможно, вам понадобится определенное средство разработки, поскольку команда разработки владеет только им. Или ваше приложение не должно интенсивно использовать процессор, так как аппаратные ресурсы клиентских компьютеров оставляют желать лучшего. Ваша задача — разработать приложение, которое удовлетворяло бы требования к производительности и указанным ограничениям. Подумайте, может, удастся добиться цели, изменяя стороны проекта, не подпадающие под ограничения. Например, обучить программистов создавать компоненты с использованием другого инструмента или ускорить доступ к данным, реализовав иную технологию.
- **Определение функций.** Функции приложения связаны с ВИС и СИС. Вы можете определить сценарии использования системы, влияющие на производительность, и для каждого описать действия пользователя и отклик приложения, в том числе и то, как осуществляется доступ к базе данных и иным системным сервисам. Кроме того, необходимо определить, как часто исполь-

зуется та или иная функция. Эта информация поможет разработать тесты для измерения производительности, которые максимально точно отображают реальную работу приложения.

- **Определение нагрузки.** Нагрузку на приложение, к примеру, определяет количество работающих с ним клиентов. Кроме того, обычно выясняют, как нагрузка меняется с течением времени. Например, число запросов сайта электронной коммерции **возрастает** в определенные промежутки времени и течение года. Параметры нагрузки пригодятся для определения единиц измерения производительности приложения.

Разработка приложения, поддерживающего взаимодействие с другими системами

Как правило, в средних и крупных организациях вычислительные среды **гетерогенны**, то есть построены на различных технологиях. Например, многие организации развертывают многоуровневые клиент-серверные приложения, **которым** необходим доступ к данным или транзакциям существующих систем. Кроме того, новому приложению часто требуется взаимодействовать с собственными разработками заказчика или с приложениями третьих фирм.

Почему приложение должно поддерживать взаимодействие с другими системами

Решение должно эффективно взаимодействовать с другими подсистемами, поскольку это позволяет:

- **снизить эксплуатационные расходы и сложность продукта.** Вряд ли в обозримом будущем клиенты сменят свою смешанную среду на однородную, а ведь способность различных систем работать **вместе** в одной среде снижает затраты на разработку и поддержку гетерогенной инфраструктуры;
- **развертывать систему оптимальным способом.** У клиентов могут быть **бизнес-требования**, продиктованные определенными приложениями и платформами. Поддерживающее взаимодействия приложение позволяет **организации** использовать существующие приложения, удовлетворяющие особым требованиям;
- **сохранить инвестиции в ИТ.** На клиентских компьютерах, как правило, уже установлено большое количество разнообразных приложений, и переход на новую платформу **осуществляется** постепенно. Понятно, что новые приложения должны «уметь» взаимодействовать с уже существующими приложениями. Кроме того, в существующих приложениях можно обеспечить поддержку работы в Web и доступ к системам вроде мэйнфреймов IBM из интранета или Интернета. Это расширяет функциональные возможности **существующих** приложений и защищает уже сделанные инвестиции в ИТ-инфраструктуру.

Реализация требований по взаимодействию в проекте

При интеграции гетерогенных приложений необходимо учитывать ряд типов взаимодействия.

- **Сетевое взаимодействие** — возможность взаимодействия систем разных производителей без использования общих протоколов. В прошлом приложения разрабатывались на основе определенных протоколов, таких, как TCP/IP, IPX/SPX или SNA. Поддержка технологий и стандартов Интернета, например

HTTP, XML, SOAP, WSDL и XML Web-сервисов, обеспечивает независимость приложения от языков программирования, платформ и архитектуры устройств.

- **Взаимодействие с различными источниками данных** означает способность приложения обращаться к данным, хранящимся как в структурированных, так и в неструктурированных системах хранения, таких, как базы данных, файловые системы или хранилища сообщений электронной почты. Приложениям масштаба предприятия часто требуется совместный доступ к данным из разнородных источников и приложений. Опубликованные стандарты обмена данными — каскадные таблицы стилей (CSS), ODBC и XML — позволяют обращаться к самым разным источникам, работающим под управлением как Windows, так и других ОС.
- **Взаимодействие приложений** относится к инфраструктуре, необходимой для взаимодействия между новыми многоуровневыми и существующими приложениями, бизнес-логикой и данными. Создавая новые многоуровневые приложения, вы должны обеспечить их совместную работу с большим количеством существующих приложений. Один из способов решения этой задачи — использовать спецификацию общего языка (Common Language Specification, CLS) — стандарт, которому в настоящее время удовлетворяют более двадцати языков программирования и который позволяет взаимодействовать сервисам, созданным на любом из CLS-совместимых языков.
В приложениях на основе Microsoft .NET вы вправе использовать ПО на базе традиционного каркаса ASP и COM, которые пока нет возможности перевести в среду .NET. Также не следует пренебрегать языком XML, который специально создан для обмена данными между приложениями без создания дополнительного кода по преобразованию данных для каждого конкретного случая.
- **Взаимодействия, связанные с управлением**, — это задачи управления учетными данными пользователей, мониторинга производительности и оптимизации гетерогенных приложений в организации.

Реализация в проекте требований по поддержке различных языков и локализации

Обеспечение поддержки многих языков и локализация — это процессы разработки приложений, которые предполагается использовать во многих странах мира. На этапе планирования необходимо четко определить и задокументировать требования по поддержке других языков и локализации с тем, чтобы учесть их в проекте.

Примечание Microsoft .NET Framework содержит три пространства имен, призванных облегчить реализацию поддержки других языков и локализацию. Это *System.Globalization*, *System.Resources* и *System.Text*.

Определение поддержки различных языков

Поддержка других языков означает проектирование и разработку приложения, способного работать с любым языком и стандартами. Она подразумевает:

- определение поддерживаемых языков и стандартов;

- дизайн функций для различных языков и стандартов;
- написание кода, способного правильно поддерживать требуемые языки и стандарты.

При реализации поддержки других регионов используется понятие языка и местных стандартов, в том числе:

- классификации символов;
- формата даты и времени;
- принятого формата чисел, валюты, единиц веса и длины;
- правил сортировки.

Определение локализации

Локализация — это процесс адаптации приложения к конкретному языку и стандартам с использованием характерных для конкретной страны ресурсов. Файл ресурсов содержит характерные для региона элементы пользовательского интерфейса. Он может быть текстовым и обычно имеет расширения *.resx* или *.resource*. Поддерживающее локализацию приложение состоит из двух основных блоков: *данных* и *кода*. Первый (он обычно содержится в файлах ресурсов) содержит все строки пользовательского интерфейса, а второй — код приложения, который не зависит от страны, языка и стандартов и подключает нужный файл ресурсов в зависимости от выбранного региона.

При создании локализованной версии приложения изменяют блок данных и объединяют его с блоком кода, который, как правило, не изменяется. Обычно необходимо базовое понимание используемого набора и всех связанных с ним возможных затруднений. Хотя все компьютеры хранят текст в виде чисел, в различных системах письма один и тот же текст представлен разными числами.

Для успешного создания локализованной версии приложения необходимо позаботиться об:

- отделении блока кода от блока данных;
- корректности чтения кодом приложения данных вне зависимости от региональных параметров,

Особенности поддержки различных языков и локализации

Далее перечислены проблемы и особенности, которые необходимо учитывать при проектировании поддержки различных языков и локализации.

- **Языковая специфика** — особенности алфавита, грамматики, синтаксиса и правил отображения символов в различных языках. Например, в разных языках читают и пишут по-разному: слева направо, справа налево или даже сверху вниз.
- **Особенности формата** — основная причина противоречий при работе с приложениями, изначально написанными для работы с другим языком. Необходимо внимательно относиться к форматам адресов и телефонных номеров, типам валют, датам и времени, размерам бумаги и единицам измерения. Автоматически решить большинство из этих проблем разработчикам поможет использование API-интерфейса поддержки национальных языков (National Language Support, NLS) в Microsoft Windows или пространства имен *System.Globalization*.
- **Проблемы со строками.** Строки — это текст, отображаемый в различных элементах пользовательского интерфейса приложения, таких, как информационные окна, диалоговые окна, строки заголовков, строки состояния и меню. В

частности, в информационных окнах, как правило, показывают результат конкатенации строк стандартного текста и контекстной информации. При смене языка полученные в строки могут оказаться грамматически некорректными.

- **Проблемы пользовательского интерфейса.** Необходимо уделять особое внимание дизайну ряда компонентов пользовательского интерфейса:
 - **сообщения** — длина сообщения в разных языках различается;
 - **меню и диалоговые окна** — при локализации их размер иногда приходится менять (например, увеличивать);
 - **значки и растровые изображения** должны содержать символы, имеющие одинаковое значение во всех странах;
 - **клавиши доступа и быстрые клавиши.** Наборы символов и клавиш на клавиатурах для разных языков обычно отличаются. Необходимо убедиться, что клавиши доступа и быстрые клавиши поддерживаются на клавиатуре целевого языка;
 - **элементы управления пользовательского интерфейса** не должны быть скрыты или использоваться в качестве части строки.

Рекомендации по поддержке различных языков и локализации

Далее приведены рекомендации, которые позволяют устранить большинство проблем и затруднений, возникающих при разработке приложений, призванных поддерживать многие языки.

- **Технические особенности.**
 - Используйте Unicode в качестве стандарта кодирования текстовых символов. Все приложения обрабатывают данные, текстовые или числовые, причем в разных языках и диалектах могут использовать различные способы кодирования данных. Unicode — это международный стандарт кодирования символов, предусматривающий 16-битное представление символов, поддерживающий более 45 000 символов и обеспечивающий уникальное представление всех символов всех существующих систем письма.
 - Реализуйте многоязыковый пользовательский интерфейс. Рекомендуется проектировать пользовательский интерфейс, который позволяет быстро переключаться между языками. Это очень удобно для пользователей, говорящих на разных языках.
 - Отслеживайте сообщения Windows, отражающие изменения языка ввода данных, и используйте эту информацию для смены режима проверки правописания, выбора шрифтов и т.п.
 - Определяйте используемый приложением язык и применяйте эту информацию при обработке дат, валют и представления чисел и представляйте их в соответствии с региональными предпочтениями. В .NET Framework параметр *culture* относится к языку пользователя (а также к его местоположению). Его определение позволяет моментально изменять представление информации — строк, формата даты и чисел — в соответствии с языком пользователя и региональными особенностями.
- **Культурные и политические особенности** — это, к примеру, отображение на карте спорных территорий, что может побудить власти запретить распространение ваших приложений в определенных регионах. Это не значит, что приложение не сможет работать в стране — просто это негативно скажется на

имидже приложения, и клиенты предпочтут продукты ваших конкурентов. Чтобы этого избежать:

- избегайте жаргона, разговорных выражений и двусмысленных формулировок;
- не используйте изображения, которые представителям других национальностей могут показаться националистическими или оскорбительными;
- не используйте карты, на которых изображены спорные региональные или национальные границы.
- **Особенности пользовательского интерфейса.**
 - Храните все элементы пользовательского интерфейса в ресурсных файлах, файлах сообщений или частных базах данных — то есть отделите их от исходного кода программы.
 - Размещайте в ресурсных файлах только те строки, которые требуют локализации, Нелокализуемые строки опишите в виде констант в исходном коде.
 - Используйте одни идентификаторы ресурсов на всем протяжении проекта. Изменение идентификаторов затруднит обновление локализованных ресурсов от версии к версии.
 - Если одна строка используется в разных контекстах, создайте несколько копий этой строки, поскольку при переводе могут появиться различия в контекстах.
 - Выделяйте текстовые буферы динамически, поскольку размер текста при переводе изменяется.
 - Не забывайте, что при локализации размер диалоговых окон может меняться.
 - Избегайте размещения текста на растровых изображениях и значках.
 - Не создавайте текстовые сообщения динамически в ходе выполнения программы.
 - Избегайте текста, в котором используется множество вставных параметров в строках форматирования.
 - При локализации на языки Ближнего Востока, например арабский или иврит, применяйте специальные API-интерфейсы для вывода текста справа налево.
 - Тестируйте локализованные приложения для всех языков, поддерживаемых операционной системой.

Занятие 2. Планирование функций администрирования

Функции администрирования — важная составная часть решения. На этом занятии вы узнаете, как проектировать и планировать такие функции администрирования, как мониторинг, перенос (*migration*) данных и спецификации лицензирования решения.

Изучив материал этого занятия вы сможете:

- ✓ спланировать мониторинг решения;
- ✓ спланировать перенос данных;
- ✓ создать спецификацию лицензирования.

Продолжительность занятия — около 10 минут.

Планирование мониторинга

Мониторинг приложений используется для контроля корректности работы и производительности приложения. Автоматизированный мониторинг позволяет обнаруживать причины сбоя и прогнозировать неполадки, а также сократить время восстановления после сбоя.

За мониторинг приложений, как правило, отвечают администраторы и другие сотрудники, занимающиеся обслуживанием системы. Очень важно задокументировать правила и процедуры мониторинга приложений. Согласование этих процедур с командой разработки позволяет обеим командам плодотворно сотрудничать в деле протоколирования и мониторинга информации, помогающей проводить исследование и диагностику неполадок. Это продолжающийся и постоянно корректируемый процесс, а администраторам и команде разработки следует постоянно улучшать процедуры мониторинга.

Протоколирование ошибок тесно связано с мониторингом, а обеспечивают его разработчики. Стратегию управления ошибками необходимо продумать на ранних стадиях этапа проектирования. Команде необходимо поддерживать связь с администраторами и **сообщать** ему, какие журналы ошибок ведет приложение. На администраторов возлагается задача информировать разработчиков о существующих механизмах мониторинга ошибок. Обе команды должны согласовать механизмы протоколирования с тем, чтобы обеспечить эффективные механизмы мониторинга разрабатываемого решения.

План мониторинга

В плане мониторинга определяется процесс мониторинга: что именно и каким образом предполагается контролировать и как представлять и использовать результаты мониторинга. Многие процедуры мониторинга в корпоративных приложениях выполняются автоматически.

План содержит детальную информацию о процессе мониторинга, которая затем представляется в функциональной спецификации. **После** добавления в функциональную **спецификацию** процесс мониторинга (ручной или автоматизированный) становится неотъемлемой частью дизайна решения. Мониторинг гарантирует, что администраторы узнают о произошедшем сбое и смогут при-

нять меры по восстановлению работоспособности отказавшего сервиса. Кроме того, некоторые организации ведут наблюдение за работой серверов, чтобы выявить особенности и схемы их использования, — это позволяет выявлять условия возникновения сбоев и принимать меры по предотвращению их в будущем.

Разделы плана мониторинга

- **Мониторинг пороговой загрузки ресурсов.** Здесь описывается состав контролируемых ресурсов, а также пороговые значения для каждого ресурса, например регистрация всех случаев, когда загрузка процессора превысила 80%.
- **Мониторинг рабочих характеристик.** В этой части перечисляются подлежащие контролю параметры приложения и отдельные его компоненты, в том числе протоколируемые и контролируемые события, частота их возникновения, время начала и окончания и результат (удачный или неудачный). Например, протоколируют попытки входа пользователей в систему и количество нескольких неудачных попыток одного пользователя подряд.
- **Анализ тенденций** в собранных в процессе мониторинга параметрах поведения системы. Результаты используются для прогнозирования поведения приложения в различных условиях, например для определения количества пользователей в различное время суток и поведения приложения при соответствующих нагрузках.
- **Обнаружение сбоев.** В этой части описывается, как разработчики, обслуживающий персонал и команда сопровождения будут использовать функциональную спецификацию и пользовательские требования для обнаружения фактов сбоя. Функциональная спецификация пригодится для определения критерия успешности решения. Эта же информация включается в пользовательские требования. Например, покупатели должны иметь возможность просматривать товары и выбирать их из списка, но если список пуст, это считается ошибкой.
- **Обнаружение ошибок.** Здесь описываются процессы, методы и инструменты, которые применяются для обнаружения и диагностики ошибок в продукте. Эти ошибки, как правило, определяются и исправляются разработчиками, администраторами и командой сопровождения без участия пользователей.
- **Журналы событий** предназначены для регистрации и последующего просмотра и анализа важных событий системы и приложения. Например, в приложении, поддерживающем взаимодействие с SQL Server, можно использовать журналы этого сервера.
- **Оповещения.** Здесь описывается, как администраторы оповещаются о обоях решения в процессе мониторинга и обработки ошибок. Существует много способов оповещения, но наиболее популярны пейджер и электронная почта.
- **Инструменты,** используемые для обнаружения, диагностики и устранения неполадок, а также для повышения производительности приложения. Команда может использовать такие инструменты, как Microsoft Systems Management Server и System Monitor.

Планирование переноса данных

Если в новом решении необходимо использовать данные из унаследованных или уже существующих источников, возникает вопрос о переносе данных. В отсутствие проверенных способов осуществления этой операции новое решение может работать со сбоями из-за особенностей существующих компонентов, не учтен-

ных при планировании. Если данные из унаследованных систем нельзя перенести в новое решение, его не удастся развернуть, и деньги, потраченные на него, не оправдаются.

(В плане переноса данных описывается процесс перехода с существующих систем или приложений к новому продукту.) Порядок такого переноса обычно важнее для инфраструктуры развертывания, чем для проекта разработки ПО, хотя он и учитывается в последнем.

План переноса состоит из ряда разделов.

- **Стратегии переноса** указывают основные принципы переноса данных. Стратегии не обязательно должны быть взаимоисключающими, но они описывают различные стороны общего **процесса перехода**. Стратегия может базироваться на системе версий (в зависимости от зрелости **бизнес-процессов**, разработки или технологии) или компонентов. Здесь также необходимо учесть особенности перехода от предыдущих систем к новому продукту. Если перенос касается бизнес-объектов и данных, в стратегии предусматривают несколько разделов, посвященных стратегиям переноса.
- **Инструменты**, которые предполагается использовать для переноса: инструменты преобразования, установки, тестирования и обучения.
- **Руководство по переносу** описывает принципы, которым необходимо следовать, например, как предполагается выполнять проверку **переносимых данных** или как переносить данные о заказах.
- **Процесс переноса описывает**, как предполагается выполняться перенос. Кроме стадий переноса здесь указывают подготовительные операции.
- **Среда тестирования**, которая в точности повторяет реальную среду работы приложения и обладает всеми атрибутами реальной среды.
- **План отката** определяет, как заказчик может вернуться к предыдущей конфигурации при неполадках в процессе переноса.

Создание спецификаций по лицензированию

Определять приобретаемое для проекта аппаратное и программное обеспечение рекомендуется как можно раньше. Если спецификации для закупок разработаны на ранних этапах, остается достаточно времени для их **утверждения**, а поставщики оборудования могут заранее планировать поставки в соответствии с календарным планом проекта. Важной частью спецификаций этого вида являются спецификации по лицензированию.

Отдельные спецификации по лицензированию создают для этапов разработки и развертывания. В процессе разработки команда использует выбранные технологии и программное обеспечение. При этом следует позаботиться о **приобретении** всех требуемых лицензий на используемые продукты.

Необходимо выяснить общее количество лицензий для каждого программного продукта, который предполагается использовать. Например, если приложение работает только под управлением **Windows XP**, не забудьте предусмотреть соответствующее число лицензий на **Windows XP**.

Занятие 3. Планирование будущих этапов

Модель процессов MSF состоит из пяти этапов: создания общей картины, планирования, разработки, стабилизации и развертывания. На этапе планирования намечают, кто, что и как будет разрабатывать, а также создают план действий, которые предполагается выполнить на последующих этапах.

На этом занятии вы узнаете, как планировать задачи для следующих этапов проекта.

Изучив материал этого занятия вы сможете:

- ✓ создавать планы для этапа разработки;
- ✓ создавать планы для этапа стабилизации;
- ✓ создавать планы для этапа развертывания.

Продолжительность занятия — около 5 минут.

Планирование этапа разработки

До начала разработки решения важно убедиться в готовности инфраструктуры разработки и среды тестирования. Среда тестирования должна как можно точнее воспроизводить реальную среду, в которой будет функционировать приложение, но здесь важно соблюсти баланс между уровнем имитации окружения и связанными с этим затратами. Важно разделять реальную производственную среду и среду разработки и тестирования, чтобы исключить негативное влияние на работу действующих систем.

План разработки

На этапе планирования команда проекта создает генеральный план и генеральный календарный график проекта. Помимо этого часто создают план и календарный график разработки. В первом описывают процесс разработки решения и задачи по созданию и сборке компонентов продукта. Этот план дополняет функциональную спецификацию, содержащую детальную техническую информацию о разрабатываемом решении. В плане также формулируются четкие правила и описываются процессы для команд, которые будут создавать продукт.

Документы для разработки подтверждают, что команда обсудила структуру и направление разработки, у ее членов нет разногласий на этот счет, и это позволяет разработчикам сконцентрироваться на создании решения. Наличие принципов и стандартов разработки способствует плодотворному взаимодействию различных команд, поскольку обеспечивает единство методов и процессов. Единые принципы и стандарты также облегчают повторное использование плодов труда различных групп и сводят к минимуму зависимость от отдельных сотрудников или групп.

Роль «разработчик» в модели команд MSF отвечает за создание плана и календарного графика разработки, то есть определяет порядок реализации основных составных частей процесса разработки.

Разделы плана разработки

- **Цели разработки** — основные направления и задачи разработки.
- **Общая стратегия поставки решения** — общий подход к поставке решения, например поэтапная поставка, горизонтальное или вертикальное внедрение или

реализация большого набора функций с последующей оптимизацией производительности.

- **Компромиссы** — принципы принятия компромиссных решений в процессе дизайна и реализации. Например, можно отказаться от некоторых функций ради того, чтобы уложиться в график или обеспечить повышенную производительность.
- **Ключевые цели дизайна** — основные задачи дизайна и их важность, например способность к взаимодействию и безопасность.
- **Среда разработки и сборки приложения** — среда, в которой предполагается выполнять разработку и сборку, а также управление ими. Здесь содержится информация о таких элементах, как инструменты управления исходным кодом, требования к средствам создания дизайна, операционные системы и прочее установленное ПО.
- **Предписания и стандарты** — перечисление и ссылки на все используемые в проекте стандарты и предписания.
- **Управление версиями и исходным кодом** — описание того, как предполагается управлять версиями и исходным кодом. Здесь перечислены конкретные инструменты и указано, как разработчики должны их использовать.
- **Процесс сборки** — описание **инкрементного** и итеративного подхода к созданию кода и сборке аппаратных и программных компонентов. Здесь также описывается порядок и частота сборки.
- **Компоненты** — высокоуровневое описание набора компонентов продукта и , порядка их разработки.
- **Средства управления конфигурацией и разработкой** — все инструменты разработки, которые будут использоваться командой в проекте. К ним относятся инструменты, применяемые на всех стадиях проекта, — при разработке, тестировании, документировании, сопровождении и развертывании.
- **Шаблоны проектирования**, которые понадобятся и для разработки исходных текстов. Команда вправе брать шаблоны из внешних и внутренних источников или создавать новые.
- **Обучение команды разработчиков** — перечисление тренингов и занятий, необходимых членам команды, чтобы успешно справиться с разработкой решения.
- **Поддержка команды разработчиков** — различные виды помощи, которая может потребоваться команде программистов, в том числе ее источники, размер и приблизительный график оказания помощи.

