

Е. П. Зараменских

# УПРАВЛЕНИЕ ЖИЗНЕННЫМ ЦИКЛОМ ИНФОРМАЦИОННЫХ СИСТЕМ

УЧЕБНИК И ПРАКТИКУМ ДЛЯ ВУЗОВ

2-е издание

*Рекомендовано Учебно-методическим отделом высшего образования  
в качестве учебника и практикума для студентов высших учебных заведений,  
обучающихся по инженерно-техническим и экономическим направлениям*

Книга доступна на образовательной платформе «Юрайт» [urait.ru](#),  
а также в мобильном приложении «Юрайт.Библиотека»

Москва • Юрайт • 2021

УДК 004.4(075.8)  
ББК 32.973.202я73  
3-34

*Автор:*

**Зараменских Евгений Петрович** — кандидат технических наук, профессор Департамента бизнес-информатики, руководитель Департамента бизнес-информатики Высшей школы бизнеса Национального исследовательского университета «Высшая школа экономики».

*Рецензенты:*

**Онокой Л. С.** — профессор, доктор социологических наук, профессор кафедры бизнес-информатики Финансового университета при Правительстве Российской Федерации;

**Омарова Н. О.** — профессор, доктор физико-математических наук, профессор кафедры бизнес-информатики и высшей математики Дагестанского государственного университета (г. Махачкала);

**Дерябин А. И.** — кандидат технических наук, доцент Национального исследовательского университета «Высшая школа экономики».

**Зараменских, Е. П.**

3-34 Управление жизненным циклом информационных систем : учебник и практикум для вузов / Е. П. Зараменских. — 2-е изд. — Москва : Издательство Юрайт, 2021. — 497 с. — (Высшее образование). — Текст : непосредственный.

ISBN 978-5-534-14023-1

В курсе рассматривается история и современное состояние информационных систем, а также все этапы их жизненного цикла — от подготовительного этапа до утилизации. Подробно разбирается теория и практика управления жизненным циклом информационных систем, самые разные методологии структурного анализа и моделирования бизнес-процессов, классические и гибкие процессы разработки информационных систем и предназначенные для этого программные инструменты, а также основы управления проектами. Особый интерес представляют практические примеры, которые содержат пошаговые инструкции по анализу бизнес-кейса, а также образцы создаваемых при анализе и проектировании документов.

Соответствует актуальным требованиям Федерального государственного образовательного стандарта высшего образования.

Для студентов высших учебных заведений, обучающихся по инженерно-техническим и экономическим направлениям.

УДК 004.4(075.8)  
ББК 32.973.202я73

*Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.*

© Зараменских Е. П., 2017  
© Зараменских Е. П., 2021, с изменениями  
© ООО «Издательство Юрайт», 2021

ISBN 978-5-534-14023-1

# Оглавление

<b>Список принятых сокращений.....</b>	<b>8</b>
<b>Предисловие .....</b>	<b>10</b>
<b>Тема 1. Информационные системы в современном мире.....</b>	<b>17</b>
1.1. История эволюции информационных систем .....	17
1.1.1. Ключевые задачи и свойства информационной системы ....	19
1.1.2. Автоматизированные системы управления .....	20
1.1.3. Корпоративные информационные системы (КИС) .....	21
1.1.4. Цифровая платформа.....	23
1.2. Этапы развития информационных технологий .....	23
Цифровая трансформация бизнеса.....	31
1.3. Развитие функциональности ИС КИС и управления процессами предприятия .....	32
1.4. Существующие классификации ИС .....	35
1.4.1. Архитектура .....	35
1.4.2. Масштабность .....	36
1.4.3. Степень структурированности задач и характер обработки данных .....	37
1.4.4. Режим работы .....	39
1.4.5. Процессы и уровни управления.....	39
1.4.6. Сфера применения.....	41
1.4.7. Функциональность .....	42
1.4.8. Прошлое и будущее в классификациях ИС .....	48
Контрольные вопросы и задания.....	50
Рекомендуемая литература .....	51
<b>Тема 2. Жизненный цикл информационной системы .....</b>	<b>52</b>
2.1. История развития концепции ЖЦ информационных систем.....	53
2.1.1. ГОСТ 34.601—90 .....	57
2.1.2. ISO/IEC 15288 (ГОСТ Р ИСО/МЭК 15288—2005).....	59
2.2. Жизненный цикл программного обеспечения .....	60
2.2.1. SWEBOK (ISO/IEC TR 19759:2015) .....	62
2.2.2. ISO/IEC 12207:2008 (ГОСТ Р ИСО/МЭК 12207—2010).....	67
2.3. Модели жизненного цикла программного обеспечения .....	69
2.3.1. Каскадная модель.....	69
2.3.2. Каскадная модель с промежуточным контролем .....	70
2.3.3. Метод разработки через тестирование (V-модель).....	71
2.3.4. Спиральная модель .....	72