Планирование этапа стабилизации

На этапе стабилизации команда тестировщиков проверяет решение, реализация функций которого к этому моменту завершена. На этом этапе в основном тестируется работа продукта в реальных условиях. Команда уделяет основное внимание устранению и определению важности ошибок и подготовке решения к окончательному выпуску.

На этапе планирования команда обычно создает следующие планы, которые в дальнейшем используются на этапе стабилизации:

- план тестирования;
- план пилотной эксплуатации.

План тестирования

Здесь описывается стратегия и методы планирования, организации и управления тестированием проекта. В плане тестирования определены цели тестирования, ме-

тодологии и инструменты, ожидаемые результаты, обязанности и требования к ресурсам. Это основной документ команды тестирования. Основная задача плана тестирования — обеспечить тщательно и качественно организованный процесс тестирования, а также определить используемые командой разработчиков критерии достаточной стабильности решения. Когда команда и заинтересованные в проекте лица знают, в каком состоянии находится решение в тот или иной момент времени, это укрепляет их уверенность на этапе разработки и стабилизации приложения.

Роль «тестировщик» модели команд MSF отвечает за разработку требований по качеству и включение их в план тестирования.

План тестирования предусматривает несколько этапов:

- тестирование компонентов кода;
- тестирование базы данных;
- тестирование инфраструктуры;
- тестирование на предмет безопасности;
- тестирование интеграции;
- тестирование принятия продукта пользователями и удобства работы с ним;
- нагрузочное тестирование, тестирование запаса по ресурсам и производительности;
- регрессионное тестирование.

Разделы плана тестирования

- **Методы и допущения тестирования** — высокоуровневое описание способов, операций и методов тестирования решения. Если методы тестирования отдельных компонентов решения отличаются, необходимо перечислить компоненты, тестируемые посредством того или иного метода.
- **Основные обязанности** — описание команд и отдельных сотрудников, отвечающих за управление процессом тестирования, а также непосредственных исполнителей.
- **Тестируемые функции** — высокоуровневое описание функций, которые необходимо проверить.
- **Ожидаемые результаты тестов** — перечень результатов, которые следует получить в результате тестирования. Здесь учитываются ожидания как команды разработчиков, так и тестировщиков. В этом разделе также определяется, должны ли результаты точно соответствовать ожидаемым, или оговаривается их допустимый диапазон.
- **Результаты** — перечень материалов, которые необходимо подготовить или разработать для успешного выполнения тестов, а также материалы, подготовленные на основании результатов тестирования.
- **Процедуры тестирования и пошаговый проход** — описание операций, которые должна выполнить команда тестировщиков в рамках проверки качества решения.
- **Контроль и отчетность** — перечень информации, которой обмениваются члены команды в процессе тестирования. Здесь описывается особая информация о состоянии процесса тестирования, которая должна накапливаться и распространяться, в том числе сведения о результатах отдельных тестов и вероятность завершения цикла тестирования в установленные сроки.
- **Инструменты и способы контроля ошибок** — полное описание стратегии и методологии отслеживания ошибок. Здесь также оговаривается, что следует считать ошибкой в коде, особенностями продукта или документацией.

- **Календарные планы** — основные циклы тестирования, задачи, контрольные точки и результаты. Здесь также перечисляются лица, ответственные за каждый из циклов тестирования и их задачи. Кроме того, в этом разделе определяются приблизительные даты начала и завершения каждого цикла тестирования и задачи, относящиеся к каждому циклу.

План тестирования для компании Adventure Woks Cycles приведен в документе *AWC Test Plan.docx* в каталоге `\Solution Documents\Chapter10` прилагаемом к книге компакт-диске.

Примечание Подробно о тестировании рассказывается в главе 11.

План пилотной эксплуатации

План опытной эксплуатации описывает размещение и тестирование версии-кандидата в промежуточной среде. Цель пилотной эксплуатации — воссоздать оборудование, ПО и компоненты, с которыми решение будет работать в реальных условиях.

План также определяет, как предполагается решать выявленные во время опытной эксплуатации неполадки и проблемы, а также как оценивать результаты опытной эксплуатации и на основании этой оценки принимать решение, готова ли система к промышленной эксплуатации.

Нередко проектной команде приходится повторять опытную эксплуатацию, чтобы убедиться в адекватности использованных в решении методов или чтобы поэкспериментировать с разными вариантами решения отдельных задач и получить отзывы пользователей и узнать их отношение к разрабатываемому решению. В пилотных версиях продукта реализуется только та часть требований или функциональной спецификации, которая необходима для проверки решения.

Примечание Для некоторых проектов пилотная эксплуатация не нужна.

План опытной эксплуатации позволяет проверить соответствие решения бизнес-требованиям и техническим спецификациям перед развертыванием в промышленной среде. Опытная эксплуатация позволяет удостовериться, что все участвующие команды знают свои роли и обязанности, а также требования к ресурсам, необходимым в процессе разработки, тестирования и развертывания пилотной версии.

Примечание Подробнее о планировании пилотного запуска рассказывается в главе 11.

Планирование этапа развертывания

На этом этапе команда развертывает решение, а также все необходимые технологии и компоненты и стабилизирует развернутую систему, проект переходит в стадию сопровождения и поддержки, и заказчик окончательно утверждает его.

План развертывания

В числе прочих документов на этапе планирования команда создает план развертывания, в котором описываются условия, обеспечивающее максимально

гладкое развертывание и переход к промышленной эксплуатации. Здесь описаны процессы подготовки, установки, обучения пользователей, стабилизации и вывода решения на рабочий режим. В этих процедурах предусмотрена реализация сценариев установки, мониторинг стабильности продукта и его проверки «на прочность». Рассматриваемый план служит руководством по переводу решения в режим промышленной эксплуатации, а также содержит детальные рекомендации, как успешно выполнить этап развертывания решения. С момента развертывания решение начинает приносить реальную пользу. Детализированный и тщательно выверенный план развертывания приближает момент получения отдачи от решения, причем как заказчиком, так и проектной командой.

Роль «менеджер по выпуску» модели команд MSF несет ответственность за проектирование и реализацию плана развертывания решения, а также за определение инфраструктуры решения и обеспечение бесперебойной работы после развертывания продукта.

Разделы плана развертывания

- **Область развертывания** — описание архитектуры решения и границы развертывания.
- **Узлы** — определение масштаба развертывания в терминах местоположений, количества рабочих станций, стран и регионов и прочих параметров, характеризующих размер и охват решения.
- **Компоненты** — перечисление и описание подлежащих развертыванию компонентов и всех критических связей между ними.
- **Архитектура** — описание архитектуры решения и влияния, которое она оказывает на развертывание.
- **Календарный план развертывания** определяет критические даты и примерный график этапа развертывания.
- **Установка** — описание процесса развертывания в целом.
- **Ресурсы, необходимые для развертывания**, — людские ресурсы, необходимые для успешного развертывания решения.
- **Служба поддержки** — описание порядка поддержки пользователей и приложений командой службы поддержки, в том числе предоставление ответов на прямые обращения пользователей и советы по устранению неполадок, а также всесторонняя помощь при освоении новых или трудных для понимания функций.
- **Рабочие места** — описание всех изменений в настройке приложений на рабочих станциях, которые могут потребоваться при развертывании.
- **Серверы** — перечень всех изменений в настройке серверов, которые могут потребоваться в процессе развертывания.
- **Связь** — перечень всех изменений в поддержке телекоммуникаций, которые могут потребоваться в процессе развертывания.
- **Координация обучения** — описание порядка координации обучения конечных пользователей и персонала службы поддержки с календарным планом развертывания.
- **Процесс установки на площадке** — описание четырех этапов развертывания решения на конкретной площадке (в отделении, отдельном здании или филиале компании): подготовки, установки, обучения и стабилизации.

Занятие 4. Создание технической спецификации

Для начала работы над созданием продукта команде разработчиков необходима техническая спецификация. На этом занятии рассказывается о технической спецификации и о составляющих частях этого документа.

Изучив материал этого занятия вы сможете:

- ✓ рассказать о назначении технической спецификации;
- ✓ перечислить составляющие части технической спецификации.

Продолжительность занятия — около 5 минут.

Что такое техническая спецификация

Техническая спецификация — это набор справочных документов, обычно содержащих артефакты физического дизайна, такие, как спецификации классов, модели компонентов, метрики и топология сети и компонентов. На этапе разработки разработчики используют техническую спецификацию для определения области действия создаваемого продукта и ожидаемых от него результатов. Техническая спецификация содержит описания разделов реестра, определения интерфейсов, DLL-библиотек и сборок, строгих имен, ключей и других элементов, имеющих отношение к разворачиванию. Если создание элементов решения завершается на этапе разработки, в техническую спецификацию вносятся соответствующие коррективы и указывается, что она теперь описывает готовый продукт.

Разделы технической спецификации

- **Общее описание архитектуры** — схема архитектуры, которую предполагается реализовать в решении.
- **Объектная модель** — описание объектной модели решения. Этот раздел описывает все объекты продукта и их функциональные возможности.
- **Интерфейсы** — исходный код и детальная информация о методах каждого интерфейса продукта.
- **Ход исполнения кода** — описание порядка работы каждого из методов решения.
- **Коды ошибок** — используются для обработки ошибок в приложении.
- **Регистрация ошибок** — порядок обработки и регистрации ошибок в решении.
- **Конфигурация** — описание порядка регистрации решения в реестре, в том числе перечень параметров реестра и их значения.
- **Вспомогательная документация** — перечень и местонахождение описывающих решение документов, например функциональной спецификации.
- **Неполадки** — описание известных неполадок приложения. Для каждой проблемы, как правило, указывается примерная дата устранения.

Техническую спецификацию для компании Adventure Works Cycles вы найдете в документе *A WC Technical Specification.docx* в каталоге `\SolutionDocuments\Chapter10` на прилагаемом компакт-диске. Это предварительная версия технической спецификации — так она выглядит до начала этапа разработки. Этот документ претерпевает изменения в процессе разработки.

Практикум. Анализ плана тестирования и технической спецификации



При выполнении упражнений используйте знания, полученные на занятиях по изучению плана тестирования и технической спецификации.

Упражнение 1. Анализ плана тестирования

Откройте документ *AWC Test Plan.doc* из папки `\SolutionDocuments\Chapter10` на прилагаемом к книге компакт-диске.

1. Изучите план тестирования и перечислите три способа применения ВИС и СИС в процессе тестирования решения для компании Adventure Works Cycles.
2. Каким образом следует признавать и документировать ошибки, чтобы откорректировать и исправить приложение?
3. Какие учетные записи пользователей следует сконфигурировать на тестовых серверах?
4. Почему описанную далее проблему следует указать в списке опасностей для тестирования?
Необходимость участия менеджеров по продажам в тестировании. Команду тестирования должен контролировать по крайней мере один торговый представитель. Решение: заранее договориться в вице-президентом по продажам о том, что в тестировании будут участвовать два торговых представителя.

Упражнение 2. Анализ технической спецификации

Откройте документ *AWC Technical Specification.doc* из папки `\SolutionDocuments\Chapter10`. Ответьте на следующие вопросы:

1. Каковы параметры метода *addOrderDetail*?
2. Будет ли в приложении два типа пользовательского интерфейса?
3. Почему раздел «Интерфейсы» все еще не определен?

Резюме

- Масштабируемость определяется как способность расширять возможности системы по предоставлению сервисов за счет расширения системных ресурсов.
- Существует два типа масштабирования: вертикальное и горизонтальное.
- Основные принципы масштабирования:
 - процессы не должны ожидать отклика дольше, чем необходимо;
 - процессы не должны конкурировать за ресурсы;
 - следует добиваться коммутативности процессов;
 - следует создавать взаимозаменяемые компоненты;
 - необходимо разделять ресурсы и сервисы.
- Доступность — это мера отказоустойчивости приложения, то есть соотношение времени, когда приложение обслуживает запросы, к запланированному времени его работы.
- Чтобы определить необходимый уровень доступности, нужно ответить на следующие вопросы.
 - Кто пользователи приложения и чего они ожидают от приложения?
 - Какое время простоя допустимо?
 - Зависят ли от сервиса какие-либо внутрикорпоративные процессы?
 - Каковы сроки разработки приложения и бюджет, выделенный для этого?
- Основные методы обеспечения доступности:
 - сокращение времени простоя;
 - сокращение внеплановых простоев за счет использования кластеризации;
 - балансировка сетевой нагрузки;
 - использование RAID-массивов для хранения данных;
 - изоляция критически важных приложений;
 - использование очередей.
- Для обеспечения надежности необходимо определить предполагаемый уровень использования приложения, создать профиль надежности и разработать решение, удовлетворяющее этому профилю.
- Следующие мероприятия позволяют создать надежное приложение:
 - отражение в спецификации требований к надежности;
 - использование качественной архитектурной инфраструктуры;
 - включение в приложение информации, необходимой для управления;
 - реализация избыточности и дублирования;
 - применение инструментов контроля качества;
 - проверка надежности приложения;
 - реализация обработки ошибок;
 - создание архитектуры приложения, которая предусматривает в случае сбоя отключение лишь некоторых функций, а не крах всего приложения.
- Производительность приложения измеряется определенными метриками, например количеством операций в единицу времени или степенью использования ресурсов.
- Чтобы определить цели, необходимо ответить на ряд вопросов.
 - Какова цель с точки зрения бизнеса?
 - Какие функции являются критически важными?

- Какие возможности требуются разным группам пользователей?
- Для интеграции гетерогенных приложений необходимо учитывать следующие типы взаимодействия:
 - сетевое взаимодействие — основополагающий тип взаимодействия систем;
 - взаимодействие с различными источниками данных, которое обеспечивает пользователям и приложениям доступ к информации и предоставляет возможность выполнять запросы данных, хранящихся как в структурированных, так и в неструктурированных системах хранения;
 - взаимодействие приложений — инфраструктура, необходимая для взаимодействия новых многоуровневых и существующих приложений;
 - взаимодействия, связанные с управлением и ориентированные на снижение трудоемкости администрирования нескольких систем, в том числе управление пользовательскими учетными записями.
- Обеспечение поддержки других языков — это процесс разработки приложений, которые могут использоваться в любой стране мира.
- Локализация — это процесс адаптации приложения к конкретному языку и стандартам с использованием характерных для конкретной страны ресурсов.
- Мониторинг приложений необходим для контроля корректности работы и производительности приложения,
- В плане мониторинга определяется процесс мониторинга: что именно и каким образом предполагается контролировать и как будут представляться и использоваться результаты мониторинга.
- В плане переноса данных описывается процесс перехода с существующих систем или приложений к новому продукту.
- Необходимо предусмотреть спецификации по лицензированию для этапов разработки и развертывания.
- План разработки описывает используемый в проекте процесс разработки решения.
- План тестирования описывает стратегию и методы, необходимые для планирования, организации и управления проверкой проекта, определяет цели тестирования, методологии и инструменты, ожидаемые результаты, обязанности и требования к ресурсам.
- План пилотной эксплуатации описывает части решения, которые предполагается проверить методом опытной эксплуатации, и предоставляет информацию, необходимую для успешного проведения пилотных испытаний.
- План развертывания описывает условия, обеспечивающее максимально гладкое развертывание и переход к промышленной эксплуатации.
- Техническая спецификация — это набор справочных документов, обычно содержащих артефакты физического дизайна, такие, как спецификации классов, модели компонентов, метрики и топологию сети и компонентов.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Каким образом выполняется вертикальное масштабирование приложения?
2. Что необходимо учитывать при проектировании масштабируемого решения?
3. Что необходимо учитывать при проектировании решения, отличающегося высокой доступностью?
4. Как кластеризация повышает доступность приложения?
5. Каким образом сократить время планового простоя приложения?
6. Почему надежность — одна из важнейших составляющих проекта приложения?
7. Как обеспечить надежность в проектируемом приложении?
8. Как определить требования к производительности?
9. Почему следует проектировать приложения, эффективно поддерживающие взаимодействие с другими системами?
10. Каким образом обеспечить в приложении возможность поддержки других языков?
11. Каково назначение плана мониторинга?
12. Каково назначение раздела, посвященного стратегиям плана переноса данных?
13. Почему необходимо создавать спецификации по лицензированию для двух этапов — разработки и развертывания?
14. Каково назначение плана разработки?
15. Каково назначение плана тестирования?
16. Кто отвечает за создание плана развертывания?
17. Что такое техническая спецификация?

Стабилизация и развертывание решения

Занятие 1. Этап стабилизации в MSF	330
Занятие 2. Тестирование и пилотная эксплуатация	334
Занятие 3. Этап развертывания в MSF	346
Занятие 4. Развертывание в промышленной среде	348
Практикум. Определение важности ошибок	355
Резюме	357
Закрепление материала	358

В этой главе

По завершении этапа разработки проект входит в стадию стабилизации и развертывания. Цель этапа стабилизации модели процессов MSF — повысить качество решения, чтобы сделать достаточно надежным для начала промышленной эксплуатации. На этапе развертывания решения развертывается в промышленной среде.

Прежде всего

Для изучения материалов этой главы необходимо:

- общее представление о технологиях Microsoft;
- понимание модели процессов MSF и их этапов.

Занятие 1. Этап стабилизации в MSF

На этапе стабилизации проводится тестирование реализованных функций решения. В рамках тестирования, которое начинается на ранних этапах процесса MSF, выполняются: определение критерия успешности и методов тестирования — на этапе создания общей картины и создание плана тестирования — на этапе планирования. Но лишь на этапе стабилизации команда тестирования завершает работу, в результате которой сборка (build) с полностью реализованными функциями достигает заданного уровня качества и решение становится пригодным для развертывания в промышленных условиях.

При тестировании на этом этапе основное внимание уделяется пригодности решения для работы в условиях живого производства. Основное внимание команда уделяет исправлению ошибок и определению их важности, а также подготовке продукта к выпуску.

На этапе стабилизации решают две основные задачи.

- **Тестирование решения.** Команда выполняет планы тестирования, созданные на этапе планирования и расширенные и опробованы на этапе разработки.
- **Пилотная эксплуатация** — развертывание решения в тестовой среде и тестирование с привлечением *будущих* пользователей и реализацией реальных СИС. Эта задача выполняется до начала этапа развертывания.

Изучив материал этого занятия, вы сможете:

- ✓ описать результаты этапа стабилизации MSF;
- ✓ описать промежуточные контрольные точки этапа стабилизации MSF;
- ✓ описать роли и обязанности членов команды на этапе стабилизации MSF.

Продолжительность занятия — около 15 минут.

Результаты этапа стабилизации

Основная цель этапа стабилизации — повысить качество продукта и стабилизировать его на уровне, пригодном для использования в промышленной среде. На этом этапе команда выполняет тестирование решения, все функции которого к этому моменту уже реализованы, а также проверяет точность и корректность сопроводительной документации, процедур обучения и прочих составных частей проекта, не имеющих непосредственного отношения к коду.

К результатам этапа стабилизации относятся:

- анализ пилотной эксплуатации;
- готовые к выпуску версии:
 - исходного кода и исполняемых файлов;
 - сценариев и документации по установке;
- справочной системы и материалов для обучения;
- информативных документов о версии;
- отчеты по тестированию и обнаруженным ошибкам;
- проектные документы.

Промежуточные контрольные точки этапа стабилизации

Помимо результатов на этапе тестирования предусмотрены промежуточные контрольные точки:

- сходимость числа ошибок;
- версия без обнаруженных ошибок;
- кандидаты на выпуск;
- «золотая» версия.

Сходимость числа ошибок

Сходимость числа ошибок означает достижение существенного прогресса в «работе над ошибками» — число устраненных ошибок превышает количество вновь обнаруживаемых (рис. 11-1).

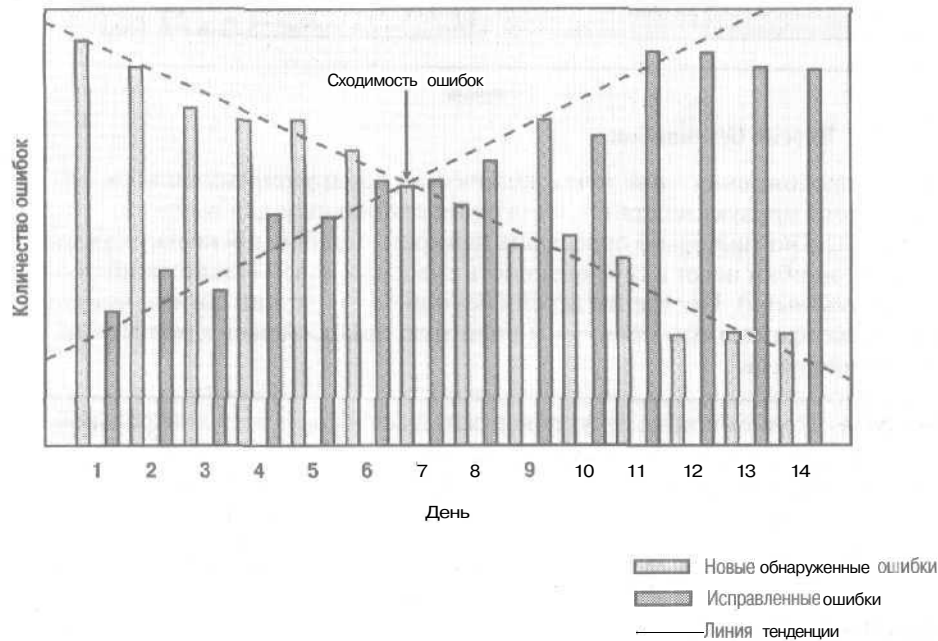


Рис. 11-1. Сходимость ошибок

Далее количество обнаруживаемых ошибок может как повышаться, так и падать — даже после значительного общего снижения, поэтому сходимость числа ошибок следует считать тенденцией, а не фиксированной точкой во времени. После начала сходимости число ошибок должно далее снижаться — вплоть до выпуска версии без обнаруженных ошибок.

Версия без обнаруженных ошибок

Это точка, в которой разработчики устранили все ошибки, найденные тестировщиками, таким образом сведя к нулю количество известных ошибок на этот момент времени (рис. 11-2).

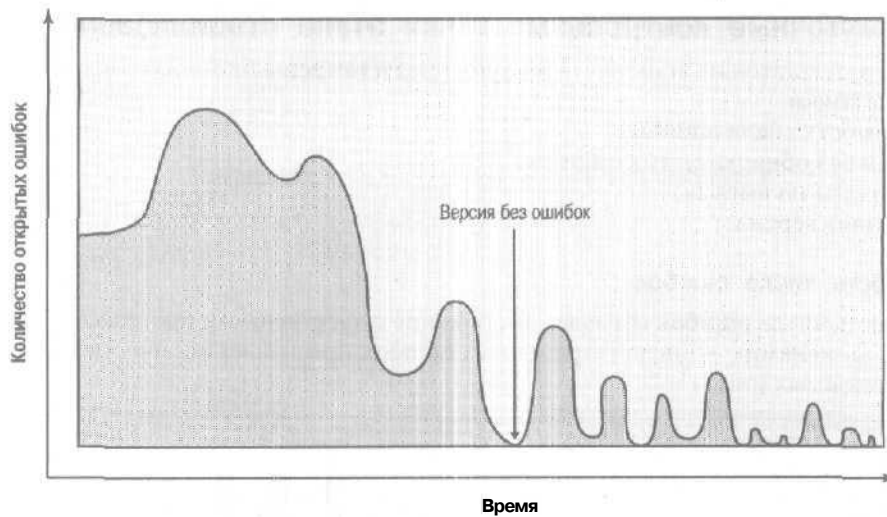


Рис. 11-2. Версия без ошибок

После прохождения этой точки количество обнаруженных ошибок должно затухать, пока продукт не станет достаточно стабильным для выпуска.

Крайне важно тщательно определять *важность* ошибок, поскольку каждое исправление ошибки несет в себе опасность внесения новой (такие ошибки называются *регрессивными*). Получение версии без ошибок — четкий сигнал, что команда вышла на «финишную прямую» и вплотную приблизилась к получению стабильного продукта.

Примечание Новые ошибки будут обнаруживаться и после этой контрольной точки, но она сигнализирует, что команда впервые может честно рапортовать о полном отсутствии неисправленных ошибок — пусть даже только в одной *версии*. Это заставляет команду сконцентрировать все *усилия*, чтобы удержаться на достигнутом уровне.

Кандидаты на выпуск

После достижения первой точки отсутствия ошибок готовится серия версий продукта — *кандидатов на выпуск* для передачи в группу пилотной эксплуатации. Каждый из этих выпусков считается промежуточной контрольной точкой. Кандидаты на выпуск передают заранее выбранной группе пользователей, которая приступает к их тестированию. Пользователи делятся *впечатлениями* с проектной командой, а та в свою очередь продолжает совершенствовать продукт и исправлять ошибки, обнаруженные в процессе пилотной эксплуатации. При каждой новой сборке кандидата на выпуск число ошибок, требующих *документирования*, определения *важности* и *исправления*, должно снижаться.

«Золотая» версия

«Золотая» *версия* — это сборка продукта, предназначенная для промышленной эксплуатации. Это контрольная точка этапа стабилизации, которая характеризуется следующими параметрами: *отсутствием дефектов* и соответствием *кри-*

териям успешного завершения разработки. В качестве «золотого» выбирается один из выпусков-кандидатов, предназначенный для промышленной эксплуатации. При выборе выпуска применяется треугольник компромиссов и данные, полученные в процессе тестирования.

Задачи ролей команды на этапе стабилизации

В процессе стабилизации у каждой роль команды свои задачи и круг ответственности (таблица 11-1). На этом этапе доминируют роли тестировщика и менеджера выпуска.

Таблица 11-1. Роли и их обязанности на этапе стабилизации

Роль	Выполняемые задачи
Менеджер решения	Поддержка запланированных связей всех участников проекта, планирование запуска решения в промышленную эксплуатацию
Менеджер программы	Ведение проекта, определение важности ошибок
Разработчик	Устранение ошибок, оптимизация кода
Специалист по удобству использования	Стабилизация руководств, обучающих и вспомогательных материалов для пользователя
Тестировщик	Тестирование, отчеты об обнаруженных ошибках и состоянии продукта, конфигурационное тестирование
Менеджер по выпуску	Установка и поддержка пилотной версии, планирование развертывания, обучение сотрудников, которые будут поддерживать и сопровождать решение

Занятие 2. Тестирование и пилотная эксплуатация

Тестирование позволяет выявить и устранить неполадки до начала развертывания решения. Оно подтверждает, что компоненты решения соответствуют требованиям плана проекта к календарному плану и качеству продукта, а также позволяет убедиться в корректности и полноте функциональности.

Команда разработчиков конструирует дизайн, создает документацию и пишет код, который тестируется помодульно и в процессе ежедневных сборок. Она отвечает за реализацию проекта в соответствии с календарным планом, за устранение обнаруженных ошибок и за документирование результатов тестирования.

Команда тестировщиков отвечает за проектирование и документирование спецификации тестов и наборов тестовых данных, за создание сценариев автоматизированного тестирования, а также инициирует приемосдаточные испытания компонентов, переданных для формальных испытаний. Эта команда также несет ответственность за оценку общего качества продукта и степень завершенности его функций, а также подтверждает готовность набора функций и компонентов решения для реализации задач проекта.

Изучив материал этого занятия, вы сможете:

- ✓ определить рекомендации по тестированию;
- ✓ описать типы тестирования в MSF;
- ✓ объяснить термины, используемые при тестировании;
- ✓ описать процессы мониторинга неполадок и ошибок;
- ✓ определить некоторые задачи по тестированию;
- ✓ описать процесс опытной эксплуатации;
- ✓ рассказать о закрытии этапа стабилизации.

Продолжительность занятия — около 40 минут.

Рекомендации по тестированию

Существует несколько проверенных опытом рекомендаций по тестированию (best practices), которые команда должна использовать при тестировании решения.

Критерии успешности

В отсутствие четко определенных критериев практически невозможно принять решение об успешности проекта. Создание *критериев успешности* — это определение условий, при выполнении которых считается, что решение соответствует требованиям. Критерии успешности иногда называют *ключевыми показателями производительности*.

Примечание Понятие критериев успешности описывается в этом разделе, однако их следует определять на этапах создания общей картины решения и планирования.

Отсутствие дефектов

Идеология *бездефектности* подразумевает, что проектная команда обязуется обеспечить самый высокий уровень качества, который ей по силам. Каждый член команды несет личную ответственность за свой участок. Тезис об *отсутствии ошибок* не означает, что готовое решение будет идеальным, без единой ошибки, а определяет конкретный уровень качества.

Типы тестирования

В MSF определены два вида тестирования: базовое и пользовательских функций.

Базовое тестирование

Базовое тестирование (coverage testing) — это низкоуровневое техническое тестирование. Когда разработчик пишет очередной отрезок исходного текста или эксперт в предметной области выполняет автоматическую установку продукта, он, как правило, выполняет низкоуровневую проверку, чтобы убедиться в корректной работе новой функции в соответствии с функциональной спецификацией. В MSF такой вид тестирования называются *базовым*; в группах по продуктам в Microsoft его еще называют *предварительным* (prefix testing).

Ясно, что, если базовое тестирование выполняет автор, риск ошибок высок. Загруженность работой и спешка, связанная с необходимостью уложиться в срок, не позволит выявить все неполадки и ошибки. В такой ситуации хорошо себя зарекомендовало *внешнее базовое тестирование* — низкоуровневое тестирование, выполняемое сторонними тестировщиками. Часто оно автоматизируется для повышения точности и скорости.

Распространенный способ реализации такого тестирования — *взаимная проверка*, при которой код тестирует не сам разработчик, а его коллега, не *принимаящий* участия в написании соответствующего отрезка кода. Эта методика дает хорошие результаты, поскольку навыки, необходимые для разработки и базового тестирования полностью совпадают.

Тестирование пользовательских функций

Тестирование на пригодность к использованию ~ считается высокоуровневым, его часто выполняют будущие пользователи продукта или представители их подгрупп. Такое тестирование очень важно, поскольку *позволяет* гарантировать, что ошибки, связанные с работой пользователя, обнаружены и исправлены. Рекомендуется создавать сценарии автоматизации и вести контрольные списки тестирования, поскольку они обеспечивают повторяемость и четкий порядок действий тестировщика, что в конечном счете способствует повышению точности результатов.

Применяемые в тестировании термины

До начала разговора о тестировании решения необходимо ознакомиться с основными терминами.

Тестирование кода перед занесением в систему управления версиями

Этот вид тестирования позволяет проверить приемлемость созданного кода и выполняется разработчиками или *тестировщиками* перед переносом (check-in) его в систему управления версиями. Подобное тестирование можно рассматривать как объединение всех внутренних базовых тестов, выполняемых до внесения результатов разработки в систему управления версиями.

Модульное тестирование

Это особая форма внутреннего *базового*, или *префиксного*, тестирования, в котором применяются средства автоматизации. Основной принцип *помодульного* тестирования заключается в независимом тестировании отдельных функций (по одной за раз).

Функциональные тесты

Как правило, *функциональные* тесты определяются пользователями решения и создаются командой тестирования. Обычно эти тесты автоматизированы и ориентированы на проверку работы сквозных цепочек операций, а не отдельных функций. Их выполняет команда тестирования.

Проверка корректности процесса сборки

Цель этих тестов — выявить ошибки в процессе сборки. Их можно рассматривать как поиск ошибок компиляции (в противоположность ошибкам времени выполнения). Такое тестирование *могут* выполнять как разработчики, так и *тестирующие* специалисты, хотя в некоторых проектах для этого предусмотрена отдельная команда.

Регрессионное тестирование

Регрессионное тестирование — это процесс повторения определенной последовательности *тестировочных* операций с каждой новой версией или сборкой решения, чтобы определить:

- не «всплыла» ли в новой сборке старая неполадка;
- полностью ли устранена ошибка;
- не появились ли новые неполадки при устранении обнаруженных ранее ошибок.

Конфигурационное тестирование

Большинство программных решений допускает несколько способов установки и конфигурирования. Задача конфигурационного тестирования — оттестировать решение в каждой из возможных его конфигураций, так как у каждой из них свои особенности и возможность проявления тех или иных неполадок или ошибок.

Тестирование совместимости

Часто от разрабатываемого решения требуется возможность интеграции и способность к взаимодействию с существующими системами и программными решениями. Данная форма тестирования ориентирована на проверку интегрируемости и способности разрабатываемого решения взаимодействовать с существующими системами.

К такому тестированию часто привлекаются группы, внешние по отношению к проектной команде. Стороннюю команду следует привлечь как можно раньше, чтобы предусмотреть ее участие в плане. Кроме того, выполнение этих тестов иногда невозможно в полностью изолированной среде.

Нагрузочное тестирование

Нагрузочные тесты специально разработаны для выявления неполадок и ошибок, проявляющих себя только при высокой нагрузке. Подвергая решение перегрузке, что, как правило, означает нагрузку *большую*, чем та, на которую оно

рассчитано, удастся обнаружить дефекты, не проявляющие себя в нормальных условиях.

Тестирование производительности

Это вид тестирования ориентирован на проверку того, удовлетворяет ли приложение требования по производительности. Часто его совмещают с нагрузочным тестированием. Например, в проекте разрабатываемого решения предусмотрено увеличение числа пересылаемых сообщений электронной почты на 15%; при тестировании следует оценить, соответствует ли реальный прирост производительности ожидаемому.

Тестирование документации и справочной системы

Тестируются все разработанные сопровождающие документы и справочные системы. Документы и файлы справочной системы проверяются на предмет соответствия целям проекта. Например, тестировщики должны выяснить, нет ли в справочных материалах и документации некорректных инструкций, устаревшей информации и снимков экранов, опечаток и т.п.

Альфа- и бета-тестирование

В терминах MSF *альфа-код* — это в основном все исходные тексты, созданные на этапе разработки модели процессов MSF, а *бета-код* — код, прошедший тестирование на этапе стабилизации. Поэтому на этапе разработки модели процесса MSF тестируется *альфа-код*, а на этапе стабилизации — *бета-код*.

Параллельное тестирование

Параллельное тестирование — это популярный метод тестирования пользовательских функций, когда одновременно испытываются существующее и новое решение. Преимущество такого тестирования заключается в возможности быстро проверить, ведет ли себя решение именно так, как ожидается. Например, одни и те же данные последовательно загружаются в существующее и новое решение, наблюдается их поведение и на основании результатов такого исследования делается вывод о ценности решения для пользователей.

Совет по планированию Параллельное тестирование можно автоматизировать — это позволяет быстро получить результаты без привлечения реальных пользователей. А недостаток параллельного тестирования в том, что, если оно не поддается автоматизации, пользователям придется выполнять двойную работу.

Классификация и контроль ошибок

Невозможно создать качественное промышленное решение без системы мониторинга неполадок и ошибок. Без такой системы нельзя определить, отвечает ли продукт критерию бездефектности.

Классификация

Процесс деления неполадок или ошибок на категории в MSF основан на системе определения категорий рисков в процессе управления рисками MSF. В системе классификации рисков в MSF члены команды должны оценивать вероят-

ность и возможное влияние определенного риска, а затем эти величины перемножаются для получения численного значения, характеризующего уровень риска и используемого для определения важности риска.

Система категорий и важности неполадок и ошибок работает *похожим* образом. Риск подразумевает возможность неблагоприятного события, а его важность оценивается двумя параметрами — вероятностью возникновения и опасностью. В отличие от рисков ошибки — это то, что уже существует. Далее перечислены важные параметры ошибок.

- **Повторяемость** (repeatability) показывает, насколько часто повторно встречается ошибка или проблема, и измеряется как процентное отношение времени проявления ошибки к общему времени испытания. Например, ошибка может повторяться всегда или только в 10% случаев. Обратите внимание, что повторяемость не может равняться нулю, поскольку это означало бы, что ошибка не проявляется вовсе.
- **Обнаруживаемость** (visibility) определяет ситуацию или среду, в которой ошибка дает о себе знать. Например, если «жучок» обнаруживается, только когда пользователь удерживает нажатой клавишу *Shift* при двойном щелчке мышью и открытии меню *Файл*, то велика вероятность, что ошибка не будет обнаружена. Ошибка или проблема, возникающая лишь при особых условиях, труднее поддается обнаружению и, следовательно, характеризуется низкой обнаруживаемостью.
- **Серьезность** (severity) отражает степень влияния неполадки на решение, код или пользователя. Например, одна ошибка может инициировать аварийное завершение работы операционной системы или приложения, а другая приводит всего лишь к изменению цвета одного пиксела в строке меню. Для корректного определения серьезности ошибки проектная команда должна составить список проблем с указанием их серьезности. Например, уровень 10 соответствует аварийному завершению работы системы с потерей данных и невозможностью их восстановления, уровень 9 — аварийному завершению работы системы с потерей данных, которые удастся восстановить.

Эти три переменные оцениваются и назначаются каждой неполадке или ошибке и используются для определения ее важности. В таблице 11-2 описаны шкалы для каждой из переменных.

Таблица 11-2. Переменные для классификации ошибок

Переменная	Шкала
Повторяемость	Процентное значение в диапазоне от 10% (числовое значение 0,1) до 100% (числовое значение 1), где 100% соответствует неполадке или ошибке, которая проявляется при каждом прогоне определенного сценария тестирования
Обнаруживаемость	Процентное значение в диапазоне от 10% (числовое значение 0,1) до 100% (числовое значение 1), где 10% соответствует неполадке или ошибке, которая обнаруживает себя только в <i>очень редких</i> и, как правило, невыясненных до конца условиях. Считается, что неполадки или ошибки, проявляющие себя в обычных условиях, обладают высокой степенью обнаруживаемости
Серьезность	Целое число в диапазоне от 1 до 10, где 10 соответствует ошибке, оказывающей самое сильное воздействие на решение или код

Важность ошибки обычно вычисляют по следующей формуле:

$$\text{Важность} = (\text{Повторяемость} + \text{Обнаруживаемость}) \times \text{Серьезность}$$

В таблице 11-3 показан пример расчета степени важности ошибок, представленный в виде матрицы.

Таблица 11-3. Матрица важности ошибок

Номер ошибки	Повторяемость	Обнаруживаемость	Серьезность	Важность
1	1	1	10	20,0
2	0,9	0,9	9	16,2
3	0,8	0,8	8	12,8
4	0,7	0,7	7	4,8
5	0,6	0,6	6	7,2
6	0,5	0,5	5	5,0
7	0,4	0,4	4	3,2

Процесс отслеживания неполадок или ошибок

Как показано на рис. 11-3, выявляют ошибки сотрудники, выполняющие роли «разработчик» и «тестировщик».



Рис. 11-3. Процесс мониторинга неполадок и ошибок

Процесс отслеживания ошибок состоит из следующих стадий.

1. Разработчики пишут код, выполняют внутреннее базовое тестирование и затем размещают код в системе управления версиями.
 2. Тестировщики выполняют ежедневные или периодические сборки и проводят внешнее базовое тестирование всего нового кода.
 3. Тестировщики вносят сведения о проблемах или ошибках в систему мониторинга неполадок и ошибок, предоставляя описание проблемы и определяя повторяемость, серьезность и обнаруживаемость каждой ошибки.
 4. Ведущие разработчики и тестировщики встречаются, чтобы:
 - определить важность каждой неполадки или ошибки. Это задача стадии «Назначение важности неполадке/ошибке» в методике тестирования MSF;
 - назначить разработчиков, ответственных за устранение неполадок и ошибок, которые не удовлетворяют критерию бездефектности;
 - удалить сведения о неполадках или ошибках, которые были обнаружены в предыдущем цикле подпроцесса, но уже устранены.
 5. Назначенные разработчики устраняют или корректируют неполадки или ошибки.
 6. Разработчики выполняют внутреннее базовое тестирование и помещают скорректированный код в систему управления версиями.
- Этот повторяющийся процесс выполняют снова при завершении разработки и размещении очередной порции кода в системе управления версиями.

Задачи тестирования

Методика тестирования MSF предусматривает выполнение двух важных задач: *проверку кода* и *создание среды тестирования*. Они очень важны для успешной реализации методики тестирования MSF.

Проверка кода

Методика тестирования MSF определяет три типа проверок кода.

- **Полная проверка (comprehensive)** — это формальная проверка, при которой разработчик представляет свой код команде или осуществляет полный пошаговый проход исходного текста.
- **Выборочная проверка (casual)** — это анализ кода коллегами, членами команды. Этот способ очень хорош для «натаскивания» новичков.
- **Независимая проверка (independent)** выполняется сторонними организациями для получения свежей и непредвзятой оценки качества кода.

Создание среды тестирования

Очень важно еще до начала разработки спроектировать и построить среду разработки и тестирования. Она должна изолироваться от среды промышленной эксплуатации решения и одновременно удовлетворять требованиям разработки и тестирования. Далее перечислены обязательные условия для тестирования.

- **Аппаратное обеспечение для сборки решения**, позволяющее тестировщикам ежедневно (или с другой периодичностью) выполнять сборки.
- **Автоматизация тестирования** обеспечивает выполнение основной работы по тестированию без участия человека.
- **Создание систем управления версиями**. Необходимо предоставить тестировщикам доступ к этим системам.

- **Создание систем отслеживания неполадок и ошибок** и предоставление тестировщикам доступа к ним.

Пилотная эксплуатация

Пилотная эксплуатация (pilot) — это тестирование решения в промышленной среде, а также его проверка ответственными за установку ПО, персоналом отдела сопровождения и конечными пользователями. Основная задача пилотной эксплуатации — продемонстрировать, что решение способно стабильно работать в условиях промышленной эксплуатации и удовлетворяет требованиям бизнеса. Есть у тестирования и второстепенная задача — дать команде возможность «отрепетировать» и при необходимости скорректировать процесс развертывания.

В процессе пилотной эксплуатации решение испытывается в реальных условиях, причем форма тестирования зависит от типа и масштаба проекта.

- При тестировании бизнес-приложения масштаба предприятия пилотная конфигурация системы может состоять из группы пользователей и набора серверов в центре данных.
- При тестировании Web-приложений опытную конфигурацию можно размещать на промежуточных серверах или в каталогах, открытых для доступа из Интернета, однако для нее указывается тестовый Web-адрес сайта.
- Производители коммерческого ПО, например Microsoft, выпуск очередного продукта передают для тестирования определенному кругу пользователей задолго до начала промышленных продаж.

У различных форм пилотной эксплуатации один общий элемент — тесты выполняются в промышленной среде. Пилотная эксплуатация продолжается, пока команда не убедится в работоспособности решения в условиях «реального боя» и готовности всех его компонентов к развертыванию.

Процесс

Пилотная эксплуатация — последний важный шаг перед полномасштабным развертыванием решения. До него необходимо в тестовой среде выяснить, способно ли приложение к интеграции.

Пилотная эксплуатация дает возможность пользователям высказать свое мнение о работе продукта. Руководствуясь этим мнением, команда устраняет все возможные неполадки или создает план действий на случай непредвиденных обстоятельств. Отклики пользователей позволяют команде определить необходимый уровень поддержки развернутого решения, а также помогают при работе над следующими версиями решения.

Внимание! В конечном итоге, пилотная эксплуатация позволяет принять решение, стоит ли начинать полномасштабное развертывание или отложить до устранения неполадок, способных сорвать развертывание.

Пилотная эксплуатация — это итеративный процесс, который начинается с планирования (создание плана опытной эксплуатации и подготовка пользователей и мест развертывания решения), а затем переходит к собственно эксплуатации, оценке результатов, устранению неполадок и развертыванию следующих версий. Такая последовательность операций продолжается, пока качество и широта функциональности решения не достигают уровня, достаточного для полного развертывания.



Рис. 11-4. Пилотная эксплуатация

Подготовка к пилотной эксплуатации

Пилотное развертывание следует должным образом отрепетировать, чтобы свести к минимуму риск неприятностей при пилотной эксплуатации. На этом этапе команда разработки проводит последние проверки и следит, чтобы ничего не поменялось со времени тестирования перед пуском в эксплуатацию. До начала пилотной эксплуатации необходимо решить ряд задач.

- Команда разработки и участники пилотной эксплуатации должны четко определить критерии успешности пилотной эксплуатации и согласовать их.
- Следует создать структуру поддержки и разработать процесс устранения возникающих неполадок. Возможно, придется организовать дополнительное обучение для персонала службы поддержки. Процедуры по решению возникающих проблем в процессе пилотной эксплуатации могут в корне отличаться от тех, что будут применяться на стадии развертывания и промышленной эксплуатации.
- Чтобы выявить возможные неполадки и убедиться в беспрепятственности грядущего процесса развертывания, необходимо выполнить пробный запуск или провести «репетицию» всех стадий процесса развертывания.
- Необходимо заручиться одобрением плана пилотной эксплуатации от пользователей. Работа над планом начинается на ранних стадиях этапа планирования, так что к моменту, когда команда тестирования начнет пробный запуск, каналы связи уже налажены и все участники готовы.

Совет Пилотная эксплуатация задает общий настрой финального развертывания, вот почему так важна тщательность проработки и высокий уровень информированности всех участников, а также глубокий анализ результатов.

Рекомендуемый план пилотной эксплуатации включает:

- область действия и задачи;
- участников, их местоположение и контактную информацию;
- план обучения пользователей, участников испытания;
- план поддержки на время пилотной эксплуатации;
- план взаимодействия;
- возможные риски и планы по обеспечению непрерывности бизнеса;
- план отката;
- календарный план развертывания и пилотных испытаний.

Проведение пилотной эксплуатации

Не поддавайтесь соблазну упростить пилотное тестирование. Отработайте по возможности все варианты и сценарии использования системы, которые будут применяться в промышленной эксплуатации, группируя различные ВИС и СИС. Также рекомендуется намеренно вызвать крах пилотной эксплуатации с тем, чтобы отработать процедуры отката, восстановления после сбоев и обеспечения непрерывной работы бизнеса.

Оценка результатов

По завершении пилотной эксплуатации оценивается успешность продукта и формулируются рекомендации по реализации следующего шага. Проектная команда должна решить, переходить ли к этапам, следующим за пилотным прогоном. Чтобы выполнить оценку и дать корректные рекомендации, проанализируйте информацию из различных источников. К ним относятся:

- формы для отзывов, размещенные на Web-сайте;
- встречи с менеджерами организации;
- отчеты об обнаруженных проблемах;
- опросы конечных пользователей;
- замечания ИТ-команды проекта;
- журналы операционной системы и приложения.

Важно получить информацию о процессах как дизайна, так и развертывания.