2.3.5. Итерационная модель .....	73
2.3.6. Инкрементная модель .....	76
2.3.7. Эволюционная модель развития .....	77
<i>Контрольные вопросы и задания.....</i>	78
<i>Рекомендуемая литература .....</i>	78
<b>Тема 3. Фазы жизненного цикла информационных систем и специфика каждой из них .....</b>	<b>79</b>
3.1. Подготовительный этап .....	79
3.1.1. Экспресс-обследование.....	80
3.1.2. Технико-экономическое обоснование .....	83
3.1.3. Оценка целесообразности проекта (TELOS) .....	84
3.1.4. Выбор программного решения.....	85
3.2. Анализ и постановка задачи .....	87
3.2.1. Информационное обследование предприятия .....	88
3.2.2. Описание бизнес-процессов .....	92
3.2.3. Сбор требований.....	93
3.2.4. Подготовка технического задания .....	98
3.3. Проектирование .....	99
3.3.1. Техническое проектирование.....	101
3.3.2. Рабочее проектирование и прототипирование .....	104
3.3.3. Объектно-ориентированный подход к проектированию....	105
3.3.4. Основы Unified Modeling Language .....	108
3.3.5. Общая характеристика UML.....	110
3.3.6. Структура UML.....	111
3.3.7. «4 + 1 представления» архитектуры ИС.....	115
3.3.8. Диаграммы в UML.....	118
3.4. Разработка .....	131
3.4.1. Закупка ПО .....	131
3.4.2. Настройка конфигураций.....	132
3.4.3. Создание ролей пользователей .....	135
3.4.4. Миграция данных .....	136
3.4.5. Разработка сценария тестирования .....	139
3.4.6. Тестовая эксплуатация.....	140
3.4.7. Доработка по результатам тестирования.....	143
3.4.8. Прием результатов тестирования .....	144
3.5. Развёртывание и внедрение .....	144
3.5.1. Закупка и настройка требуемой ИТ-инфраструктуры.....	146
3.5.2. Ввод начальных остатков .....	149
3.5.3. Обучение пользователей .....	149
3.5.4. Развёртывание системы на рабочих местах .....	151
3.5.5. Основные виды тестирования .....	153
3.5.6. Опытно-промышленная эксплуатация .....	155
3.5.7. Приемо-сдаточные испытания и интеграционное тестирование .....	158

3.6. Эксплуатация.....	159
3.6.1. Сопровождение эксплуатации.....	160
3.6.2. Модернизация.....	167
3.7. Утилизация .....	174
Технические аспекты .....	174
Организационные аспекты .....	175
Коммерческие аспекты.....	176
Юридические аспекты.....	176
Контрольные вопросы и задания.....	177
Рекомендуемая литература .....	178
<b>Тема 4. Анализ и постановка задачи .....</b>	<b>179</b>
4.1. Основы структурного анализа.....	179
4.2. Функциональный анализ и проектирование .....	182
4.2.1. Основы функционального анализа и проектирования .....	183
4.2.2. Методология SADT .....	184
4.2.3. Семейство методологий IDEF .....	186
4.2.4. Методология DFD .....	190
4.3. Формирование информационной модели.....	192
4.3.1. Методологии проектирования данных .....	194
4.3.2. Инфологическое и даталогическое проектирование .....	197
4.3.3. Диаграммы IDEF1X .....	198
4.3.4. Диаграммы «сущность-связь» (ERD) .....	202
4.3.5. Информационный инжиниринг.....	208
4.4. Описание бизнес-процессов .....	208
4.4.1. Бизнес-процессы в анализе и проектировании ИС.....	209
4.4.2. Нотации основных методологий моделирования .....	211
4.4.3. Программные продукты моделирования деятельности организации.....	219
4.5. Объектно-ориентированный анализ.....	221
Контрольные вопросы и задания.....	222
Рекомендуемая литература .....	222
<b>Тема 5. Проектирование .....</b>	<b>223</b>
5.1. Общие требования к методологиям проектирования ИС .....	224
5.2. История развития методологий проектирования ИС .....	226
5.3. Каноническое проектирование .....	232
Каноническое проектирование на основе ГОСТ 34.601—90.....	234
5.4. Проектирование с использованием CASE-средств .....	236
5.4.1. Unified Process и Rational Unified Process .....	236
5.4.2. Рабочий поток «Бизнес-моделирование».....	245
5.4.3. Рабочий поток «Анализ и проектирование».....	248
5.4.4. Рабочий поток «Реализация».....	255
5.4.5. Рабочий поток «Развертывание».....	257
5.4.6. CASE-средства объектно-ориентированного анализа и проектирования.....	259