Оцените, что сработало отлично и какая часть приложения показала себя хуже, чем ожидалось, и соответственно скорректируйте и усовершенствуйте план. Следует собрать информацию об:

- обучении, необходимом пользователям для нормальной работы с решением;
- процессе развертывания;
- необходимой поддержке продукта;
- связях;
- обнаруженных неполадках и ошибках;
- пожеланиях по улучшению решения.

Обратная связь требуется для проверки соответствия дизайна спецификации, а также требованиям бизнеса. Пилотный запуск оценивается для получения ответов на следующие вопросы:

- соответствует ли пилотная эксплуатация критериям, сформулированным до ее начала;

- если были определены параметры, по которым оценивалась успешность пилотной эксплуатации, то следует «измерить» ее в этих параметрах и оценить, насколько полученные значения соответствуют ожидаемым.
После оценки данных команда должна принять решение и выбрать одну из стратегий.
- **Медленное продвижение вперед.** Подготовить очередную версию-кандидат на выпуск и предоставить его целевой группе, а затем поочередно другим группам. Предоставление решения нескольким группам может быть запланировано заранее или же стать следствием неприемлемых результатов первого опытного запуска.
- **Откат.** Возвращение пилотной группы к ее предыдущему состоянию.
- **Приостановка пилотной эксплуатации.** Приостановка или отмена внедрения решения.
- **Исправление и продолжение эксплуатации.** Исправить ошибки в сборке, участвующей в пилотном запуске, и продолжить работу над решением.
- **Перейти к этапу развертывания.** Перейти к развертыванию пилотной сборки в промышленных условиях.

Результаты пилотной эксплуатации

По завершении пилотного тестирования команда готовит отчеты обо всех «уроках», а также о способах обработки полученной информации и разрешении выявленных проблем. К результатам пилотного тестирования относятся:

- определение прочих рисков;
- часто задаваемые вопросы, которые пригодятся при обучении;
- часто совершаемые пользователями ошибки, которые необходимо описать в документации и также внести в программу обучения;
- возможное одобрение и поддержка со стороны пользователей, участвующих в пилотной эксплуатации;
- документация об особенностях работы продукта и устранения неполадок;
- обновление документации, особенно файлов справочной системы и планов развертывания;
- информация о том, все ли критерии успешности выполнены.

Закрытие этапа стабилизации

Закрытие этапа стабилизации соответствует достижению контрольной точки процесса одобрения. Команда должна задокументировать результаты выполненных ею задач с тем, чтобы передать проект на одобрение руководству.

Завершающий момент этапа стабилизации — контрольная точка «Подтверждение готовности к выпуску». Она достигается, когда решены все текущие проблемы, продукт готов к выпуску и к полному развертыванию. В этой контрольной точке заказчикам и пользователям, административному и сопровождающему персоналу и ключевым участникам проекта предоставляется возможность оценить решение и выявить оставшиеся ошибки, которые необходимо решить перед переходом к развертыванию и, в конечном счете, к выпуску продукта.

После завершения всех связанных со стабилизацией задач команда должна прийти к формальному соглашению о достижении контрольной точки готовности к выпуску. При переходе от этой контрольной точки к следующему за ней этапу развертывания ответственность за дальнейшие поддержку и сопровожде-

ние решения официально переходит от проектной команды к административному и обслуживающему персоналу.

Проектные команды, как правило, отмечают прохождение этой контрольной точки формальной процедурой завершения работы над проектом. Ключевые фигуры со стороны заказчика (обычно это представители каждой из ролей команды и прочие важные лица) подтверждают прохождение контрольной точки, подписывая документ о ее достижении. Этот документ становится результатом проекта и сдается в архив на хранение.

Занятие 3. Этап развертывания в MSF

Контрольная точка «Подтверждение готовности к выпуску» в конце этапа стабилизации свидетельствует о готовности решения к развертыванию и работе в промышленной среде.

На этом этапе команда развертывает необходимые для решения технологии и компоненты, а также стабилизирует развернутую систему, проект переходит на стадию сопровождения и поддержки, а заказчик окончательно утверждает его. После развертывания команда проводит оценку проекта и опрос пользователей, чтобы выяснить степень их удовлетворенности. В этот период можно продолжать операции по стабилизации, так как компоненты проекта переводятся из тестовой в промышленную среду.

Изучив материал этого занятия, вы сможете:

- ✓ описать динамику состава команды в процессе развертывания;
- ✓ рассказать о целях, контрольных точках и результатах этапа развертывания.

Продолжительность занятия — около 15 минут.

Контрольные точки и результаты этапа развертывания в MSF

К основным задачам этого этапа в MSF относятся подготовка к развертыванию и собственно развертывание решения. После развертывания, стабилизации и перехода к сопровождению проектная команда выполняет оценку проекта.

Этап завершается достижением контрольной точки «Решение развернуто». К этому моменту созданы:

- системы сопровождения и поддержки;
- процедуры и процессы;
- база знаний, отчеты и журналы;
- хранилище документов, где размещаются все версии документов, конфигураций, сценариев и кода, разработанных в течение проекта;
- отчет о закрытии проекта:
 - проектные документы;
 - результаты опроса пользователей;
 - план дальнейших мероприятий.

Основные задачи команды на этапе развертывания

Существует несколько вариантов перехода проектной команды от этапа стабилизации к этапу развертывания. Один из них — развертывание силами имеющейся в компании заказчика группы сопровождения. Если развертывание полностью выполняется этой группой, представители команды разработки, как правило, продолжают участвовать в проекте еще некоторое время, чтобы устранить неполадки, возникающие в процессе передачи продукта. Другой вариант — создать отдельную команду развертывания, пригласив в нее отдельных членов из команды разработки и группы сопровождения. Роль «менеджер выпуска» отвечает за координирование развертывания,

В таблице 11-4 перечислены области ответственности каждой из ролей команды в процессе развертывания.

Таблица 11-4. Роли и обязанности ролей на этапе развертывания

Роль	Выполняемые задачи
Менеджер решения	Поддержка связей с заказчиком, оценка и завершение проекта
Менеджер программы	Проверка соответствия решения его области действия, управление стабилизацией
Разработчик	Устранение неполадок, поддержка развертывания решения
Специалист по удобству использования	Обучение, управление графиком обучения
Тестировщик	Тестирование решения, выявление, определение и разрешение неполадок, а также отчеты по ошибкам решения
Менеджер по выпуску	Управление по развертыванию на местах, одобрение изменений

Сценарии развертывания

Вот некоторые из развертываемых решений: Web-приложения и Web-сервисы, клиент-серверные приложения, наборы приложений, системы масштаба предприятия и мобильные приложения.

Модель процессов MSF работает независимо от типа проекта, который предстоит развернуть, однако сложность и продолжительность развертывания сильно меняется в зависимости от вида и размера приложения. Например, Web-приложение развертывается довольно просто, требует минимума усилий и не вызывает фундаментальной перестройки инфраструктуры, географического распределения и конфигурации рабочих станций, а инфраструктурные проекты, предусматривающие настройку рабочих станций, — гораздо более длительный и трудоемкий процесс. Очень важно, чтобы выбранный командой вариант развертывания решения соответствовал типу проекта.

Кроме сценариев, при развертывании решения необходимо учесть типы аппаратного обеспечения и требования к операционным системам. Сценарий развертывания решения сильно зависит от места развертывания — в центре обработки данных, группе промышленных серверов или на мобильных устройствах. Далее перечислены некоторые типы аппаратного обеспечения и требования операционных систем, которые определяют сценарии развертывания:

- корпоративные серверы;
- центры обработки данных (доступные через Интернет, локальные для отделов, глобальные);
- Web-сервисы;
- клиенты (ПК, мобильные).

Занятие 4. Развертывание в промышленной среде

На этом занятии рассказывается об особенностях развертывания решения в промышленной среде и стадиях этого процесса.

Изучив материал этого занятия, вы сможете:

- ✓ рассказать о планировании развертывания;
- ✓ отличить базовые компоненты от характерных для конкретного места установки;
- ✓ развернуть базовые компоненты;
- ✓ развернуть компоненты «на местах»;
- ✓ объяснить, как подготовиться к сопровождению;
- ✓ описать стадии развертывания решения в промышленной среде;
- ✓ рассказать о завершающих операциях этапа развертывания.

Продолжительность занятия — около 30 минут.

Планирование развертывания

В процессе разработки, и особенно в конце этапа стабилизации, ведущий менеджер выпуска распределяет задачи по развертыванию между членами команды. Команда должна оценить состояние проекта и результаты тестов, а также обновить план развертывания, созданный еще на этапе планирования. Кроме того, необходимо разработать последовательность процедур, которые обеспечат успех развертывания.

Тестовая среда и среда промышленной эксплуатации

В процессе подготовки к развертыванию физическая инфраструктура, системное и прикладное ПО тестируется, устанавливается и конфигурируется в среде, которая максимально точно копирует промышленную.

Документация

Проектная команда и представители службы поддержки анализируют и обновляют документацию:

- **диаграммы развертывания**, созданные на этапе планирования;
- **план тестирования** всех областей, которые будут затронуты внедряемым решением. Команда сопровождения должна обновить планы тестирования, созданные на этапе планирования;
- **план обеспечения безопасности**, необходимый команде сопровождения для обучения всего персонала стандартам безопасности и правилам, которые должны твердо соблюдаться в проекте;
- **план резервного копирования** для предотвращения потерь данных. На этапе планирования команда создает план процессов, в котором описываются выполняемые командой операции по резервному копированию файлов и данных для предотвращения их потери. План обновляется при планировании развертывания;

- **план анализа производительности системы и загруженности мест установки.** Команда сопровождения отвечает за ежедневную поддержку и выполнение прочих регулярных обслуживающих операций;
- **план работы с журналами** и прочих административных задач;
- **план восстановления после сбоев**, который команда создает на этапе планирования. В нем описано, какие меры следует принимать по отношению к продукту, оборудованию, персоналу и данным в случае аварии. При планировании развертывания команда должна регулярно пересматривать и корректировать этот документ;
- **план обеспечения непрерывности бизнеса**, который команда создает на этапе планирования. В нем описываются мероприятия, проводимые в случае приостановки деятельности предприятия. При планировании развертывания команда должна регулярно пересматривать и корректировать этот план;
- **информацию об обучении.** Команде следует обеспечить должный уровень навыков персонала службы поддержки решения, а также предоставить четкие инструкции на случай сбоя системы.

Проверка плана развертывания

План развертывания создается на этапе планирования и постоянно корректируется в процессе разработки. Однако при планировании развертывания менеджер выпуска может внести в план развертывания изменения, обусловленные коррективами решения в процессе разработки и стабилизации. Менеджер выпуска должен пересмотреть и скорректировать план развертывания и подтвердить, что выполнены следующие задачи:

- проверена и одобрена стратегия развертывания;
- готовы, проверены и одобрены процедуры установки, конфигурирования, тестирования, сопровождения и поддержки;
- задокументированы, проверены и одобрены процедуры развертывания;
- **готовы** и оттестированы компоненты аппаратного и программного обеспечения;
- роли команды развертывания четко определены, а ее члены определены и обучены;
- существует план и достаточно ресурсов для передачи решения в руки службы поддержки.

Окончательное одобрение со стороны заказчика

Ключевые участники проекта проверяют документацию и решение и подтверждают их готовность к развертыванию. Проектная команда **подготавливает** соответствующий официальный документ, подтверждающий, что заинтересованные лица проверили решение и сопутствующую документацию и признали решение окончательным и пригодным для внедрения.

Базовые компоненты и компоненты, характерные для отдельных мест установки

Для эффективного развертывания очень важно разделить компоненты на базовые и характерные для конкретных мест установки.

Базовые компоненты

Они расположены в центральном местоположении и обеспечивают работу решения в целом. В основном к ним относятся технологии, обеспечивающие работу решения. Вот несколько примеров базовых компонентов:

- контроллеры доменов;
- сетевые маршрутизаторы;
- серверы баз данных;
- почтовые маршрутизаторы;
- серверы удаленного доступа.

Компоненты, характерные для конкретных мест установки

Они расположены в отдельных местах установки и обеспечивают доступ и работу пользователей с решением. К ним относятся:

- локальные маршрутизаторы;
- файловые серверы и серверы печати;
- клиентские приложения, такие, как Microsoft Office XP.

В большинстве случаев базовые компоненты развертывают раньше индивидуальных. Например, к моменту развертывания Microsoft Exchange Server, необходимо реализовать магистральную сеть, объединяющую все места установки, и лишь затем развертывать компоненты узлов.

Развертывание базовых компонентов

Развертывание базовых компонентов подразумевает выбор адекватной стратегии и само развертывание. Выбор стратегии требует глубокого понимания как решения, так и требований заказчика.

Базовые компоненты часто используются в разных местах решения, поэтому представляют собой критически важную часть продукта. Решая вопрос о порядке развертывания решения, важно выяснить, какие из компонентов не жизненно важны для работы решения — это позволит выбрать наиболее эффективную стратегию развертывания. В большинстве проектов затраты на одновременное развертывание всех базовых компонентов непомерно высоки и часто неоправданны.

Совет Дублирующие устройства, устанавливаемые для обеспечения избыточности или «запаса прочности» решения, до развертывания на местах устанавливать не обязательно.

Для развертывания базовых компонентов применяются две основные стратегии.

- **Последовательное развертывание.** Все базовые компоненты развертываются до начала развертывания компонентов на местах установки. Этот подход менее рискован и годится для быстро внедряемой, небольшой инфраструктуры.
- **Параллельное развертывание.** Базовые компоненты развертываются по мере необходимости параллельно с развертыванием базовых технологий на местах. Такой подход более экономически оправдан в крупных средах или проектах, на развертывание которых к тому же отведено довольно длительное время.

В зависимости от сценария решения иногда требуется, чтобы базовая технология, на которой основано решение, устанавливалась до или одновременно с развертыванием решения.

Развертывание компонентов для индивидуальных мест установки

Развертывание на местах можно считать процессом в процессе. Оно предусматривает выполнение тщательно продуманного плана установки решения (рис. 11-5).

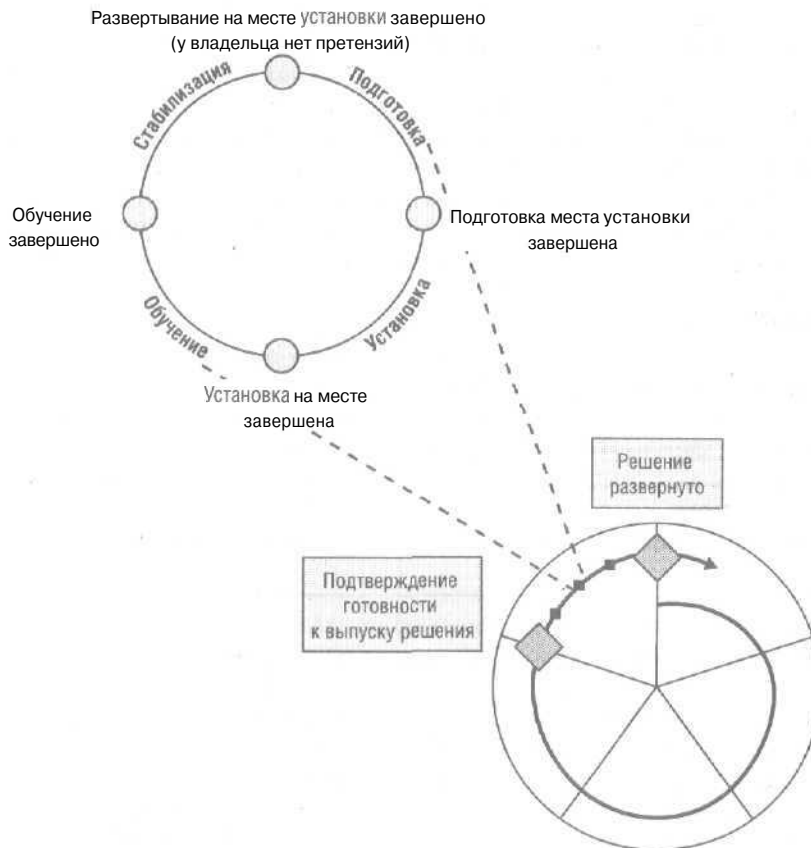


Рис. 11-5. Процесс развертывания в одном из мест установки

Развертывание можно проводить в каждом месте последовательно (при этом потребуются меньше команд) или параллельно (необходимо больше команд). Последнее требует более качественной координации и предоставляет меньше возможностей для постепенного наращивания нагрузки на решение. Однако последовательное развертывание иногда вызывает у пользователей замешательство, особенно если новое решение должно сосуществовать с уже действующими системами.

Развертывание на местах также подразумевает, что пользователи будут работать с системой в режиме промышленной эксплуатации. Команда должна предпринять определенные меры для создания необходимой инфраструктуры сопровождения и поддержки этих пользователей, когда они начнут работать с системой. Развертывание каждого места установки можно подразделить на следующие четыре этапа:

- подготовку к развертыванию;

- установку компонентов;
- обучение в процессе развертывания;
- стабилизацию.

Подготовка к развертыванию

Подготовка к развертыванию в одном месте установки состоит из трех операций: определения списка наличных ресурсов, составления графика и информирования. Координатор часто выполняет эти операции удаленно. Он проверяет всю информацию о развертывании, собранную при планировании, создает окончательный календарный график и информирует команду о времени начала развертывания. Эта стадия необходима для проверки информации о месте установки с тем, чтобы убедиться, что не произошло существенных изменений со времени сбора требований на этапе планирования. Другая задача стадии — проверить наличие всего необходимого для развертывания.

Установка компонентов

Чтобы как можно меньше влиять на нормальное течение процессов, команда не начинает установку решения, пока не закончены все приготовления. В некоторых случаях возникающие в последний момент проблемы, не учтенные на подготовительном этапе, заставляют отложить развертывание.

Примечание В некоторых случаях команда решает отложить запуск до тех пор, пока решение не станет стабильным и пользователи не пройдут соответствующее обучение.

В процессе установки выполняются следующие задачи:

- установка или обновление необходимого аппаратного и программного обеспечения;
- проверка корректности работы установленных систем;
- предоставление системы в распоряжение пользователей.

В процессе развертывания команду следует обеспечить механизмом поддержки на случай возможных неполадок проблем. Как правило, разработчики и тестировщики основной команды проекта должны быть готовы оказать помощь команде, занимающейся развертыванием.

Обучение в процессе развертывания

В обязанности команды развертывания вменяется обучение всех пользователей. На этапе планирования проектная команда разработала план обучения для всех групп пользователей. Обучение можно проводить до, в процессе или после установки компонентов.

Во время обучения в процессе развертывания в месте установки следуйте рекомендациям:

- каждому пользователю необходима своя, предназначенная только для него программа;
- обучающий материал следует варьировать исходя из знаний и опыта каждого пользователя;
- необходимо применять предусмотренные учебным планом мультимедийные средства;
- обязательно обучение местного персонала поддержки.

Ни в коем случае не учите по принципу «всех — под одну гребенку». Если вы создадите единую учебную программу, то новичкам будет сложно освоить материал, а опытные пользователи будут откровенно скучать. Подумайте об альтернативных вариантах обучения, таких, как сеансы интенсивного обучения и программы с наставником.

Стабилизация

Стабилизация — важная часть процесса развертывания узла. Не стоит самоустраняться, установив решение, — это практически всегда плохая идея. Не рекомендуется бросать пользователей на произвол, пока они не ощутят себя достаточно комфортно при промышленной эксплуатации, — и эта обязанность возлагается на команду развертывания. Для этого команде следует применять критерии успешности проекта, получить одобрение заказчика и отзывы удовлетворенных пользователей, а также всеми силами способствовать нормальному течению процесса развертывания. Развертывание в каждом месте установки должно сопровождаться официальной сдачей-приемкой работ. На этапе планирования в каждом месте установки назначается сотрудник, уполномоченный принимать установленное решение.

Период затишья

Так называемый период затишья начинается после завершения развертывания. В это время команда передает свои полномочия группе сопровождения и поддержки, которая будет обслуживать продукт. Он длится примерно 15–30 дней, в течение которых собираются статистические данные, касающиеся поведения системы. Данные, полученные в период затишья, становятся базой для создания соглашения об уровне сервиса.

План поддержки и критерий приемки решения, определенные проектной командой на этапе планирования, определяет уровни сервиса, которые становятся основой соглашения об уровне сервиса (*service-level agreement, SLA*); последнее определяет работу персонала поддержки после передачи им ответственности за работу продукта.

SLA — это соглашение между поставщиком услуг и заказчиком, которое определяет обязанности обеих сторон, в том числе обязанности поставщика предоставлять определенный сервис оговоренного качества и в оговоренном объеме. Оно ограничивает требования, выдвигаемые заказчиком, рамками, определенными в соглашении.

Передача проекта команде сопровождения и поддержки

Завершение проекта предусматривает передачу функций по сопровождению и поддержке персоналу заказчика. В большинстве случаев ресурсы для управления новой системой уже есть, иначе приходится разрабатывать новые системы поддержки.

Далее перечислены задачи, которые выполняются при передаче проекта команде сопровождения и поддержки:

- **Активизация систем отчетности о работе системы.** Не забудьте переадресовать звонки пользователей и просьбы об устранении неполадок в информационно-справочную службу. Передайте обязанности по устранению системных сбоев, операциям по коррекции работы системы и анализу тенденций службе поддержки.

- **Публикация базы знаний.** База знаний обеспечивает простой доступ к информации о поправках, к ней обращается как команда сопровождения, так и пользователи.
- **Проверка процедур.** Перед передачей системы в руки команды поддержки удостоверьтесь в полноте и корректности выполнения всех процедур.

Примечание Очень важно, чтобы проектная команда передала обязанности по развертыванию группе поддержки **компании-заказчика**. Если этого не произойдет, проектная команда сама превратится в группу поддержки. **Заказчик** будет продолжать беспокоить команду вопросами и требованиями по устранению неполадок.

Закрытие проекта

По завершении развертывания и передачи проекта в собственность заказчику, а обязанностей по поддержке — службам поддержки и сопровождения команда переходит к процедуре закрытия проекта.

После того как заказчик подписывает акт приемки, проект считается официально завершенным, а контрольная точка «Решение развернуто» — достигнутой. Далее перечислены задачи, выполняемые при закрытии проекта.

- **Опрос пользователей для определения их удовлетворенности.** Один из механизмов обратной связи — получение отзывов от заказчика. Это важная часть отчета о закрытии проекта, он служит подтверждением успеха проекта. Опрос должен как минимум охватить спонсоров проекта и ключевых участников, в том числе и заинтересованных лиц.
- **Подготовка отчета о закрытии проекта.** Отчет о закрытии проекта — последний физический результат развертывания. Он содержит финальные версии всех основных результатов: документа об общей картине и области действия решения, функциональной спецификации и т.д.

Отчет о закрытии проекта, как правило, включает сводку полученной от заказчика и пользователей информации и краткое описание того, что следует предпринять далее. В сущности, отчет дает ответ на вопрос: «Куда двигаться дальше?»

- **Оценка проекта.** Как правило, команда оценивает проект с точки зрения команды и с точки зрения заказчика. Большинство технологических проектов заканчиваются встречей, или *анализом проекта*, на котором члены команды оценивают проект и определяют области, над которыми следует лучше поработать в следующих проектах.

Анализ проекта — это возможность для всей команды подвергнуть разбору процесс и итоги проекта. Команда документирует возникшие при оценке проекта соображения, чтобы применить их в будущих проектах.