5.5. Типовое проектирование .....	261
5.5.1. Типовое проектирование на основе Accelerated SAP .....	265
5.5.2. Типовое проектирование на основе Accelerated SAP (ASAP) & Agile .....	269
5.5.3. Типовое проектирование на основе SAP Activate .....	272
5.6. Документирование требований .....	273
5.6.1. Документирование требований в ГОСТ 34.602—89 .....	273
5.6.2. Документирование требований по Software Requirement Specification (SRS) .....	276
5.6.3. Документирование требований по Rational Unified Process .....	277
5.6.4. Выявление атрибутов требований .....	279
Контрольные вопросы и задания .....	281
Рекомендуемая литература .....	281
<b>Тема 6. Разработка .....</b>	<b>282</b>
6.1. Жесткие и гибкие подходы к разработке ИС .....	282
6.2. Традиционные подходы к разработке .....	283
6.2.1. Microsoft Solution Framework (MSF) .....	285
6.2.2. Rapid Application Development (RAD) .....	286
6.3. Гибкие методологии и подходы .....	288
6.3.1. Scrum .....	291
6.3.2. Kanban .....	295
6.3.3. eXtreme Programming .....	295
6.3.4. Feature Driven Development (FDD) .....	297
6.3.5. Domain-Driven Design (DDD) .....	300
6.4. CASE-средства (Computer Aided Software Engineering) .....	300
6.5. Тестирование .....	304
6.5.1. Роли и результаты тестирования .....	306
6.5.2. Задачи тестирования .....	307
6.5.3. Виды тестирования .....	310
6.5.4. Правила и рекомендации .....	312
6.5.5. Инструментальные средства тестирования .....	314
Контрольные вопросы и задания .....	316
Рекомендуемая литература .....	317
<b>Тема 7. Особенности проектов в области информационных технологий на фазах ЖЦИС .....</b>	<b>318</b>
7.1. Управление фазами ЖЦИС в контексте проектной деятельности .....	322
7.1.1. Управление заинтересованными сторонами .....	322
7.1.2. Управление содержанием .....	327
7.1.3. Управление сроками проекта .....	329
7.1.4. Управление стоимостью проекта .....	337
7.1.5. Управление рисками .....	342
7.1.6. Управление качеством .....	354

ISO 10006:2003 (ГОСТ Р ИСО 10006—2005) .....	356
ISO 15504/SPICE .....	359
7.1.7. Управление командой проекта .....	363
7.1.8. Управление портфелем проектов .....	371
7.1.9. Офис управления проектами .....	375
7.2. Корпоративные методологии .....	381
7.2.1. Методологии компаний Microsoft.....	381
7.2.2. Oracle Unified Method .....	390
7.3. Российские и международные стандарты .....	397
7.3.1. PMBoK .....	397
7.3.2. PRINCE2 .....	403
7.3.3. ISO 21500:2012.....	407
7.3.4. ГОСТ Р 54869—2011 .....	409
Контрольные вопросы и задания.....	410
Рекомендуемая литература .....	410
<b>Тема 8. Пример бизнес-кейса и практическое задание .....</b>	<b>412</b>
8.1. Описание компании.....	413
8.1.1. Описание бизнес-процессов .....	416
Прием, обработка и доставка посылок и отправлений .....	416
8.1.2. Описание ИТ-инфраструктуры .....	425
Программное обеспечение .....	426
8.1.3. Отзывы о компании на сайте.....	427
8.2. Контрольное задание .....	428
8.2.1. Постановка Задачи.....	428
8.2.2. Планирование проекта .....	430
8.2.3. Структурный анализ деятельности компании .....	437
Функциональный анализ .....	442
Анализ информационной модели .....	449
8.2.4. Требования к информационной системе.....	450
8.2.5. Выбор готового программного обеспечения.....	456
8.2.6. Проектирование .....	460
8.2.7. Тестирование.....	470
8.2.8. Заключение к контрольному заданию.....	470
8.2.9. Примеры документов .....	471
8.3. Дополнительное задание по кейсу: разработка проектного решения системы мониторинга отправлений, курьеров и транспортных средств .....	492
8.3.1. Обследование мониторинга деятельности (As-Is) .....	492
8.3.2. Разработка модели (To-Be) .....	493
8.3.3. Разработка модели перехода к целевой модели (Roadmap) .....	493
<b>Нормативные документы .....</b>	<b>495</b>

## **Список принятых сокращений**

- ARIS** (ARchitecture of Integrated Information Systems) — архитектура интегрированных информационных систем
- BCP** (Business continuity plan) — план обеспечения непрерывности бизнеса
- BPMN** (Business Process Modeling Notation) — нотация моделирования бизнес-процессов организации
- BSC** (Balanced Scorecard) — сбалансированная система показателей
- CASE** (Computer-Aided Software Engineering) — набор инструментов и методов программной инженерии для проектирования программного обеспечения; CASE-средства — инструменты автоматизации процессов проектирования и разработки программного обеспечения
- CobiT** (Control Objectives for Information and Related Technology) — задачи информационных и смежных технологий
- DRP** (Disaster recovery plan) — план аварийного восстановления
- eEPC** (Extended Event-driven Process Chain) — расширенная событийная цепочка процессов
- ISACA** (Information Systems Audit and Control Association) — международная ассоциация аудита и контроля за информационными системами
- IPMA** (International Project Management Association) — некоммерческая профессиональная ассоциация по управлению проектами
- IT** (Information Technology) — информационные технологии
- ITIL** (IT Infrastructure Library) — библиотека инфраструктуры информационных технологий
- ITSM** (IT Service Management) — управление IT-услугами
- NPV** (Net Present Value) — чистая приведенная стоимость проекта
- OMG** (Object Management Group) — группа объектного управления
- PMBOK** (Project Management Body of Knowledge) — свод знаний по управлению проектами
- PRINCE2** (PRojects IN Controlled Environments) — методология управления программами/проектами в организации
- RAD** (Rapid Application Development) — средства быстрой разработки приложений
- ROI** (Return On Investment) — возврат инвестиций

**RUP** (Rational Unified Process) — унифицированный процесс разработки программного обеспечения, созданный компанией Rational Software

**SEP** (Software Engineering Process) — процесс инжиниринга программного обеспечения

**SLA** (Service Level Agreement) — соглашение об уровне предоставления услуг

**SWEBOK** (Guide to the Software Engineering Body of Knowledge) — свод знаний по программной инженерии

**TCO** (Total Cost of Ownership) — совокупная стоимость владения

**UML** (Unified modeling language) — универсальный язык моделирования

**UP** (Unified Process) — унифицированный процесс

**АРМ** — автоматизированное рабочее место

**АС** — автоматизированная система

**АСУ** — автоматизированная система управления

**ГИС** — географическая информационная система

**ЖЦ** — жизненный цикл

**ЖЦИС** — жизненный цикл информационной системы

**ИС** — информационная система

**ИТ** — информационные технологии

**КИС** — корпоративная информационная система

**КСПД** — корпоративные сети передачи данных

**КХД** — корпоративное хранилище данных

**ЛВС** — локальная вычислительная сеть

**НСИ** — нормативно-справочная информация

**ОПЭ** — опытно-промышленная эксплуатация

**ОС** — операционная система

**ПО** — программное обеспечение

**ПР** — программное решение

**ПС** — программные средства

**СУБД** — система управления базой данных

**СХД** — система хранения данных

**ТЗ** — техническое задание

**ТЭО** — технико-экономическое обоснование

**ЦОД** — центр обработки данных

**ЭВМ** — электронная вычислительная машина

**ЭЦП** — электронная цифровая подпись

## Предисловие

В настоящее время неоспоримым является утверждение, что активное применение информационных технологий способствует существенному увеличению результативности любого предприятия. Действительно, ИТ повышают эффективность компаний, позволяют сократить издержки, оптимизировать производственные процессы, а зачастую именно за счет информационных технологий организация получает конкурентное преимущество и опережает конкурентов.