Внимание! Дополнительно оценку проекта выполняют вместе со спонсорами и ключевыми заинтересованными лицами для определения степени удовлетворенности заказчика, выявления нерешенных проблем и принятия решения о том, что следует предпринимать далее.

- **Одобрение заказчиком.** По завершении проекта менеджер решения получает у заказчика подпись на акте приемки, что демонстрирует одобрение решения заказчиком и является свидетельством того, что проектная команда слагает с себя полномочия.

Практикум. Определение важности ошибок



При выполнении упражнений используйте знания, полученные на занятиях по определению важности ошибок.

Команда тестирования проекта протестировала Web-приложение электронной коммерции и обнаружила ошибки, которые перечислены далее (степень их важности пока не известна).

- В 12 из 75 тестов браузер «зависал», когда пользователь попытался отправить заказ на сервер. В этих случаях статус заказа на сервере свидетельствовал о завершении оформления заказа, но пользователь не получал подтверждения или номера заказа.
- Когда торговый представитель предоставляет скидку более 15%, система принимает ее, не требуя подтверждения от менеджера.
- Когда пользователь просматривает страницу подтверждения заказа и щелкает ссылку «Продолжить выбор товара», он оказывается на странице входа в систему. Создается впечатление, что такая ситуация возникает, когда длина имени пользователя превышает 8 символов. В тестовой базе данных ошибка наблюдалась у 20 пользователей из 172.
- При входе покупателя в систему цвет фона страницы справочной системы (обычно белый) приобретает грязноватый оттенок.
- В 50 тестах на странице «Отслеживание заказов» дважды отобразились данные о не соответствующих действительности заказах.
- Нагрузочный тест, имитирующий одновременную работу 1000 пользователей, показал падение производительности на 18%. Среднее число пользователей, одновременно работающих с Web-сайтом электронной коммерции — 300.
- На странице «Подтверждение заказа» отсутствует уведомление об авторском праве.
- В 6 из 130 тестов начальная страница не загрузилась. Для отображения пришлось обновлять страницу.
- Каждый раз при загрузке домашней страницы пропадает кнопка «Просмотр корзины». Это происходит, даже если в корзине уже есть выбранный товар.
- В 2 из 40 тестов менеджер по продажам не смог предоставить скидку, превышающую 15%. Во всех тестах использовался одна и та же учетная запись менеджера, но заказы состояли из разных товаров.
- В 8 из 50 тестов при щелчке ссылки «Продолжить совершать покупки» на странице «Подтверждение заказа» возникла ошибка 404 протокола HTTP (HTTP 404 - File Not Found).
- Не отображается список каталогов при просмотре подробной информации о велосипедных шлемах.
- Если менеджер по продажам при просмотре подробной информации о товаре нажимает клавишу *Enter*, в браузере открывается страница справочной системы.
- В 3 из 25 тестов не загрузилась страница «Подтверждение заказа».
- В 16 из 25 тестов на странице «Управление заказами» не загрузилась детальная информация о заказе.
- Каждый раз, когда администратор пытался подключиться к сайту администрирования по протоколу SSL, он получал следующее сообщение об ошибке: «Невозможно отобразить страницу. Запрашиваемая вами страница в настоящее время недоступна. Возможно, это связано с техническими проблемами».

Web-сайта или неверной настройкой браузера». Администратор не в состоянии получить доступ к данным для управления онлайн-каталогом.

Подсчитайте повторяемость, обнаруживаемость и серьезность каждой из приведенных выше ошибок. Вывод о некоторых значениях можно сделать на основании представленной информации. Затем по приведенной в этой главе формуле вычислите важность каждой ошибки.

Для выполнения упражнения пользуйтесь таблицу в файле *C11Ex1.doc* в папке *\SolutionDocuments\Chapter1* на прилагаемом компакт-диске. Одно из возможных решений этого упражнения приводятся в файле *C11Ex1_Answer.doc* в той же папке.

Резюме

- Цель этапа стабилизации — снижение риска, возникающего при вводе решения в промышленную эксплуатацию.
- Для успеха этапа стабилизации необходимо, чтобы команда сменила отношение к проекту и переключилась с разработки новых функций на обеспечение должного качества решения.
- Цели этапаразвертывания:
 - перенести решение в промышленную среду;
 - обеспечить плавную передачу решения из рук проектной команды в руки группы сопровождения;
 - признание заказчиком факта завершения проекта.
- Развертывание компонентов, характерных для конкретного местаустановки, состоит из нескольких стадий: подготовки, установки, обучения и формального одобрения.
- Проектная команда прекращает работы в рамках проекта только после подписания заказчиком акта приемки.
- Результатами этапа развертывания являются системы сопровождения и поддержки, хранилище документов, где размещаются все версии документов и кода, разработанных в течение проекта.

Закрепление материала



Приведенные ниже вопросы помогут вам лучше усвоить основные темы данной главы. Если вы не сумеете ответить на вопрос, повторите материал соответствующего занятия. Ответы для самопроверки — в приложении «Вопросы и ответы» в конце книги.

1. Назовите некоторые особые цели этапа стабилизации.
2. Как анализ сходимости ошибок сигнализирует о стабилизации проекта?
3. Почему точка отсутствия ошибок считается положительной тенденцией проекта?
4. Что такое кандидат на выпуск и каковы его особенности?
5. Что такое пилотная эксплуатация и каковы ее цели?
6. Почему нельзя упрощать и облегчать процедуры пилотного тестирования?
7. Каковы результаты этапа развертывания?
8. Кто (или какая группа) лучше всех остальных выполняет развертывание решения?
9. Приведите примеры базовых компонентов. Приведите примеры компонентов, характерных для места установки.
10. Что такое «период затишья» и каковы его особенности?
11. Какие документы оформляются при закрытии проекта?
12. На какую из ролей MSF возлагается обязанность получить итоговую подпись от заказчика?
13. Назовите два типа оценки проекта, которые MSF рекомендует проводить в конце этапа развертывания?

ПРИЛОЖЕНИЕ

Вопросы и ответы

Глава 1

Закрепление материала

1. Опишите различия между водопадной и спиральной моделями и расскажите, как они используются в модели процессов MSF.

Водопадная модель основана на контрольных точках. Для достижения контрольной точки и перехода на следующий этап необходимо завершить все задачи предыдущего. Четко определяемые контрольные точки в этой модели позволяют легко контролировать ход и график проекта и определять обязанности и ответственность соответствующих членов группы. Эта модель наиболее подходит для проектов, где требования четко определены и не изменяются на протяжении всего проекта.

Основное назначение спиральной модели — предоставление возможности регулярно корректировать требования к продукту и оценку проекта. При каждой итерации проектная команда оценивает проект и планирует следующий цикл. Однако в спиральной модели отсутствуют четко определенные контрольные точки, поэтому оценка хода проекта затруднена. Эта модель эффективна при быстрой разработке приложений в небольших проектах.

В модели процессов MSF собрано все лучшее из водопадной и спиральной моделей: планирование на основе промежуточных контрольных точек и предсказуемость из водопадной модели наряду с обратной связью и коллективным творческим подходом, характерными для спиральной модели.

2. Кто согласно модели команд MSF отвечает за процесс проектирования?
За процесс проектирования отвечает менеджер программы.
3. Треугольник компромиссов описывает три типа возможных компромиссов. Какой существует четвертый вид компромисса и почему к нему никогда не следует прибегать?
Качество — еще одно измерение в дополнение к набору функций, ресурсам и календарному графику. Однако снижение требований к качеству — самая крайняя мера, к которой лучше не прибегать из-за возможных неприятных последствий.
4. Заполните матрицу компромиссов на основании приведенного ниже утверждения, «Учитывая, что зафиксирована функциональность, мы определим необходимые ресурсы и в случае необходимости скорректируем график».

	Зафиксировано	Определено	Корректируемо
Ресурсы		x	
График			x
Набор функций	x		

5. Объясните, зачем в модели процессов MSF используются ежедневные сборки.
Ежедневные сборки позволяют проектной команде и клиенту **регулярно контролировать** текущее состояние и **оценивать** стабильность проекта. Этот метод может применяться к проектам разработки как программного обеспечения, так и инфраструктуры оборудования.
6. Какой этап завершается контрольной точкой «Подтверждение готовности к выпуску продукта»?
 Этап **стабилизации** завершается **контрольной точкой** «Подтверждение готовности к выпуску продукта» и утверждением решения **проектной командой и клиентом**.
7. На каком этапе модели процессов MSF создается документ предварительной оценки рисков?
 На этапе **создания** общей картины решения начинается процесс оценки риска и создается **соответствующий документ**. Однако оценка риска не прерывается и продолжается на протяжении всего времени разработки проекта. Поэтому документ оценки рисков подвергается **изменениям** и обновлению по ходу проекта. Кроме того, в процессе работы над проектом в рамках **процедуры** оценки рисков создаются планы снижения риска.
8. Когда в модели процессов MSF создаются сценарии тестирования?
Основная часть тестов создается на **этапе разработки**. Однако **тестировщики участвуют** в проекте с самого его начала.
9. Перечислите виды тестирования, выполняемого на этапе стабилизации.
- Тестирование компонентов.
 - Тестирование баз данных.
 - Тестирование **инфраструктуры**.
 - Тестирование **защиты**.
 - Тестирование интеграции.
 - Проверка принятия продукта пользователями и удобства использования;
 - Нагрузочное **тестирование** продукта, а также тестирование ресурсоемкости и производительности.
 - Регрессное тестирование.
 - Регистрация числа ошибок.
10. Почему так важно создать документ общей картины решения на соответствующем этапе модели процессов MSF?
 Общая картина решения служит руководством команде для реализации бизнес-задач. Каждый член проектной команды должен одобрить и поддержать **общую** картину решения. Такое одобрение и поддержка необходимы для успеха проекта. Если члены команды не соглашаются с общей картиной решения, успех проект не гарантирован.
11. Перечислите ключевые задачи, выполняемые на этапе планирования?
- Разработка **функциональной спецификации**.
 - Создание промежуточной среды, а также сред разработки и тестирования.
 - Разработка проектных планов.
 - Создание календарного графика проекта.
 - Разработка **дизайна** и архитектуры решения.
12. Что такое «период затишья» и какие операции выполняются на этой стадии?
 «Период затишья» — это время между контрольными точками «Развернутое решение **стабилизировано**» и «Развертывание **завершено**». В **это** время **команда** занимается измерением производительности и быстродействия решения. Эти данные необходимы **для оценки** трудоемкости сопровождения и поддержки.

Глава 2

Закрепление материала

1. В чем отличие методов группового и **индивидуального** интервьюирования?
 Интервью — это встреча один на один члена проектной команды и пользователя или действующего лица. Интервьюер задает пользователю или участнику вопросы, **касающиеся** его работы и процессов.

Беседы с целевыми группами — это одна из форм группового интервью, в котором сотрудники, обсуждая определенную тему, предоставляют организатору встречи необходимую информацию. В группу должны входить представители всех задействованных пользователей и участников процесса. Применяйте этот метод, когда за отведенное время невозможно опросить всех пользователей. Члены целевой группы помогают друг другу заполнять пробелы в собираемой информации.

2. Когда при сборе информации следует предпочесть создание прототипов наблюдению за пользователями?

Наблюдение за действиями пользователя — это процесс сбора информации, заключающийся в наблюдении за тем, как пользователь выполняет свои задачи в реальной рабочей среде. Вы можете по ходу задавать пользователю любые необходимые вопросы. Но такой метод не позволяет выявить все задачи; из поля зрения выпадают задачи, выполняемые редко. Кроме того, иногда требуется документировать действия пользователя, например последовательность нажатия клавиш. В этом случае предпочтительнее метод создания прототипов.

3. Как выбрать наиболее эффективный метод сбора информации в конкретном проекте?

Чтобы определить наиболее эффективный метод сбора, необходимо рассмотреть преимущества и недостатки каждого метода. Следует определить тип информации, которую необходимо собрать, и количество отведенного на это времени. Если задачу нужно выполнить быстро, опросам рекомендуется предпочесть интервьюирование или беседы с целевыми группами.

4. Зачем создают варианты использования системы?

Варианты использования системы позволяют:

- определить бизнес-процесс и все действия от его начала и до конца;
- задокументировать проблемы контекста и среды;
- установить связь между потребностями бизнеса и требованиями пользователей;
- описать потребности и требования в контексте использования;
- сориентировать усилия пользователей и команды разработчиков.

5. Что такое каталог действующих субъектов?

Каталог действующих субъектов содержит информацию обо всех участниках ВИС, в том числе имя субъекта, его обязанности, источник этой информации.

6. Что такое ORM?

ORM — язык моделирования, который позволяет представлять информацию в виде элементарных фактов. Элементарный факт определяет, что у объекта есть свойство и один или несколько объектов участвуют в отношениях. В ORM модели выражаются в естественных понятиях, таких, как объекты и роли. ORM основан на предположении, что в реальности объекты играют определенные роли.

7. Что такое UML?

UML — это простой, расширяемый и выразительный язык визуального моделирования. Он представляет собой набор нотаций и правил моделирования программных систем различной степени сложности. UML позволяет создавать простые, хорошо документированные и легкие для понимания модели ПО.

8. Каково назначение различных UML-представлений?

Существует пять UML-представлений:

- пользовательское представление, которое выражает цели и задачи системы с точки зрения пользователей и их требований к системе;
- структурное представление, которое отражает статическое или нерабочее состояние системы;
- представление поведения, которое отражает динамическое или изменяющееся состояние системы;
- представление реализации, которое представляет структуру логических элементов системы;
- представление окружения среды, которое отражает распределение физических элементов системы.

9. Что представляет собой действующий субъект в ВИС?

Действующий субъект — это сущность-объект, взаимодействующий с системой для формирования события. Действующим субъектом может быть пользователь системы или сущность, например другая система или база данных, которая расположена за пределами системы.

10. Каково назначение СИС?

СИС детально описывают конкретный экземпляр ВИС. Для полного описания ВИС обычно требуется **создать** значительное число СИС.

11. Какова последовательность создания сценария использования системы?

Сценарий использования **создается** в следующей последовательности:

- определяются предварительные условия СИС, а также информация или условия для выполнения сценария;
- определяются постусловия СИС, то есть то, что требуется выполнить после завершения СИС;
- действие разбивается на **дискретные** шаги;
- определяются исключения, которые могут **возникнуть** на каждом шаге. Вам может понадобиться **разработать** СИС для этих исключений;
- определяются **требования**, к которым относится данный СИС, для последующего контроля в возможности **трассировки**.

Глава 3

Закрепление материала

1. Какова цель создания общей картины решения?

Общая цель создания общей картины решения заключается в выработке единого видения целей решения и **согласовании** его с заказчиком. Вот некоторые из целей, **преследуемых** при создании **общей** картины решения:

- определить **цели и ограничения** проекта и согласовать их с заказчиком;
- решить вопросы **реализуемости** решения, **заручиться** одобрением **участников** проекта;
- определить область действия проекта;
- оценить ресурсы, необходимые для выполнения проекта;
- определить и **спланировать** основные контрольные точки проекта.

2. Каковы обязанности различных ролей на этапе создания общей картины решения?

В модели команд MSP предусмотрено шесть ролей, обязанности которых перечислены ниже.

- Менеджер решения отвечает за корректную **реализацию** требований **заказчика**.
- Менеджер **программы** определяет цели проекта, критерии и метрики **успеха**, формулирует концепцию решения и **формирует** инфраструктуру проекта.
- Разработчик отвечает за обеспечение информации о технических следствиях разработки продукта и **реализуемости** решения.
- Специалист по удобству использования отвечает за анализ потребностей и проблем, **возникающих** у пользователей.
- **Тестировщик** отвечает за предоставлении информации о **достижении** целей по **качеству** решения и определяет действия, которые необходимы для достижения желаемого уровня качества
- Менеджер по выпуску отвечает за определение всего **необходимого** для развертывания продукта.

3. Каковы результаты этапа создания общей картины решения?

- Создается документ общей картины и области действия решения, который описывает **цели и ограничения** проекта.
- Создается документ структуры проекта, который описывает организационную структуру проекта и процесс руководства проектом.
- Создается документ оценки риска, который содержит начальное определение и анализ рисков, связанных с проектом, а также планы **мероприятий** по обеспечению непрерывности бизнеса.

4. Зачем определяют область действия проекта?

Область действия проекта определяет, что входит и что не входит в рамки проекта. Необходимо определить **функции**, которые **заказчик** считает обязательными для решения, и сосредоточиться на их **реализации**. Для этого требуется проанализировать все параметры проекта и определить все компромиссы. Одно из **очевидных** преимуществ определения области действия проекта заключается в том, что **команда** может сосредоточиться на первоочередных задачах и

не тратить время и **усилия** на необязательные функции. Определение области действия проекта также позволяет команде четко определить **все задачи проекта**.

5. Чем завершается этап **создания** общей картины решения?

Стадия создания общей картины решения считается завершенной при **достижении контрольной точки** «Утверждение документа **общей картины** *a* области действия проекта». Эта контрольная точка предусматривает совещание с участием ключевых участников, клиентов и представителей каждой роли в команде, на котором окончательно одобряется документ **общей картины** и области действия решения. Этот документ является **формальным соглашением** в отношении целей и задач проекта и решения, а также подтверждают согласие всех **сторон** на продолжение работы над проектом.

6. Какие решения относительно управления изменениями регистрируются в документе структуры проекта?

В разделе «Управление **изменениями**» документа структуры проекта **описываются процессы**, которые команда должна соблюдать при управлении изменениями в проекте, Вот некоторые из вопросов, на которые отвечает этот раздел.

- Как определяется изменение в проекте?
- Какова дата выпуска? Кто и как ее определяет? Что при этом учитывается?
- Кто устанавливает порядок управления **изменениями**?
- Как планируется **определять** и отслеживать изменения? Кто будет отслеживать эту информацию?
- Как **оценивать** влияние изменения? Каков порог числа отклонений, после которого необходима серьезная перепланировка?

7. В чем различие между **бизнес-целями** и проектными целями?

Бизнес-цели определяют задачи, которые клиент планирует решить, внедрив новый продукт. Проектные **цели** представляют атрибуты решения, которые проектная команда планирует **развивать**. Проектные цели необходимы **команде** для иллюстрации целей, которые **планируется** достичь за счет реализации решения.

8. Перечислите основные рекомендации по созданию пользовательских профилей.

В процессе создания **пользовательских** профилей необходимо:

- определить какие цели преследуют и что ожидают от решения конечные пользователи;
- определить факторы, которые влияют на способность пользователей работать с **решением**;
- определить особенности поддержки и **выяснить**, с какими проблемами пользователи **столкнулись** при работе с аналогичными **решениями**;
- выяснить, будет ли решение поддерживать другие языки и необходима ли локализация;
- описать **местоположение** пользователей, в том числе **географическое** и **физическое**, а также число пользователей в каждом **местоположении**, пропускную **способность** и **загрузку линий** связи между отдельными местоположениями;
- описать **информационные** потоки между пользователями, в том числе тип связи, важность и объем данных, которыми обмениваются различные группы пользователей;
- **проанализировать** иерархическую структуру **организации** и ее **влияние** на обмен информацией между различными уровнями **иерархии**;
- проанализировать область действия пользовательских **функций**;
- проанализировать **политики** принятия решений, которые непосредственно влияют на эффективную работу создаваемого продукта.

9. Перечислите обязательные компоненты документа **оценки** риска.

Документ **оценки** риска должен содержать **следующие разделы**:

- определение рисков — **описание** отдельных видов риска;
- вероятность риска — вероятность реализации **рискового события**;
- **серьезность** риска — **разрушительность** возможных последствий;
- подверженность риску — уровень подверженности риску;
- профилактические мероприятия — усилия по **предотвращению** или снижению риска;
- мероприятия по обеспечению непрерывности бизнеса — порядок и последовательность операций, выполняемых в случае **наступления** **рискового события**;

- ответственный за риск — член команды, который несет ответственность за регулярный **контроль** риска.
10. Как оценивать риски проекта?
Необходимо применять **единую** шкалу оценки вероятности и последствий **всех** рисков. Если определить влияние **одних** рисков **посредством** безразмерного числа, а других — **в** денежном выражении, их не удастся сравнить. Кроме того, вы должны оценить важность рисков в соответствии с их **влиянием**. Этот список **необходимо** регулярно обновлять на протяжении проекта, уточняя серьезность различных типов риска.

Глава 4

Закрепление материала

1. Какова **цель** этапа планирования?
 Цель этапа **планирования** — **достичь** подробного и согласованного **понимания** задачи, которую необходимо решить, и определить решающий эту задачу продукт. Такой продукт должен **удовлетворять** всем предположениям и ограничениям в соответствии с тем, как они поняты командой.
2. Чем отличаются обязанности ролей «менеджер решения» и «менеджер программы*» на этапе планирования?
 На этапе планирования менеджер решения **представляет** пользователей и отвечает за то, чтобы продукт удовлетворял их **потребности**. Менеджер программы заботится о том, чтобы решение создавалось в соответствии со **спецификациями, определенными менеджером решения**.
3. Каковы основные результаты этапа планирования?
 Основные результаты этапа **планирования** таковы:
 - функциональная **спецификация**, где описано, что и как планируется создать;
 - генеральный план проекта — набор планов для каждой из шести ролей команды;
 - генеральный календарный график проекта определяет временные рамки генерального плана проекта и синхронизирует календарные графики, разработанные разными командами;
 - **обновленный** генеральный документ оценки риска, в котором описываются **риски**, связанные с разработкой решения, и мероприятия по **предотвращению** наступления рискованных событий.
4. Каким образом функциональные спецификации **становятся** «черновым» планом для команды разработки?
 Функциональная спецификация описывает, что именно будет представлять из себя продукт. Она позволяет проектной команде создать **стратегию разработки** и оценить время и усилия, необходимые для разработки **решения**.
5. Каковы цели создания функциональных **спецификаций**?
 Цели **функциональной** спецификации таковы:
 - обеспечить **единое** понимание требований бизнеса и пользователей;
 - разбить задачу на более мелкие части и представить решение в виде логических модулей;
 - обеспечить среду для **планирования**, составления **календарных** графиков и построения решения;
 - зафиксировать соглашение между командой и заказчиком, **в** котором четко оговариваются все особенности продукта.
6. Какие виды риска возникают при отказе от создания функциональных **спецификаций**?
Некоторые виды возможных **рисков, возникающие** в связи с отказом от создания функциональной **спецификации**, таковы:
 - разработанное решение не полностью удовлетворяет требования заказчика;
 - менеджеру проекта не удастся точно рассчитать бюджет и график проекта. **Информация, содержащаяся** в функциональных **спецификациях**, помогает команде оценить усилия и набор **навыков**, необходимых для разработки решения;
 - члены команды не до конца понимают, чего ждет заказчик, и представления заказчика и подрядчика о **продукте** будут **различаться**. **Вследствие** этого теряется уверенность в том, что разрабатываемое решение — как раз то, что нужно **заказчику**;
 - информации, которой располагает команда, недостаточно, чтобы **проконтролировать** качество решения и его соответствие ожиданиям заказчика.

7. Каковы различия между требованиями бизнеса и пользователей решения?

Бизнес-требования к решению определяются клиентом и ориентированы на решение задач и целей бизнеса.