Представить деятельность современной компании без информационных технологий практически невозможно, и в особенности это касается процессов управления. И заказчики, и исполнители всегда стремятся к гарантированному получению высоких результатов в проектах, связанных с ИС. Но, к сожалению, на сегодняшний день в данных областях наблюдается очень высокий процент неудачных проектов. Именно поэтому сейчас как никогда актуальна потребность в поиске новых методов и возможностей для увеличения числа успешных проектов, результаты которых приносят настоящую пользу предприятиям.

Практика показала, что достичь подобных результатов можно за счет выполнения определенной последовательности действий, которые лежат в основе жизненного цикла любой информационной системы. Только в этом случае можно получить запланированный результат с заранее заданным уровнем качества и в рамках согласованного бюджета.

Данный курс содержит информацию о различных аспектах и этапах жизненного цикла информационных систем и состоит из восьми тем.

Важно понимать, для чего создаются ИС, из каких компонентов они состоят, а также с какими типами ИС мы сталкиваемся. Именно этому и посвящена *первая тема* данного курса.

В данной теме раскрывается уже достаточно долгая история информационных систем, сумевших за прошедшие десятилетия изменить не только экономику, но и весь мир. В теме подробно рассмотрено, как менялись информационные системы на протяжении своей истории и в каком виде они существуют сегодня.

Отдельного внимания заслуживает история отечественных ИС, которые многие десятилетия представляли собой реальную альтернативу западным аналогам. В период активного развития отечест-

венных ИС было создано множество уникальных продуктов, а также был накоплен колоссальный опыт, который важно учитывать в современных системах.

Знание функциональности современных ИС и их места в деятельности предприятий является основой для любого дальнейшего обучения, поскольку без этого фундамента сложно рассматривать применение ИС в современном многообразном мире. Первая тема показывает, как современные ИС покрывают основные сферы деятельности современных предприятий, что отчетливо отражено в приведенной классификации ИС.

Говоря о прошлом и настоящем, всегда хочется попытаться заглянуть в будущее. Заключительная часть первой темы посвящена анализу прошлого и будущего в классификации ИС. Иными словами, она рассказывает о новых требованиях, предъявляемых современным потребителем к ИС, и о том, как это уже в обозримом будущем отразится на классификации ИС.

Другое базовое понятие, которому во многом и посвящен данный курс, — это этапы создания, эксплуатации и утилизации, или жизненный цикл ИС, который подробно рассматривается во второй теме.

На основе практики создания и эксплуатации ИС были сформированы модели, стандарты и методологии, которые описывают ЖЦИС и являются его методологической основой. Жизненный цикл информационной системы широко отражен как в международных стандартах ISO/IEC и в отечественном ГОСТ, так и во фреймворках по управлению ИТ, созданных непосредственно практиками применения ИТ в интересах бизнеса. Многообразие стандартов и методологий формирует у обучающихся основные представления о современном развитии данной области знания.

Важно, что жизненный цикл рассматривается не только с точки зрения ИС. Отдельно разбираются стандарты и методологии, описывающие жизненный цикл программного обеспечения. Их широкое применение обусловлено потребностью в создании качественного ПО, способного удовлетворять требования заказчика и быть по-настоящему полезным. Надо помнить, что ПО является основной частью информационной системы, поэтому подходы к управлению ЖЦИС и ЖЦПО будут во многом совпадать, но понятие ИС шире, и в нем надо дополнительно учитывать аппаратную составляющую, данные, персонал и ИТ-процессы.

В теме рассмотрены модели жизненного цикла ПО, которые уже успели стать классическими. Их подробное описание призвано сформировать у обучающегося глубокое понимание необходимых этапов создания сложного ПО.

Третья тема данного курса посвящена общим стадиям жизненного цикла любой ИС. В общем виде здесь рассмотрены основные

этапы, которые проходит ИС, начиная с момента замысла и заканчивая снятием с эксплуатации.

Фаза планирования проекта имеет принципиальное значение для успеха всех дальнейших работ. Именно на этой стадии формируется общее понимание предполагаемых результатов проекта. Вместе с этим оценивается целесообразность проекта, а также его ожидаемый эффект: действительно, задача ИТ — принести бизнесу конкретную пользу, а не стать результатом следования модному тренду.

Вслед за этим производится выявление потребностей заинтересованных лиц, поиск решения. Проектная команда анализирует предприятие, разбивая его на логические составные части: функции, бизнес-процессы, информационные потоки, структуру и т. п. В результате удается задокументировать требования заинтересованных лиц, сформулировать требования к ИС и подготовить техническое задание или спецификацию требований, которые определят облик и функциональность создаваемой ИС, а значит, и будущее решение.

На следующих стадиях проектирования и разработки формируется подробное описание и непосредственный программный код будущей информационной системы, а тестировщики по подготовленным тестам дают заключение о соответствии созданного ПО требованиям.

Но, разумеется, проект по созданию информационной системы не заканчивается простым созданием программного кода или настройкой закупленной конфигурации от вендора. Информационную систему нужно развернуть на рабочих местах и внедрить в деятельность предприятия. С этой целью производится закупка требуемого аппаратного обеспечения и создание ИТ-инфраструктуры, обучение пользователей, опытная эксплуатация и непосредственная сдача системы заказчику.

Следующая стадия, эксплуатация ИС, — самая ожидаемая с точки зрения бизнеса. Именно на этой фазе система позволяет вернуть вложенные инвестиции и достичь поставленных целей. Для ее бесперебойного функционирования требуется деятельность специалистов, способных осуществлять сопровождение и техническую поддержку. Это может быть выполнено как силами внутренней ИТ-службы, так и передано на аутсорсинг.

Практически всегда ИС неоднократно проходит модернизацию для обеспечения соответствия изменяющимся бизнес-потребностям. В связи с этим важно понимать, как, с помощью каких процедур и инструментов, можно внести изменения в уже существующую ИС и улучшить ее.

И, разумеется, рано или поздно для любой системы наступит пора утилизации. Для организации процессов утилизации ИС требуется глубокое понимание технических, организационных и коммерческих аспектов данного этапа жизненного цикла.

Для формирования целостной картины жизненного цикла ИС его фазы в третьей теме рассмотрены на относительно высоком уровне абстракции, что позволяет сформировать целостное представление о том пути, который проходят заказчики и ИТ-специалисты.

Можно утверждать, что успех или неудача любого проекта заложивается еще на самых ранних стадиях, а конкретно — в момент анализа и постановки задачи. Четвертая тема данного курса описывает действия, необходимые для сокращения числа ошибок и неудач, которые могут быть заложены в проекты еще на самом старте. В теме подробно рассматривается подход, получивший название структурного анализа. Данный подход позволяет получить представление о деятельности любой компании.

В ходе структурного анализа аналитики строят функциональные и информационные модели, моделируют потоки данных и т. д. Наборы этих моделей позволяют получить представление о предметной среде, в которой будет использоваться ИС. Основные нотации, применяемые на сегодняшний день, подробно рассмотрены в четвертой теме.

Отдельное внимание удалено бизнес-процессам, которые связывают различные отделы и сотрудников предприятия воедино и позволяют компании достичь конкретных целей. Получившие наибольшее распространение нотации моделирования бизнес-процессов также изложены в соответствующих параграфах. Моделирование бизнес-процессов, применяемое при автоматизации, является составной частью более широкого понятия — управления бизнес-процессами.

Анализ предметной области становится базисом для дальнейшего проектирования. Именно ему и посвящена пятая тема учебника. Эта тема подробно раскрывает общую логику методологий, посвященных проектированию ИС, а также принципы проектирования.

Особое внимание уделяется объектно-ориентированному анализу и проектированию. Тема включает описание общей логики и структуры языка Unified Modeling Language (UML). Возможности UML сегодня позволяют использовать данный язык в качестве инструмента для проектирования архитектуры ИС. Представление прецедентов, логическое представление, представление реализации, процессное представление и представление развертывания составляют единую архитектуру информационной системы, которая позволяет рассмотреть ИС на высоком уровне абстракции. Представления архитектуры с примерами изложены в пятой теме. Там же показана их взаимосвязь друг с другом и с диаграммами UML. Представления архитектуры позволяют взглянуть на ИС с разных сторон, с разных точек зрения, что позволяет не упустить ни один значимый элемент или проектный аспект.

Все представления архитектуры подробно детализируются при помощи UML-диаграмм. С их помощью производится непосред-

ственное проектирование информационной системы с требуемым уровнем детализации. В пятой теме подробно рассмотрены основные диаграммы UML: диаграмма прецедентов, диаграмма классов, диаграмма деятельности, диаграммы взаимодействия, диаграмма состояний, диаграмма компонентов и диаграмма развертывания. Для каждой диаграммы приведены основные элементы графической нотации и примеры, иллюстрирующие возможности их практического применения.