Пользовательские требования поступают от конечных пользователей решения и фокусируются на применении решения для эффективного выполнения ежедневных задач.

8. Каковы преимущества концептуального дизайна?

Концептуальный дизайн позволяет:

- разработать часть функциональной спецификации;
- разработать эффективный пользовательский интерфейс;
- определить, как взаимодействуют различные компоненты решения;
- спроектировать решение, которое удовлетворяет потребности бизнеса и пользователей,

9. Чтобы увеличить продажи на 15% в следующем финансовом году, компания планирует создать Интернет-магазин, для чего предполагается разработать высокоскоростной сайт, обеспечивающий безопасную обработку кредитных карт пользователей и доступный 24 часа в сутки, 7 дней в неделю. Возможность покупать продукты через сайт будет предоставляться только зарегистрированным пользователям. Каковы пользовательские, системные, процедурные и бизнес-требования к этому сайту?

Требования для описанного сайта таковы:

- **бизнес-требование:** Web-сайт должен обеспечить 15-процентный рост продаж в годовом исчислении;
- **пользовательские требования:**
 - посетитель не должен тратить на покупку более 5 минут;
 - посетителю должен предоставляться доступ только к данным его личной кредитной карточки;
- **системное требование:** совершать покупки на сайте разрешается только зарегистрированным пользователям;

I операционное требование: необходимо обеспечить постоянную доступность сайта для посетителей.

10. Каковы цели стадии анализа в процессе концептуального дизайна?

Стадия анализа необходима для:

- анализа пользовательских и бизнес-процессов и операций;
- документирования и моделирования контекстов, потоков работ, последовательностей задач и бизнес-среды.

11. Какие преимущества дает синтез информации?

Синтез позволяет представить информацию всей команде и узнать мнение всех членов команды о требованиях и решениях пользователей. Это позволяет гарантировать единство понимания членами команды пользовательских требований и ожиданий.

12. Какие задачи необходимо выполнить в процессе создания проекта будущего решения?

При создании проекта будущего решения команда выполняет следующие задачи:

- определяет общую картину будущего решения;
- осуществляет перепроектировку текущих процессов для обеспечения оптимальной поддержки ключевых бизнес-операций и процессов;
- создает точные целевые сценарии использования системы;
- проверяет сценарии состояния путем их пошагового прогона.

13. Какие преимущества дает проверка концептуального дизайна?

Проверка концептуального дизайна:

- снижает риск;
- позволяет обнаружить пропущенную информацию;
- помогает обнаружить различное видение и интерпретацию решения, особенно руководством и пользователями;
- подтверждает объемы работ;
- помогает определить приоритеты;
- обеспечивает базу для создания логического дизайна.

14. Перечислите четыре категории сервисов.
- Пользовательские сервисы представляют собой модули прикладной логики, обеспечивающие работу пользовательского интерфейса.
 - Бизнес-сервисы — это модули прикладной логики, которые обеспечивают выполнение бизнес-правил в нужной последовательности.
 - Сервисы данных — это модули прикладной логики, обеспечивающие самый низкий уровень детализации при работе с данными.
 - Системные сервисы — это модули логики приложения, обеспечивающие функции, лежащие вне бизнес-логики.
15. Изучите и распределите по категориям следующие сервисы:
- отображения подробной информации о сотруднике;
 - обновления информации о сотруднике;
 - предоставления данных о сотруднике;
 - электронную почту.
- Сервисы классифицируются так:
- отображение подробной информации о сотруднике — пользовательский сервис;
 - обновление информации о сотруднике — бизнес-сервис;
 - предоставления данных о сотруднике — сервис данных;
 - электронная почта — системный сервис.
16. Каковы особенности уточненных требований бизнеса?
- Бизнес-требования должны быть: четко определенными, краткими, поддающимися проверке, выраженными на языке бизнеса без использования жаргона; кроме того, они должны быть организованы в иерархию зависимых требований.
17. Каким образом уточняются варианты использования системы на стадии анализа в процессе концептуального дизайна?
- В процессе уточнения диаграмм ВИС выполняются следующие задачи:
- создаются подчиненные ВИС;
 - создаются СИС для каждого подчиненного ВИС;
 - выполняется проверка всех ВИС И СИС на основании исходных интервью, прочей документации, а также с привлечением пользователей.
18. Каковы критерии оценки стоимости решения?
- Критерии таковы:
- ресурсы, трудозатраты и время;
 - расходы на эксплуатацию технологий и технических новинок;
 - ежегодные и периодические издержки за срок службы решения.

Глава 5

Практикум

Упражнение 1

1. При заполнении корзины покупателя проверяется наличие определенного товара на складе.
Объекты: «корзина покупателя», «товарные запасы» и «товар».
2. Новый пользователь создает учетную запись, чтобы размещать заказы.
Объекты: «пользователь», «чеет» и «заказы».
3. Пользователь добавляет товар в корзину.
Объекты: «пользователь», «товар» и «корзина покупателя».
4. Пользователь находит и открывает свой заказ для проверки его состояния.
Объекты: «пользователь» и «заказ».

Закрепление материала

1. Из каких стадий состоит логический дизайн?
Логический дизайн состоит из:
 - анализа, во время которого команда определяет объекты, сервисы, атрибуты и отношения;

- оптимизация, в течение которой уточняется **список** объектов, проверяется **проект** и реализуется контроль.
2. Каковы результаты логического дизайна?
- К результатам логического дизайна относятся:
- логическая модель объектов;
 - **высокоуровневый дизайн** пользовательского интерфейса;
 - логическая модель данных.
3. Следует ли на этапе логического дизайна уделять особое внимание технологиям?
- Хотя логический дизайн **считается** независимым от технологий, в его **процессе** рекомендуется рассмотреть физические ограничения и **возможности**, чтобы **убедиться** в возможности реализации проекта. Опытные команды и **проектировщики** обычно используют **стадию** оптимизация логического дизайна для выяснения **технологических** ограничений и создания **псевдофизического** дизайна. **Логический дизайн** — это основание для **оценки** возможности **реализации** физического дизайна и определения альтернативных реализаций.
4. Каковы преимущества логического дизайна?
- Логический дизайн:
- помогает эффективно справляться со сложностью проекта;
 - отражает и поддерживает **требования** концептуального дизайна;
 - дает **ясное** представление о проекте в целом;
 - позволяет найти точки соприкосновения множества разнообразных **подразделений** и их систем;
 - служит отправной точкой для физического дизайна.
5. Как в СИС определяются сервисы?
- Чтобы определить **сервисы** объекта следует вернуться к сценарию использования **системы** и выяснить, что должен делать **объект**, какие **типы** данных **поддерживать** и какие **операции** выполнять. Объект, **работающий с информацией**, также выполняет с ней различные операции.
6. Как в СИС определяются атрибуты?
- Чтобы **определить** атрибуты объекта, вернитесь к сценариям использования **системы**: **поищите** слова или фразы, **содержащие** дополнительные определения объекта
7. Что такое диаграмма последовательности?
- Диаграммы **последовательности** показывают **действующие** лица и объекты, **участвующие** во взаимодействии, вместе с инициируемыми ими событиями в хронологическом порядке. Она также отображает зависимости объектов в системе.
8. Как проектируются таблицы и столбцы в хранилище данных создаваемой системы?
- Для каждого объекта, **содержащего** данные, создается отдельная **таблица**. **Атрибуты** объекта становятся **столбцами** таблицы, связанной с объектом. В **каждой** строке таблицы **хранятся** поля одного экземпляра **объекта**.
9. Зачем уточняется список объектов?
- Уточняя набор объектов, **принимают** во **внимание** ряд **соображений**:
- если два **объекта** предоставляют одинаковую информацию или управляют **одним** действием, их следует объединить в единый объект;
 - существительные в СИС могут быть ключевыми атрибутами других объектов;
 - иногда приходится создать новый объект(ы) для управления и координации **набора сервисов**.
10. Как выполняется индивидуальная проверка объектов?
- В ходе **этой** проверки определяются входные и выходные **данные** объекта, а также возможности и набор **функций**, который он должен **предоставлять**. Выходные данные и **поведение** необходимо **точно** прогнозировать для любых входных данных. Также следует **проверить** независимые части объекта.
- И. Какова цель управления в логическом дизайне?
- Управление в логическом дизайне:
- обеспечивает **транзактную** целостность **сценария**;
 - координирует **сервисы** между объектами;
 - определяет взаимозависимости объектов.

12. Работая на логическим дизайном системы, вы обнаружили не описанный ранее сценарий. Что следует предпринять?
Прежде всего вы должны пересмотреть всю текущую документацию и убедиться, что новая информация действительно не учтена в анализе. Если это действительно так, выясните влияние, которое окажет включение новых функций на текущую версию решения, а также эффект от исключения этих функций из текущей версии. Получив эту информацию, необходимо уведомить клиента об изменении ситуации и совместно решить, как действовать далее на основании новой информации. Имейте в виду, что область действия проекта может измениться, так как она часто зависит от того, как вы поступите с новой информацией.
13. Каковы обязанности роли тестировщика в процессе логического дизайна?
В процессе логического дизайна тестировщик отвечает за проверку соответствия моделей логического дизайна концептуальному дизайну и удовлетворение им всех требований, а также за создание высокоуровневого плана тестирования.

Глава 6

Закрепление материала

1. Каковы цели физического дизайна?
На этапе физического дизайна решаются следующие задачи:
- преобразование логического дизайна в спецификации набора компонентов;
 - создание базовой версии, основы для начала разработки;
 - выбор наиболее подходящих технологий;
 - создание структурного представления решения с точки зрения проектной команды.
2. Чем отличаются концептуальный, логический и физический дизайн?
В процессе концептуального дизайна проектная команда описывает решение с точки зрения бизнеса и конечных пользователей. При логическом дизайне решение описывается с точки зрения проектной команды. Во время физического дизайна проблема рассматривается с точки зрения группы разработчиков.
3. Чем занимается группа разработчиков (программистов) в процессе физического дизайна?
В течение физического дизайна группа разработчиков оценивает технологии, создает прототипы и готовится к разработке.
4. Что включает в себя модель развертывания?
Модель развертывания состоит из:
- топологии сети, отражающей расположение оборудования и его подключение;
 - топологии данных, которая показывает местоположение данных в топологии сети;
 - топологии компонентов, которая отражает расположение компонентов и их сервисов в привязке к топологии сети.
5. Чем занимается проектная команда на стадии исследования?
На стадии исследования проектная команда выясняет ограничения — архитектуру, бизнес-процессы и инфраструктуру предприятия, а также определяет требования по производительности, доступности и масштабируемости. Кроме того, команда выполняет анализ различий между требованиями и ограничениями.
6. Как проектная команда сокращает разрыв между требованиями и ограничениями?
Разрыв между требованиями и ограничениями обычно разрешают так.
- Выявляют места инфраструктуры, где требования противоречат ограничениям.
 - Анализируют разрыв между требованиями и ограничениями и определяют, следует ли принимать какие-либо решения для разрешения конфликта.
 - Определяют жизненно важные для проекта требования.
 - Проводят мозговой штурм при участии всех групп, связанных с проектом.
7. Как на стадии анализа проектная команда использует список объектов и сервисов, созданных в процессе логического дизайна?
Проектная команда использует перечень объектов и сервисов для:
- распределения сервисов по категориям: пользовательские сервисы, бизнес-сервисы, сервисы данных и системные сервисы;

- выявления сервисов, оставшихся незамеченными.
8. Как проектная команда уточняет диаграмму классов на стадии анализа?
В процессе физического дизайна команда предпринимает следующие действия по уточнению и совершенствованию диаграмм классов:
- определяет объекты, не выявленные в процессе логического дизайна;
 - консолидируют логические объекты (при необходимости);
 - уточняет методы, учитывая параметры;
 - уточняет атрибуты.
9. Как вы выбираете технологии-кандидаты для решения?
При оценке и отборе технологий, которые предполагается применить в будущем решении, учитываются особенности бизнеса, корпоративной архитектуры и технологий.
10. В чем различие между топологией сети и топологией данных модели развертывания?
Топология сети — это карта инфраструктуры, которая указывает местоположение оборудования и подключений, а топология данных определяет положение хранилищ данных в топологии сети.
- П. В чем различие между стратегиями распределения и объединения?
Стратегия распределения — это обоснование выбранного расположения отдельных сервисов в архитектуре решения. Распределение выполняется на основе сервисов, а не компонентов.
Стратегия объединения в наборы — это обоснование распределения сервисов по компонентам. В рамках одного решения может быть несколько стратегий.
12. В чем различие между связностью и сочленением?
Связность — это степень связи между различными внутренними элементами компонента, а сочленение — между разными компонентами. Компонент должен обладать высокой степенью связности, но допускать свободное сочленение.
13. Опишите назначение модели программирования.
Модель программирования:
- описывает, как проектная команда может использовать отобранные технологии;
 - определяет стандарты, которые должны соблюдаться в процессе реализации проекта;
 - определяет конкретные правила, обеспечивающие согласованную реализацию компонентов.
14. Что такое интерфейс компонента?
Интерфейс компонента:
- представляет собой соглашение о взаимодействии между компонентом-поставщиком и компонентом-потребителем;
 - является средством доступа к поддерживаемому им сервису;
 - » предоставляет один или несколько сервисов;
 - поддерживает основные атрибуты объекта.
15. Какие существуют типы пользователей на уровне пользовательских сервисов приложения?
На уровне пользовательских сервисов возможны пользователи двух типов: люди и компьютерные системы. Пользовательский интерфейс предоставляет удобные для человека визуальные средства взаимодействия. Пользовательские сервисы предоставляют возможности навигации, проверки и логику обработки ошибок.

Глава 7

Закрепление материала

1. Какова роль презентационного уровня в архитектуре бизнес-приложения?
Презентационный уровень — это часть бизнес-приложения, на котором происходит взаимодействие пользователя с уровнем бизнес-функций системы. Презентационный уровень содержит компоненты, которые обеспечивают взаимодействие пользователей с приложением. В простейших реализациях презентационного уровня используются элементы пользовательского интерфейса из состава графического пользовательского интерфейса (GUI), встроенного в ОС, например Microsoft Windows Forms и Microsoft ASP.NET Web Forms. Для организации более сложного диалога с пользователем создают компоненты пользовательского процесса.

2. Каковы отличительные черты удачного пользовательского интерфейса?
 Ниже перечислены **характеристики** удачно спроектированного и эффективного интерфейса,
- **интуитивно понятный дизайн;**
 - продуманное использование площади экрана;
 - **простота навигации;**
 - **управляемая навигация;**
 - проверка **входных** данных на корректность;
 - меню, панели **инструментов** и **справочная система;**
 - эффективная обработка **событий/**
3. Чем отличается дизайн с высокой и низкой детализацией?
 • Дизайн с низкой детализацией содержит основную структуру и базовые **функции** интерфейса, а также возможные пути **навигации**, а дизайн с высокой **детализацией** содержит подробную информацию о **компоновке** экрана и элементах интерфейса.
 • Дизайн с низкой **детализацией** позволяет быстро и эффективно оценить альтернативные варианты дизайна. С другой стороны, хотя дизайн с высокой **детализацией** и быстр в реализации и **изменении**, но для него **обязательны** компьютер и программное **обеспечение**.
 • **Дизайн с низкой детализацией** полезен для «**мозговых штурмов**» и получения обратного отклика. Обычно дизайн с высокой детализацией создается на **основе** дизайна с низкой детализацией.
4. Какие средства доступны разработчикам для организации помощи пользователям?
 Под системой помощи пользователям обычно подразумевают следующие **составляющие**:
- интерактивная справочная система;
 - **всплывающие подсказки;**
 - мастера;
 - строки **состояния;**
 - **специальные возможности.**
5. Какие существуют типы моделей пользовательского приложения и в каких случаях их стоит применять?
 • Стандартный интерфейс Windows применяется, когда **необходимо** обеспечить работу пользователей в автономном режиме и когда необходима богатая функциональность системы, а также при **интеграции** с пользовательскими интерфейсами **других** приложений.
 • **Web-интерфейс** пользователя позволяет **обеспечить** поддержку стандартных пользовательских интерфейсов на многих устройствах и платформах. Поэтому **Web-интерфейс** рекомендуется применять, когда **приложение** будет работать на самых **разнообразных** клиентских устройствах.
 • **Интерфейс** для мобильных устройств должен уместиться на небольшой площади экрана много информации и оставаться при этом достаточно удобным в работе.
 • **Пользовательский интерфейс** на основе документов применяется в некоторых **приложениях** и позволяет пользователям входить в систему или просматривать данные в форме документов **стандартных** инструментальных средств, предназначенных для повседневной работы.
6. В чем **разница** между компонентами **пользовательского** интерфейса и пользовательского процесса. Приведите пример, когда применение компонентов пользовательского процесса предпочтительнее.
Компоненты пользовательского **интерфейса** составляют пользовательский интерфейс, а компоненты **пользовательского** процесса инкапсулируют процессы взаимодействия пользователя с приложением. Пример компонента пользовательского процесса — вход пользователя в систему. Пользователь может выполнить эту **операцию** посредством Web-браузера или Windows-приложения. Интерфейсы отличаются, но процесс входа идентичен. Абстрагирование процесса входа в систему от интерфейса способствует повторному использованию и **единообразию** приложения.
7. Как отделяют интерфейс от пользовательского процесса?
 1. Выясните, какие бизнес-процессы будет поддерживать **пользовательский** процесс. Определите, как это **выглядит** с точки зрения пользователя. (Обычно для этого сверяются с диаграммами последовательностей, **вариантами** и сценариями использования системы, созданными в процессе анализа **требований**.)

2. Определите, **какие данные необходимы для бизнес-процесса**. Пользовательский процесс должен «уметь» при необходимости **обращаться** к этим данным.
3. Определите, какую **дополнительную информацию** о состоянии следует **поддерживать** во время сеанса **пользователя** для **нормальной обработки** и получения данных в **пользовательском интерфейсе**.
4. Создайте наглядные схемы потоков пользовательского процесса и способы **получения** и **передачи** потоков управления каждым элементом пользовательского **интерфейса**.
8. Вы проектируете систему, **осуществляющую запросы** с использованием службы терминалов Windows. Какой тип пользовательского интерфейса следует выбрать?
На удаленном компьютере следует предусмотреть приложение с Windows-интерфейсом. Для **организации** доступа к удаленному компьютеру с локальных компьютеров в качестве гонкого клиента достаточно использовать удаленный рабочий стол (Remote Desktop).
9. На этапах **создания** общей картины решения и планирования выяснилось, что пользователи **будущего** решения **работают** на различном оборудовании в разных странах **земного шара** и в большинстве своем не имеют доступа к корпоративной **интрасети**. Какой тип клиента предпочтительнее в такой ситуации?
Более всего подходит Web-клиент. Поскольку конфигурация у **различных** пользователей может сильно **отличаться**, а у **многих** нет доступа к интрасети, то наиболее удачно **решение** на основе безопасного Web-доступа.
10. Вы закончили проектирование пользовательского интерфейса. Как проверить дизайн до его реализации?
Проект пользовательского интерфейса создается на основе информации, полученной в процессе интервьюирования пользователей, определенных и **одобренных** заказчиком требований, а также множества ВИС и диаграмм действий, созданных в процессе **концептуального** и логического дизайна. Готовый дизайн пользовательского интерфейса сравнивается с **информацией** из каждого источника — так проверяется соответствие пользовательского **интерфейса** всем требованиям к нему.

Глава 8

Закрепление материала

1. Каким образом выполняется проектирование **модели** данных на этапе планирования?
В течение концептуального **дизайна** проектная команда **исследует** и анализирует **требования** к данным, пытаясь **выяснить**, какую информацию должно хранить и обрабатывать **бизнес-решение**. При разработке логического проекта команда определяет набор объектов **данных**, создаваемый на **основании** требований к данным. На этой стадии определяются **отношения** между различными объектами и создается **схема** базы данных.
2. Каково назначение схемы базы данных?
Схема базы данных определяет **организацию** данных в базе. На стадии логического **дизайна** **члены проектной** команды создают схему базы данных, что позволяет сначала **определить, что** требуется построить, и лишь затем — **как** это сделать.
3. Каково назначение атрибутов?
Атрибуты:
 - описывают объект;
 - закрепляются за объектом, который они описывают наиболее точно;
 - являются **столбцами** в базе данных.
4. Зачем определяют типы в базе данных?
Типы данных в базе:
 - определяют порядок хранения **информации** в БД;
 - определяют формат данных;
 - позволяют **реализовать** проверку корректности данных;
 - корректный выбор типов данных позволяет оптимизировать **хранение** данных.
5. Каким образом в большинстве СУБД поддерживаются связи вида «многие-ко-многим»?
В большинстве СУБД отношения вида «многие-ко-многим» поддерживаются за счет **создания** **таблицы соединения**. Таким образом хранится информация об отношениях между объектами.

6. Как оптимизировать транзакции, чтобы повысить производительности системы?

Транзакции должны быть как можно **короче**, и **применять** их следует **только** там, где они действительно необходимы. Кроме того, следует избегать **создания распределенных транзакций**.

7. Какое влияние оказывает индексирование на доступ к данным?

Запросы **индексированных** данных выполняются **намного** быстрее и **эффективнее**. Не требуется просмотра всей таблицы в поиске нужных данных — индекс (а это, по сути, карта данных) СУБД позволяет быстро направить **запрос** прямо к нужной информации.

8. Чем отличаются горизонтальное и вертикальное разбиение данных на разделы?

В **случае горизонтального** разбиения таблица с **большим количеством** записей «режется» на несколько таблиц с одинаковым набором столбцов. При вертикальном разбиении таблицы с большим количеством столбцов распределяется по нескольким разделам (**подтаблицам**) со строками, **обладающими одинаковыми уникальными** идентификаторами.

9. Каковы преимущества нормализации?

- **Сокращение** дублирования информации.
- Сокращение числа случаев нарушения целостности данных.
- **Сокращение** числа пустых полей (оптимизация хранения).

10. Что такое денормализация?

Денормализация — это **обратный** по отношению к нормализации процесс, цель которого — создание таблиц с большим количеством **полей**, чтобы сократить количество соединений в запросах.

- П. Какие три типа целостности данных обычно применяются в БД?

Существует три типа целостности данных:

- предметная целостность — определение набора **разрешенных** значений данных для столбца и допустимость значения **NULL**;
- целостность **объекта-сущности** требует, чтобы каждая строка в таблице имела уникальный идентификатор в поле основного ключа;
- **ссылочная целостность** обеспечивает поддержку постоянной **связи** между основными (в таблице родительских объектов-сущностей) и внешними ключами (в таблице дочерних **объектов-сущностей**).

12. Как определяют требования к целостности данных?

В процессе определения требований к **целостности данных**:

- требования к данным **проверяют** на предмет уникальности диапазонов значений и **ограничений**, которые позволяют гарантировать **существование** и корректную **реализацию** объектов-сущностей;
- после **определения** ограничений и диапазонов выясняют, привязаны ли они к объекту или отношению между объектами;
- обеспечивают **ссылочную целостность**, что позволяет гарантировать **поддержку** всех **отношений**.

13. Каковы критерии определения бизнес-правил?

Определяя **бизнес-правила**, учитывайте ряд критериев.

- условия, которым **должны** удовлетворять данные, чтобы они считались корректными;
- условия, которых необходимо **избегать**;
- порядок событий.

14. Каким образом ключи применяются для реализации **ссылочной** целостности?

Определенные основными и внешними **ключами** отношения физической модели напрямую соответствуют **ключевым** параметрам ядра базы данных. Эти параметры автоматически обеспечивают **ссылочную целостность** связанных таблиц.

15. Каковы преимущества компонентов при реализации **бизнес-правил**?

Основные преимущества таковы:

- код хранится в одном или нескольких местах и его легко обновлять;
- масштабируемость за счет добавления дополнительных прикладных серверов и распределения возросшей вычислительной нагрузки.

Глава 9

Закрепление материала

- Каковы недостатки традиционных моделей безопасности?
Недостатки традиционных моделей безопасности таковы:
 - они основаны на идентификаторе пользователя, а не кода;
 - они сильно подвержены атакам вирусов и червей.
- Каковы принципы создания защищенного кода?
Чтобы спроектировать защищенное приложение, вы должны знать следующие принципы безопасности и применять их при создании стратегии защиты.
 - Полагайтесь только на проверенные системы безопасности.
 - Никогда не доверяйте данным, полученным извне.
 - Считайте все внешние системы незащищенными.
 - Придерживайтесь принципа наименьших привилегий.
 - Минимизируйте количество доступных компонентов и данных.
 - Используйте безопасные параметры по умолчанию.
 - Не полагайтесь на защиту, основанную на незнании или сокрытии определенной информации.
 - Устраняйте дыры в безопасности системы, классифицируя уязвимые места системы по схеме STRIDE.
- Из перечисленных ниже высказываний выберите те, что применимы к переполнению буфера:
 - контроль типов предназначен для предотвращения переполнения буфера;
 - из-за переполнения буфера приложение зачастую перестает отвечать на запросы или неправильно работает;
 - злоумышленники могут использовать переполнение буфера для запуска произвольного кода;
 - сообщение об ошибке из-за переполнения буфера может представлять угрозу безопасности приложения.**Все перечисленные высказывания в той или иной мере применимы к ситуации переполнения буфера.**
- На каком из этапов MSF необходимо создавать модель опасностей:
 - при планировании;
 - при разработке;
 - при стабилизации?**Модель опасностей создается на этапе планирования.**
- Что такое модель STRIDE?
STRIDE — это метод, применяемый для определения и классификации опасностей, грозящих приложению. Каждая буква в сокращении STRIDE обозначает определенную категорию опасности: подмена сетевых объектов (spoofing identity), модификация данных (tampering), отказ от причастности (repudiation), раскрытие информации (disclosure), отказ в обслуживании (denial of service) и повышение привилегий (elevation of privilege).
- Ознакомьтесь с приведенным сценарием атаки и определите, по каким категориям STRIDE следует классифицировать возникающие в данной ситуации опасности.
«Карл заметил, что Боб оставил свой компьютер без присмотра и не заблокировал его. Карл сел за компьютер Боба и открыл приложения для работы с электронной почтой. Он отправил Алисе письмо от имени Боба. Затем он закрыл почтовый клиент и покинул рабочее место, причем и его действия, и уход остались незамеченными».
В описанной ситуации возможны четыре вида атаки: подмена сетевых объектов, модификация данных, отказ от причастности и повышение привилегий.
- Что такое защита доступа из кода?
Безопасность доступа из кода, обеспечиваемая .NET Framework, защищает компьютерные системы от вредоносного или просто ошибочного кода и позволяет безопасно запускать мо-

бильный код. Безопасность доступа из кода позволяет регулировать уровень доверия коду в зависимости от его происхождения и признаков.

8. Что такое основанная на ролях защита?

Этот механизм защиты в первую очередь ориентирован на устранение угрозы подмены сетевых объектов путем запрещения не прошедшим авторизацию пользователям выполнять операции, для которых такая процедура обязательна. Она предусматривает проверку идентификационных данных и принадлежность пользователя к определенной роли. Помимо классов, предназначенных для реализации основанной на ролях защиты в других механизмах аутентификации, в .NET Framework есть классы для определения пользователей и групп Windows.

9. Какие провайдеры аутентификации поддерживает ASP.NET?

ASP.NET поддерживает три провайдера аутентификации: на основе форм, паспорта Microsoft (Microsoft Passport) и Windows-аутентификацию.

10. Какие три типа защиты обеспечивают Web-сервисы?

Защита Web-сервисов обеспечивается на трех уровнях:

- на уровне платформы, или на транспортном уровне («точка-точка»);
- на уровне приложения (пользовательский механизм аутентификации);
- на уровне сообщения (сквозная).

11. Перечислите стадии разработки стратегии авторизации и аутентификации в приложении.

Стадий разработки общей стратегии аутентификации и авторизации несколько.

1. Определение ресурсов.
2. Выбор стратегии авторизации.
3. Выбор объектов безопасности, используемых для контроля доступа к ресурсам.
4. Определение типа объектов безопасности — перемещаемые или нет.
5. Выбор метода аутентификации.
6. Определение механизмов перемещения объектов безопасности в системе.

Глава 10

Практикум

Упражнение 1

1. Изучите план тестирования и перечислите три способа применения ВИС и СИС в процессе тестирования решения для компании Adventure Works Cycles.

В процессе тестирования решения для компании Adventure Works Cycles ВИС и СИС следует применять для определения действий, подлежащих тестированию:

- при проверке пользовательских функций;
- при проверке административных функций;
- при создании сценариев тестирования.

2. Каким образом следует признавать и документировать ошибки с целью откорректировать и исправить приложение?

На основе каждой ошибки, удовлетворяющей критериям изменений, создаются запросы на изменение.

3. Какие учетные записи пользователей следует сконфигурировать на тестовых серверах?

На тестовых серверах определяются следующие учетные записи:

- сетевой администратор;
- торговый представитель 1;
- торговый представитель 2;
- менеджер по продажам;
- Web-клиент 1;
- Web-клиент 2;
- торговый посредник.

4. Почему описанную далее проблему следует указать в списке опасностей для тестирования?

Необходимость участия менеджеров по продажам в тестировании. Команду тестирования яолжен контролировать по крайней мере один торговый представитель. Решение: заранее

договориться Б вице-президентом по продажам о том, что в тестировании будут участвовать два торговых представителя.

Поскольку большинство функций определено в СИС, в которых участвуют торговые представители, подобное требование значительно повысит надежность результатов тестирования. Кроме того, решение будет одобрено сотрудниками отдела продаж (при условии, что два отобранных представителя увидят, что их вклад оказался полезным и инициировал конкретные запросы на изменение). Дополнительное преимущество заключается в том, что двое сотрудников заказчика научатся быстро и эффективно внедрять приложение и смогут обучать работе с ним других сотрудников.

Упражнение 2

1. Каковы параметры метода `addOrderDetail`?

К параметрам метода `addOrderDetail` относятся: `ProductID` и `Quantity`.

2. Будет ли в приложении два типа пользовательского интерфейса?

Да, на основе Windows Forms и Web Forms.

3. Почему раздел «Интерфейсы» все еще не определен?

Интерфейсы документируются только после их разработки и проверки. Псевдокод не должен оставаться в технической спецификации — после завершения разработки его необходимо заменить актуальным кодом.

Закрепление материала

1. Каким образом выполняется вертикальное масштабирование приложения?

При вертикальном масштабировании увеличивают объем памяти, устанавливают дополнительные или более быстродействующие процессоры или просто переносят приложение на другую, более мощную машину.

2. Что необходимо учитывать при проектировании масштабируемого решения?

Для обеспечения масштабируемости необходимо следовать нескольким рекомендациям.

- процессы не должны ожидать отклика дольше, чем необходимо;
- процессы не должны «бороться» за ресурсы;
- следует добиваться коммутативности процессов;
- следует создавать взаимозаменяемые компоненты;
- необходимо разделять ресурсы и сервисы (службы).

3. Что необходимо учитывать при проектировании решения, отличающегося высокой доступностью?

Доступность — это упреждение, выявление и разрешение программных и аппаратных неполадок до того, как они приведут к сбою в работе сервисов или повреждению данных.

4. Как кластеризация повышает доступность приложения?

Кластер — это несколько компьютеров, соединенных сетью и логически объединенных кластерным ПО. При сбое активного сервера, его нагрузка автоматически переносится на пассивный сервер, текущие клиентские процессы переключаются, а потерпевший сбой сервис автоматически перезапускается. Даже в случае недоступности ресурса пользователя, подключенные к серверному кластеру, испытывают определенную задержку, однако обслуживание не прекращается.

5. Каким образом сократить время планового простоя приложения?

Один из лучших способов сокращения времени планового простоя — «горячее» обновление, при котором перед обновлением компонента кластерного сервера группа серверных ресурсов переносится на другой сервер, обновляемый сервер отключается, выполняется обновление компонента, а затем сервер снова включают в сеть. В это время нагрузка распределяется между другими серверами, и работа приложения не прерывается.

6. Почему надежность — одна из важнейших составляющих проекта приложения?

- Ненадежные системы обходятся очень дорого. Пользователи не жалуют ненадежный Web-сайт, что немедленно отрицательно сказывается на его доходности и объеме будущих продаж.

- Большие затраты на восстановление поврежденных данных лишь усугубляют убытки, возникающие из-за сбоев приложения.
 - **Ненадежные** системы трудно **сопровождать** и развивать, поскольку **точки сбоев**, как правило, трудно обнаружить.
7. Как обеспечить надежность в проектируемом приложении?
- Для этого **необходим** ряд **мероприятий**:
- отражение в **спецификации** требований к надежности;
 - **использование качественной** архитектурной инфраструктуры;
 - включение в приложение **информации**, необходимой для управления;
 - реализация избыточности и дублирования;
 - » применение **инструментов** контроля качества;
 - проверка **надежности** приложения;
 - реализация обработки ошибок;
 - создание архитектуры приложения, которая **предусматривает** в случае сбоя отключение лишь **некоторых функций**, а не крах всего приложения.
8. Как **определить** требования к **производительности**?
- Для создания качественных требований к **производительности** необходимо выяснить ограничения проекта, определить сервисы, которые должно предоставлять **приложение**, и оценить нагрузку на приложение.
9. Почему следует **проектировать** приложения, **эффективно** поддерживающие взаимодействие с другими системами?
- Приложение, **эффективно** поддерживающие взаимодействие с другими системами, позволяет:
- снизить **эксплуатационные** расходы и сложность продукта;
 - разворачивать систему оптимальным способом;
 - сохранить **существующие** инвестиции в ИТ.
10. Каким образом обеспечить в приложении возможность поддержки других языков?
- Для обеспечения возможности поддержки других языков необходимо:
- определить, **какие** языки и стандарты необходимо поддерживать;
 - разработать **дизайн функций** для различных языков и **стандартов**;
 - написать код, способный правильно поддерживать требуемые языки и стандарты.
11. Каково назначение плана мониторинга?
- В **плане** мониторинга определяется процесс мониторинга: что именно и каким образом предполагается контролировать и как **представлять** и использовать **результаты** мониторинга. Многие процедуры мониторинга в корпоративных приложениях выполняются **автоматически**.
12. Каково назначение раздела, **посвященного** стратегиям плана переноса данных?
- В разделе **стратегии** переноса указывают основные принципы переноса данных. Стратегии не всегда являются **взаимоисключающими**, но они описывают различные стороны общего процесса перехода. Стратегия может базироваться на системе версий (в зависимости от **зрелости бизнес-процессов**, разработки или технологии) или компонентов. Здесь **также** необходимо учесть особенности **перехода** от **предыдущих** систем к **новому** продукту. Если перенос касается **бизнес-объектов** и данных, в стратегии предусматривают несколько разделов, **посвященных** стратегиям переноса.
13. Почему необходимо создавать **спецификации** по лицензированию для двух этапов — разработки и развертывания?
- На этапе разработки команда применяет отобранные технологии и продукты. Необходимо позаботиться о наличии всех необходимых **лицензий** на **используемые** продукты.
- На этапе **развертывания** следует определить число **лицензий**, необходимых для всего используемого ПО. Оно зависит от **типа** решения и числа пользователей **решения**.
14. Каково назначение плана разработки?
- В **плане разработки** описывают процесс создания решения и задачи по конструированию и сборке компонентов продукта. В нем формулируются **четкие правила** и описываются процессы для **команд-разработчиков**. Наличие **документов** предполагает, что команде ясна структура к **направлению** разработки, у ее членов нет разногласий на этот счет, и это позволяет разработчикам сконцентрироваться на создании решения. Единые **принципы** и стандарты также облег-

чают повторное использование плодов труда различных групп и сводят к минимуму зависимость от отдельных сотрудников или групп.

15. Каково назначение плана тестирования?

В плане тестирования описываются стратегия и методы планирования, организации и управления тестированием проекта. В плане тестирования определены цели тестирования, методологии и инструменты, ожидаемые результаты, обязанности и требования к ресурсам. Это основной документ команды тестирования. Основная задача плана тестирования — обеспечить тщательно и качественно организованный процесс тестирования, а также определить используемые командой разработчиков критерии достаточной стабильности решения.

16. Кто отвечает за создание плана развертывания?

На роль «менеджер по выпуску» модели команд MSF возлагается ответственность за проектирование и реализацию плана развертывания решения, а также за определение инфраструктуры решения и обеспечение бесперебойной работы после развертывания продукта.

17. Что такое техническая спецификация?

Техническая спецификация — это набор справочных документов, обычно содержащих артефакты физического дизайна, такие, как спецификации классов, модели компонентов, метрики и топологию сети и компонентов. На этапе разработки разработчики используют техническую спецификацию для определения области действия создаваемого продукта и ожидаемых от него результатов.

Глава 11

Закрепление материала

1. Назовите некоторые особые цели этапа стабилизации.

К целям этапа стабилизации относятся:

- повышение качества решения;
- устранение существующих неполадок;
- перемещение фокуса от разработки к качеству;
- стабилизация решения;
- подготовка к выпуску продукта.

2. Как анализ сходимости ошибок сигнализирует о стабилизации проекта?

Неубывание разницы между вновь обнаруживаемыми и устраненными ошибкам свидетельствует об отсутствии прогресса в области стабильности решения. Это очень серьезная проблема; команда должна отнестись к ней соответствующим образом и принять срочные меры по ее решению.

3. Почему точка отсутствия ошибок считается положительной тенденцией проекта?

Получение версии без ошибок — четкий сигнал, что команда вышла на «финишную прямую» и близка к получению стабильного продукта.

4. Что такое кандидат на выпуск и каковы его особенности?

После достижения первой точки отсутствия ошибок готовится серия версий продукта — кандидатов на выпуск для передачи в группу пилотной эксплуатации. Цель выпуска кандидата на выпуск — предоставить продукт заранее выбранной группе пользователей, которая приступает к его тестированию. Пользователи делятся впечатлениями с проектной командой, а та в свою очередь продолжает совершенствовать продукт и исправлять ошибки, обнаруженные в процессе пилотной эксплуатации.

5. Что такое пилотная эксплуатация и каковы ее цели?

Пилотная эксплуатация — это тестирование решения в промышленной среде, а также его проверка ответственными за установку ПО, персоналом отдела сопровождения и конечными пользователями. Основная задача пилотной эксплуатации — продемонстрировать, что решение способно стабильно работать в условиях промышленной эксплуатации и удовлетворяет требования бизнеса.

6. Почему нельзя упрощать и облегчать процедуры пилотного тестирования?

В процессе пилотного тестирования следует по возможности отработать все варианты и сценарии использования системы, которые будут применяться в промышленной эксплуатации,

группируя различные ВИС и СИС. Упрощенное пилотное испытание не даст достаточно информации для выбора стратегии дальнейшего развития проекта. Рекомендуется намеренно вызвать крах пилотной эксплуатации, чтобы отработать процедуры отката, восстановления после сбоев и обеспечения непрерывной работы бизнеса. В результате тщательного пилотного тестирования, выполненного активными пользователями, команда получает информацию о дальнейшем направлении работы: вернуться на предыдущую стадию, приостановить пилотную эксплуатацию, чтобы внести исправления и вернуться к испытаниям, или перейти на этап развертывания (конечная цель пилотных испытаний).

7. Каковы результаты этапа развертывания?

К результатам этапа развертывания относятся:

- системы сопровождения и поддержки;
- процедуры и процессы;
- база знаний, отчеты и журналы;
- хранилище документов, где размещаются все версии документов, конфигураций, сценариев и кода, разработанных в течение проекта;
- отчет о закрытии проекта:
 - проектные документы;
 - результаты опроса пользователей;
 - план дальнейших мероприятий.

К. Кто (или какая группа) лучше всех остальных выполняет развертывание решения?

Это зависит от организации и проекта. В одних случаях лучше всего для этой роли подходит проектная команда, в других — для этого создается специальная команда развертывания. Иногда развертывание поручают внешней команде. В конечном счете важнее всего — создать качественный план развертывания и предоставить его человеку или группе, которая будет заниматься развертыванием.

9. Приведите примеры базовых компонентов. Приведите примеры компонентов, характерных для места установки.

К базовым компонентам относятся: контроллеры доменов, сетевые маршрутизаторы, серверы баз данных, почтовые маршрутизаторы, серверы удаленного доступа. К компонентам, характерным для мест установки, относятся: локальные маршрутизаторы, файловые серверы и серверы печати.

10. Что такое «период затишья» и каковы его особенности?

Так называемый «период затишья» начинается по завершении развертывания. В это время команда передает свои полномочия группе сопровождения и поддержки, которая будет обслуживать продукт. Он длится примерно 15–30 дней, в течение которых собираются статистические данные, касающиеся поведения системы.

11. Какие документы оформляются при закрытии проекта?

При закрытии проекта оформляются следующие документы:

- результаты опроса пользователей, чтобы определить степень их удовлетворенности;
- отчет о закрытии проекта (в том числе документ о формальном закрытии проекта, финальные версии всех основных результатов, сводка полученной от заказчика и пользователей информации и краткое описание того, что следует предпринять далее);
- документ оценки проекта.

12. На какую из ролей MSF возлагается обязанность получить итоговую подпись от заказчика?

По завершении проекта менеджер решения получает у заказчика подпись на акте приемки, что демонстрирует одобрение решения заказчиком и является свидетельством того, что проектная команда слагает с себя полномочия.

13. Назовите два типа оценки проекта, которые MSF рекомендует проводить в конце этапа развертывания?

Рекомендуется, чтобы проект оценивали команда и заказчик. Команда фиксирует, что удалось реализовать в проекте и что следует улучшить. В обзоре заказчика указываются нерешенные проблемы проекта, уровень качества и удовлетворения заказчика, а также операции, которые необходимо выполнить по завершении проекта.

Предметный указатель

A

accessibility aids *см.* специальные возможности

activity diagram *см.* диаграмма, действий

actors catalog *см.* каталог действующих субъектов

API (application programming interface) 33, 205

association *см.* отношения, ассоциации

B

behavioral view *см.* UML, представление, поведения

C

candidate requirements *см.* требования, кандидаты

class diagrams *см.* UML, диаграммы, классов

Class-Responsibility-Collaboration *см.* CRC cohesion *см.* связность

collaboration *см.* диаграмма, сотрудничества

collaboration diagrams *см.* UML, диаграммы, коллективного взаимодействия

component diagrams *см.* UML, диаграммы, компонентов

composite key *см.* ключ, составной

concurrency *см.* параллелизм

coupling *см.* сочленение

coverage testing *см.* тестирование, базовое

CRC (Class-Responsibility-Collaboration) 166

CSDP (ORM conceptual schema design procedure) 54

D

data layer *см.* уровень данных

data store *см.* хранилище данных

denial of service *см.* отказ в обслуживании

dependency *см.* отношения, зависимость

deployment diagram *см.* UML, диаграммы, развертывания

deployment view *см.* UML, представление, развертывания

design view *см.* UML, представление, дизайна

development *см.* разработчик

disclosure *см.* раскрытие информации

DNA (Windows Distributed interNet Applications) 132

E

elevation of privilege *см.* повышение привилегий

entity *см.* сущность

environment view *см.* UML, представление, окружения

exception *см.* исключение

Extensible Markup Language *см.* XML

F

fact instance *см.* экземпляр факта

fact type *см.* тип, факта

FORML (Formal Object-Role Modeling Language) 53

functional specification *см.* функциональная спецификация

G

generalization *см.* отношения, обобщение

H

help-desk *см.* служба поддержки

high-fidelity design *см.* дизайн, с высокой детализацией

horizontal partitioning *см.* таблица, разбиение, горизонтальное

I

implementation view *см.* UML, представление, реализации

isolated storage *см.* изолированное хранилище

L

low-fidelity design *см.* дизайн, с низкой детализацией

M

meaningful sample population *см.* значимый пример популяции

Microsoft Intermediate Language *см.* MSIL

Microsoft Solutions Framework *см.* MSF

MSF (Microsoft Solutions Framework) 1-4, 52, 346

- модель процессов 105, 118

- создание общей картины приложения 13

- управление

- - готовностью 7

- - компромиссами 9, 10

- - проектом 8

- - рисками 6, 93

- этап планирования 105, 107

MSIL (Microsoft Intermediate Language) 280

multiplicity *см.* множественность

O

object diagrams *см.* UML, диаграммы, объектов

Object Role Modeling *см.* ORM

object type *см.* тип объекта

ORM (Object Role Modeling) 49, 53

ORM conceptual schema design procedure *см.* CSDP

P

predicate *см.* предикат

primary key *см.* ключ, основной

problem statement *см.* формулировка задач

process view *см.* UML, представление, процессов

product management *см.* менеджер, решения

program management *см.* менеджер, программы

Q

quiet period *см.* период затишья

R

realization *см.* отношения, реализация

release management *см.* менеджер, по выпуску

repudiation *см.* отказ от причастности

return of investment *см.* ROI

ROI (return of investment) 137

S

scaling out *см.* масштабирование, горизонтальное

scaling up *см.* масштабирование, вертикальное

sequence diagram *см.* диаграмма, последовательности

service-level agreement *ан.* SLA

set *см.* множество

SLA (service-level agreement) 353

smart key *см.* ключ, интеллектуальный

spoofing identity *см.* подмена сетевых объектов

state diagrams *см.* UML, диаграммы, состояний

state management *см.* управление состоянием

status display *см.* строка состояния

STRIDE 272, 274

structural view *см.* UML, представление, структурное

T

tampering *см.* модификация данных

testing *см.* тестировщик

tooltip *см.* всплывающая подсказка

U

UML (Unified Modeling Language) 50, 111, 162, 190

— диаграммы 51

— — БИС 51

— — классов 51

— — коллективного взаимодействия 51

— — компонентов 51

— — объектов 51

— — последовательностей 51

— — развертывания 51

— — состояний 51

— — представление 50, 52

— — в виде набора ВИС 50

— — дизайна 51

— — окружения 51

— — поведения 51

— — пользовательское 50

— — процессов 51

— — развертывания 51

— — реализации 51

— — структурное 51

UML view *см.* UML, представление

Unified Modeling Language *см.* UML

UoD (universe of discourse) *см.* предметная область

usage scenario *см.* СИС

use case *см.* ВИС

use case diagrams *см.* UML, диаграммы,

ВИС

use case view *см.* UML, представление, в виде набора ВИС

user experience *см.* специалист по удобству использования

user interface components *см.* элементы пользовательского интерфейса

user process components *см.* компоненты пользовательского процесса

user view *см.* UML, представление, пользовательское

V

vertical partitioning *см.* таблица, разбиение, вертикальное

vision document *см.* общая картина решения

W

WAP (Wireless Application Protocol) 225

Windows Distributed interNet Applications *см.* DNA

Wireless Application Protocol *см.* WAP

wizard *см.* мастер

X

XML (Extensible Markup Language) 245

A

авторизация 154, 283, 289, 290, 292

анализ информации 42

артефакт 111, 114

архитектура предприятия 32, 42

— бизнес 32

— приложения 32

— процедуры 33

— технологии 33

атрибут 160, 161, 170, 171, 244

аудит 155, 296

аутентификация 154, 283, 289, 290, 293

— Windows 285

— на основе форм 284

— с использованием Passport 284

Б

базовые компоненты 349

бизнес-спонсор 6

В

вариант использования системы *см.* ВИС

версия

— базовая 119

— без обнаруженных ошибок 331

ВИС (вариант использования системы) 15,

43, 45, 60, 63, 75, 80, 86, 111, 115, 153

— диаграмма 121, 126

— отношения 163

— подчиненный 126

— проверка 126

— проверка корректности 129

внутренняя документация 47

всплывающая подсказка 223

Д

действующий субъект 47, 60

диаграмма

— — действий 186, 193

— — классов 186, 191

— — компонентов 193

— — последовательностей 166, 186, 192

— — последовательности 167

- - сотрудничества 162
- дизайн
 - высокоуровневый пользовательского интерфейса 150, 171
 - интерфейса 231
 - интуитивно понятный 218
 - концептуальный 15, 107, 108, 114, 117, 118, 121, 129, 145
 - - анализ 119, 121
 - - исследование 119
 - - оптимизация 120, 134
 - - проверка модели 137
 - - цели 118
 - логический 15, 107, 108, 114, 145, 147, 151, 153, 166, 185
 - - задача 149
 - - множество объектов 173
 - - модель управления 175
 - - отношения 166
 - - результат 150
 - пользовательского интерфейса 215, 218
 - презентационного уровня 216
 - с высокой детализацией 222
 - с низкой детализацией 222
 - физический 15, 107, 108, 114, 145, 150, 182, 184, 185, 186, 204, 207
 - - анализ 187, 190
 - - исследование 187, 188
 - - область действия 184
 - - рационализация 187, 196
 - - реализация 188
 - - результат 186
 - - цель 185
- документ
 - о требованиях 43, 46
 - общей картины и области действия решения 75, 77-79, 86
 - оценки риска 75, 94
 - структуры проекта 75, 88
- З**
- заказчик 6
- значимый пример популяции 55
- «золотая» версия 332
- И**
- изолированное хранилище 283
- индекс 252
 - кластеризованный 253
 - некластеризованный 253
- интерфейс
 - дизайн 231
 - для мобильных устройств 225
 - пользовательский 207, 216, 217, 221, 224, 226, 228, 230, 292, 293
 - пользовательский на основе документов 226
 - стандартный Windows 224
 - компонента 206
 - прикладного программирования см. API
- исключение 63
- источник информации 39
 - артефакты 39
 - люди 39, 40
 - системы 39
- итеративная справочная система 222

К

- кандидат на выпуск 332
- каталог
 - бизнес-правил 47
 - действующих субъектов 47
- класс 162
- ключ 248
 - интеллектуальный 248
 - основной 248
 - составной 248
- композиция 162
- компоненты пользовательского процесса 215
- конечный пользователь 6
- контроль типов 280
- контрольная точка 346
- концепция решения 84, 85
- кэширование 132
- Л**
- локализация 313
- М**
- мастер 223
- масштабирование
 - вертикальное 302
 - горизонтальное 303
- менеджер
 - по выпуску 5, 75, 109, 152, 186, 333
 - программы 5, 74, 108, 152, 185, 333
 - решения 5, 74, 108, 152, 185, 333
- множественность 171
- множество 54, 57
- моделирование ролей объекта см. ORM
- модель
 - базы данных 208
 - данных 150, 170
 - компонентов 186
 - объектов 150, 168, 173
 - предварительная развертывания 194
 - программирования 186, 204
 - процессов 2
 - - водопадная 2, 3
 - - спиральная 2, 3
 - развертывания 201
 - решения 129
- модификация данных 272, 274, 275
- модуль 153
- мониторинг 316
 - анализ тенденций 317
 - журналы событий 317
 - обнаружение ошибок 317
 - обнаружение сбоя 317
 - оповещение 317
 - план 316
 - пороговой загрузки ресурсов 317
 - рабочих характеристик 317
- Н**
- набор 199
- нотация моделирования 49
- О**
- общая картина решения 39, 72, 75
- объект 157, 171
 - бизнес 158
 - определение 157

- проверка 174
- ограничение 67, 83, 243
 - круговое 59
 - - антимметричное 59
 - - асимметричное 59
 - - ациклическое 59
 - - интранзитивные 59
 - - нереклексивное 59
 - - симметричное 59
 - на значение 57
 - на множества 58
 - обязательности принадлежности к роли 57
 - уникальности 56
- окупаемость инвестиций *см.* ROI
- олицетворение 296
- отказ в обслуживании 272, 274, 275
- отказ от причастности 272, 274, 275
- отношения 162, 164, 171
 - ассоциация 162
 - зависимость 162
 - моделирование 166
 - обобщение 162
 - реализация 162
- П
- параллелизм 251
- период затишья 21, 353
- пилотная эксплуатация 322, 330, 334, 341, 343
- повышение привилегий 272, 274, 275
- подмена сетевых объектов 272, 274
- подтип 57
- пользовательский интерфейс управления 177
- правило 243
- предварительный план 84
- предикат 54
- предметная область 53
- презентационный уровень 215
- приложение
 - архитектура 131
 - - клиент-серверная 131
 - - многоуровневая 131
 - - многоуровневая с Web-браузером а качестве клиента 132
 - - основанная на кэше 132
 - - основанная на объектах без сохранения состояния 132
 - оптимизация 251
 - тестирование 251
- проект
 - версия 11, 82
 - допущение 115
 - зависимость 115
 - задача 117
 - закрытие 354
 - инициатор 6
 - история 115
 - итерация 11
 - компромисс 82
 - компромиссы 9
 - контрольная точка 109
 - мониторинг 9
 - область действия 9, 80, 83, 106, 115
 - обновляемые документы 12
 - ограничения 9
 - оценка 83, 354
 - периодические сборки 12
 - поддержка 116
 - предполагаемый результат 109
 - предположение 83
 - приоритет 82
 - сопровождение 116
 - управление изменениями 9
 - цели 85
 - - бизнес-цели 85
 - - проектные 86
 - проектная команда 5
 - прозрачность размещения 303
 - прототип 228, 230
 - профиль пользователя 79, 86
 - процедура концептуального проектирования ORM-схемы *см.* CSDP
 - процесс пользовательский 233

Р

 - раздел 253
 - разработчик 5, 74, 108, 152, 186, 333
 - раскрытие информации 272, 274, 275
 - распределенные Web-приложение Windows *см.* DNA
 - расшифровка 281
 - реквизит 283
 - репликация 33
 - репозиторий 33
 - решение 129
 - риск 93

С

 - сбор информации 32
 - беседа с целевыми группами 34, 36
 - интервьюирование 34, 36
 - наблюдение за действиями пользователя 34
 - обучение, проводимое пользователями 34; 38
 - опросы 34, 37
 - создание прототипов 34, 38
 - стратегия 40
 - сборка 330
 - свойство 160
 - связность 198
 - сервис 129, 158, 171
 - асинхронный коммуникационный 155
 - безопасности 130
 - бизнес 130
 - взаимодействие 155
 - данных 130
 - обмена сообщениями 130, 156
 - обработки ошибок 130
 - общий 191
 - определение 159
 - пользовательский 130
 - распределение по уровням 199
 - резервного копирования 130
 - системный 130, 155
 - транзакционный 155
 - учетная запись 295
 - СИС (сценарий использования системы) 15, 43, 60, 63, 64, 86, 111, 115, 126, 153, 158, 160, 172

- проверка корректности 129
- прогон 174
- система 60
- система управления базами данных см. СУБД
- словарь 48
- служба поддержки 38
- соглашение об уровне сервиса см. SLA
- сочленение 198
- специалист по удобству использования 5, 74, 186, 333
- специальные возможности 223
- столбец 170, 245
- стратегия защиты 115
- строка 245
- строка состояния 223
- СУБД (система управления базами данных) 242, 243
- сущность 170, 244
- сходимость числа ошибок 331
- сценарий
 - будущего состояния 64
 - использования системы см. СИС
 - текущего состояния 64
- Т**
- таблица 170, 245
 - денормализация 257
 - нормализация 254
 - разбиение
 - вертикальное 253
 - горизонтальное 253
- тестирование 18
 - альфа- 337
 - базовое 335
 - бета- 337
 - внешнее базовое 335
 - документации и справочной системы 337
 - задача 340
 - кода перед занесением в систему управления версиями См.
 - конфигурационное 336
 - модульное 336
 - на пригодность к использованию 335
 - нагрузочное 336
 - префиксное 336
 - производительности 337
 - регрессионное 336
 - совместности 336
 - среда 340
- тестировщик 5, 75, 109, 152, 186, 333
- тестовая среда 348
- техническая спецификация 324
- тип
 - данных 246
 - пользователя 246
 - системный 246
 - объекта 54, 59
 - факта 54, 55
- топология
 - данных 196
 - компонентов 196
 - компонентов и данных 195
 - сети 194, 196
- транзакция 252
- требования 66, 81, 115, 173
 - бизнес 117, 125
 - иерархия 122
 - к удалению 115
 - к установке 115
 - кандидаты 117
 - операционные 124
 - отображение 117
 - но интеграции 115
 - пользовательские 117, 124
 - проектные 197
 - процедурные 117, 124
 - системные 117, 124
 - четко определенное 122
- У**
- управление состоянием 197
- уровень данных 240
- Ф**
- формулировка задач 78
- функциональная спецификация 111, 112, 115
 - архитектура и строение инфраструктуры 114
 - архитектура системы 114
 - варианты использования системы 114
 - контекстные модели 114
 - концептуальные модели предлагаемого решения 114
 - логическая модель базы данных 114
 - модели задач 114
 - модели последовательностей задач 114
 - модель логических объектов и сервисов 114
 - описание экранов пользовательского интерфейса 114
 - последовательности экранов пользовательского интерфейса 114
 - правила применения технологий 114
 - распределение сервисов по компонентам 114
 - риск 113
 - сценарии использования системы 114
 - топология распределения компонентов 114
 - физическая модель базы данных 114
- Х**
- хеширование 282
- хранилище данных 33, 240
- Ш**
- шифрование 281
- Э**
- экземпляр 54
- экземпляр факта 54
- элементы пользовательского интерфейса 215
- этап
 - планирования 15, 301
 - развертывания 20, 322, 346
 - разработки 17, 319
 - стабилизации 18, 19, 320, 330, 331, 344
- Я**
- ячейка 284

ЛИЦЕНЗИОННОЕ СОГЛАШЕНИЕ MICROSOFT

прилагаемый к книге компакт-диск

ЭТО ВАЖНО — ПРОЧИТАЙТЕ ВНИМАТЕЛЬНО. Настоящее лицензионное соглашение (далее «Соглашение») является юридическим документом, оно заключается между Вами (физическим или юридическим лицом) и Microsoft Corporation (далее «корпорация Microsoft») на указанный выше продукт Microsoft, который включает программное обеспечение и может включать сопутствующие мультимедийные и печатные материалы, а также электронную документацию (далее «Программный Продукт»). Любой компонент, входящий в Программный Продукт, который сопровождается отдельным Соглашением, подпадает под действие именно того Соглашения, а не условий, изложенных ниже. Установка, копирование или иное использование данного Программного Продукта означает принятие Вами данного Соглашения. Если Вы не принимаете его условия, то не имеете права устанавливать, копировать или как-то иначе использовать этот Программный Продукт.

ЛИЦЕНЗИЯ НА ПРОГРАММНЫЙ ПРОДУКТ

Программный Продукт защищен законами Соединенных Штатов по авторскому праву и международными договорами по авторскому праву, а также другими законами и договорами по правам на интеллектуальную собственность.

1. ОБЪЕМ ЛИЦЕНЗИИ. Настоящее Соглашение дает Вам право:

- a) **Программный продукт.** Вы можете установить и использовать одну копию Программного Продукта на одном компьютере. Основным пользователем компьютера, на котором установлен данный Программный Продукт, может сделать только для себя вторую копию и использовать ее на портативном компьютере.
- b) **Хранение или использование в сети.** Вы можете также скопировать или установить экземпляр Программного Продукта на устройстве хранения, например на сетевом сервере, исключительно для установки или запуска данного Программного Продукта на других компьютерах и своей внутренней сети, но тогда Вы должны приобрести лицензию на каждый такой компьютер. Лицензию на данный Программный продукт нельзя использовать совместно или одновременно на других компьютерах.
- c) **License Pak.** Если Вы купили эту лицензию в составе Microsoft License Pak, можете сделать ряд дополнительных копий программного обеспечения, входящего в данный Программный Продукт, и использовать каждую копию так, как было описано выше. Кроме того, Вы получаете право сделать соответствующее число вторичных копий для портативного компьютера в целях, также оговоренных выше.
- d) **Примеры кода.** Это относится исключительно к отдельным частям Программного Продукта, заявленным как примеры кода (далее "Примеры"), если таковые входят в состав Программного Продукта.
 - i) **Использование и модификация.** Microsoft дает Вам право использовать и модифицировать исходный код Примеров при условии соблюдения пункта (d)(iii) ниже. Вы не имеете права распространять в виде исходного кода ни Примеры, ни их модифицированную версию.
 - ii) **Распространяемые файлы.** При соблюдении пункта (d)(iii) Microsoft дает Вам право на свободное от отчислений копирование и распространение в виде объектного кода Примеров или их модифицированной версии, кроме тех частей (или их модифицированных версий), которые оговорены в файле Readme, относящемся к данному Программному Продукту, как не подлежащие распространению.
 - iii) **Требования к распространению файлов.** Вы можете распространять файлы, разрешенные к распространению, при условии, что: a) распространяете их в виде объектного кода только в сочетании со своим приложением и как его часть; б) не используете название, эмблему или товарные знаки Microsoft для продвижения своего приложения; в) включаете имеющуюся в Программном Продукте ссылку на авторские права в состав этикетки и заставки своего приложения; г) согласны освободить от ответственности и взять на себя защиту корпорации Microsoft от любых претензий или преследований по закону, включая судебные издержки, если таковые возникнут в результате использования или распространения Вашего приложения; и д) не допускаете дальнейшего распространения конечным пользователем своего приложения. По поводу отчислений и других условий лицензии применительно к иным видам использования или распространения распространяемых файлов обращайтесь к Microsoft.

2. ПРОЧИЕ ПРАВА И ОГРАНИЧЕНИЯ

- **Ограничения на реконструкцию, декомпиляцию и дисассемблирование.** Вы не имеете права реконструировать, декомпилировать или дисассемблировать данный Программный Продукт, кроме того случая, когда такая деятельность (только в той мере, которая необходима) явно разрешается соответствующим законом, несмотря на это ограничение.
- **Разделение компонентов.** Данный Программный Продукт лицензируется как единый продукт. Его компоненты нельзя отделять друг от друга для использования более чем на одном компьютере.

- Аренда. Данный Программный Продукт нельзя сдавать и прокат, передавать во временное пользование или уступать для использования в иных целях.
 - Услуги по технической поддержке. Microsoft может (но не обязана) предоставить Вам услуги по технической поддержке данного Программного Продукта (далее «Услуги»). Предоставление Услуг регулируется соответствующими правилами и программами Microsoft, описанными в руководстве пользователя, электронной документации и/или других материалах, публикуемых Microsoft. Любой дополнительный программный код, предоставленный в рамках Услуг, следует считать частью данного Программного Продукта и подпадающим под действие настоящего Соглашения. Что касается технической информации, предоставляемой Вами корпорации Microsoft при использовании ее Услуг, то Microsoft может задействовать эту информацию в деловых целях, в том числе для технической поддержки продукта и разработки. Используя такую техническую информацию, Microsoft не будет ссылаться на Вас.
 - Передача права программное обеспечение. Вы можете безвозвратно уступить все права, регулируемые настоящим Соглашением, при условии, что не оставите себе никаких копий, передадите все составные части данного Программного Продукта (включая компоненты, мультимедийные и печатные материалы, любые обновления, Соглашение и сертификат подлинности, если таковой имеется) и принимающая сторона согласится с условиями настоящего Соглашения.
 - Прекращение действия Соглашения. Без ущерба для любых других прав Microsoft может прекратить действие настоящего Соглашения, если Вы нарушите его условия. В этом случае Вы должны будете уничтожить все копии данного Программного Продукта вместе со всеми его компонентами.
3. АВТОРСКОЕ ПРАВО. Все авторские права и право собственности на Программный Продукт (в том числе любые изображения, фотографии, анимации, видео, аудио, музыку, текст, примеры кода, распространяемые файлы и апплеты, включенные в состав Программного Продукта) и любые его копии принадлежат корпорации Microsoft или ее поставщикам. Программный Продукт охраняется законодательством об авторских правах и положениями международных договоров. Таким образом, Вы должны обращаться с данным Программным Продуктом, как с любым другим материалом, охраняемым авторскими правами, с тем исключением, что Вы можете установить Программный Продукт на один компьютер при условии, что храните оригинал исключительно как резервную или архивную копию. Копирование печатных материалов, поставляемых вместе с Программным Продуктом, запрещается.

ОГРАНИЧЕНИЕ ГАРАНТИИ

ДАННЫЙ ПРОГРАММНЫЙ ПРОДУКТ (ВКЛЮЧАЯ ИНСТРУКЦИИ ПО ЕГО ИСПОЛЬЗОВАНИЮ) ПРЕДОСТАВЛЯЕТСЯ БЕЗ КАКОЙ-ЛИБО ГАРАНТИИ. КОРПОРАЦИЯ MICROSOFT СНИМАЕТ С СЕБЯ ЛЮБУЮ ВОЗМОЖНУЮ ОТВЕТСТВЕННОСТЬ, В ТОМ ЧИСЛЕ ОТВЕТСТВЕННОСТЬ ЗА КОММЕРЧЕСКУЮ ЦЕН НОЗЬ ИЛИ СООТВЕТСТВИЕ ОПРЕДЕЛЕННЫМ ЦЕЛЯМ. ВСЕ РИСК ПО ИСПОЛЬЗОВАНИЮ ИЛИ РАБОТЕ С ПРОГРАММНЫМ ПРОДУКТОМ ЛОЖИТСЯ НА ВАС.

НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ КОРПОРАЦИЯ MICROSOFT, ЕЕ РАЗРАБОТЧИКИ, А ТАКЖЕ ВСЕ, ЗАНЯТЫЕ В СОЗДАНИИ, ПРОИЗВОДСТВЕ И РАСПРОСТРАНЕНИИ ДАННОГО ПРОГРАММНОГО ПРОДУКТА, НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ЗА КАКОЙ-ЛИБО УЩЕРБ (ВКЛЮЧАЯ ВСЕ, БЕЗ ИСКЛЮЧЕНИЯ, СЛУЧАИ УПУЩЕННОЙ ВЫГОДЫ, НАРУШЕНИЯ ХОЗЯЙСТВЕННОЙ ДЕЯТЕЛЬНОСТИ, ПОТЕРИ ИНФОРМАЦИИ ИЛИ ДРУГИХ УБЫТКОВ) ВСЛЕДСТВИЕ ИСПОЛЬЗОВАНИЯ ИЛИ НЕВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ ДАННОГО ПРОГРАММНОГО ПРОДУКТА ИЛИ ДОКУМЕНТАЦИИ, ДАЖЕ ЕСЛИ КОРПОРАЦИЯ MICROSOFT БЫЛА ИЗВЕЩЕНА О ВОЗМОЖНОСТИ ТАКИХ ПОТЕРЬ, ТАК КАК В НЕКОТОРЫХ СТРАНАХ НЕ РАЗРЕШЕНО ИСКЛЮЧЕНИЕ ИЛИ ОГРАНИЧЕНИЕ ОТВЕТСТВЕННОСТИ ЗА НЕПРЕДНАМЕРЕННЫЙ УЩЕРБ, УКАЗАННОЕ ОГРАНИЧЕНИЕ МОЖЕТ ВАС НЕ КОСНУТЬСЯ.

РАЗНОЕ

Настоящее Соглашение регулируется законодательством штата Вашингтон (США), кроме случаев (и лишь в той мере, насколько это необходимо) исключительной юрисдикции того государства, на территории которого используется Программный Продукт.

Если у Вас возникли какие-либо вопросы, касающиеся настоящего Соглашения, или если Вы желаете связаться с Microsoft по любой другой причине, пожалуйста, обращайтесь в местное представительство Microsoft или пишите по адресу: Microsoft Sales Information Center, One Microsoft Way, Redmond, WA 98052-6399.

HARD 'n' SOFT

www.hardnsoft.ru



МНОГОГРАННЫЙ
КОМПЬЮТЕРНЫЙ
ЖУРНАЛ



СПЕШИ

СПЕШИ



СПЕШИ ДОСТИЧЬ БОЛЬШЕГО!

Издательство «Русская Редакция» —

партнер **Microsoft Press** в России —

предлагает широкий выбор литературы по современным информационным технологиям. Мы переводим на русский язык бестселлеры ведущих издательств мира, а также сотрудничаем с компетентными российскими авторами,



 РУССКАЯ РЕДАКЦИЯ

e-mail: info@rusedit.ru; www.rusedit.ru

конкурс
«Читатель
месяца»

Хотите сэкономить на обучении до \$ 1000?

Издательство «Русская Редакция» и учебный центр компании «Инвента» проводят конкурс «Читатель месяца» и будут ежемесячно выбирать двух самых активных читателей книг серии «Учебный курс».

Просто вырежьте купон из книги, помеченной на обложке специальным значком «Читатель месяца», и пришлите нам по адресу: **123317, Россия, г. Москва, ул. Антонова-Овсеенко, д. 13. Издательство «Русская Редакция».**

Лотерея определит победителей месяца. Один купон — один голос!
Чем больше купонов вы пришлете, тем больше у вас шансов выиграть!

Призы победителям — бесплатное обучение в учебном центре «Инвента» в Москве!

Но это не все! Помимо выбранного вами курса по программе сертификации Microsoft, победителей ждут и другие призы — скидка на дальнейшее обучение в учебном центре и подарок от «Русской Редакции».

Подробности конкурса — на сайте издательства «Русская Редакция» (www.rusedit.ru/bonus) и на сайте компании «Инвента» (www.inventa.ru). Там же все новости о конкурсе и о победителях. Телефон для справок (095) 775-8777

Купон участника конкурса «Читатель месяца»

РУССКАЯ РЕДАКЦИЯ

ИНВЕНТА

Ф. И. О.:

E-mail:

Телефон:

Род занятий:

ВНИМАНИЕ! Незаполненные купоны не принимаются.

Конкурс проводится исключительно за счет организаторов и данный купон не может рассматриваться как коммерческое предложение.

Купон из книги **Анализ требований и создание архитектуры решений на основе Microsoft .NET. Учебный курс MCSD. Экз. № 70-300. ISBN 5-7502-0248-8**

КОНКУРС