Кроме того, в данной теме представлен обзор основных CASE-средств проектирования при помощи языка UML. Знание этих инструментов позволяет сформировать у студентов практические навыки проектирования, которые окажутся полезными не только в стенах университета, но и непосредственно пригодятся в работе.

Следующий за проектированием этап жизненного цикла информационной системы — это разработка. Вопросам разработки ИС посвящена шестая тема данного курса. В ней рассматриваются традиционные и гибкие подходы к разработке информационных систем, а также применение CASE-средств для достижения максимальной эффективности данного этапа.

В последние годы постепенно становятся все более популярными гибкие методологии и подходы к разработке, поэтому их изучению удалено значительное внимание. Действительно, сегодня многие компании и проектные команды отказываются от жесткого формализма традиционных методологий и стремятся действовать как можно более гибко и свободно. Надо отметить, что гибкие методологии никогда не заменят традиционные, задача специалистов — грамотно применять подходы в зависимости от типа проекта.

Помимо непосредственной разработки, шестая тема также посвящена вопросам тестирования. Подробно рассмотрены задачи и виды тестирования. Понимание того, с какой целью выполняются тесты, — это залог успешного тестирования, позволяющего существенно повысить качество программного продукта. Отдельно приведены рекомендации по уменьшению числа ошибок, возникающих на этапе тестирования. В данной теме рассмотрены основные роли участников тестирования. Знание и понимание этих ролей позволяет повысить качество ПО.

Как и в предыдущей теме, здесь обзорно рассмотрены основные CASE-средства тестирования. Большинство этих средств применяется сегодня на практике.

Отдельного внимания заслуживает проектный подход, применяемый при реализации этапов. Все фазы жизненного цикла ИС имеют проектный характер, за исключением фазы эксплуатации. Фаза эксплуатации имеет операционный характер. На сфере управления проектами фокусируется седьмая тема курса. Применение проектного управления позволяет получить запланированные результаты,

а в итоге — по-настоящему эффективную систему, которая позволит компании добиться поставленных целей в своей производственной деятельности.

ИТ-проект очень многогранен. В рамках такого проекта обязательно производится выявление заинтересованных сторон, имеющих непосредственное отношение к проекту и способных оказывать на него влияние, и управление ими. Классическая триада — управление сроками, управление стоимостью и управление качеством — это именно то, что может привести проект к успеху или неудаче. При этом данные блоки имеют свою специфику, которая будет представлена в седьмой теме.

Управление рисками проекта становится все более важной частью ИТ-проекта. Масштабы систем растут, вместе с ними увеличивается стоимость и важность проекта, а соответственно, и риски. Это имеет колossalное значение, поскольку в ряде случаев коммерческие показатели компаний могут существенно снизиться из-за влияния неучтенных рисков.

Отдельного внимания заслуживает управление портфелем проектов и современный тренд на формирование офиса управления проектами. Наибольшую значимость эти направления имеют в крупных организациях, специализирующихся на проектах или регулярно сталкивающихся с потребностью управления множеством взаимосвязанных проектов.

На сегодняшний день существуют российские и международные стандарты управления проектами информационных систем. Их изучение позволяет получить проверенные практикой знания для успешной реализации ИТ-проектов. Отдельного внимания заслуживают корпоративные методологии, предложенные основными вендорами. Они содержат обобщение опыта по созданию и внедрению решений, предлагаемых этими поставщиками.

Чтобы сделать обучение более приближенным к реальности, восьмая тема содержит описание производственной деятельности реальной компании, специализирующейся в сфере доставки. Будучи слабо автоматизированной, организация столкнулась с проблемами в своем рыночном сегменте и приняла решение автоматизировать свою деятельность. Описание компании, приведенной в восьмой теме, основано на реальном кейсе.

Соответствующее задание и пример его реализации также включены в тему. При этом само задание и пример выполнения сопровождаются множеством комментариев, которые позволят сформировать корректное представление о практическом применении методов, методологий, инструментов и программных средств, описанных в курсе.

Несмотря на специфику предметной области и отдельные сложности, данный кейс может рассматриваться как пример при орга-

низации контрольных, курсовых работ, проектных практикумов для формирования у студентов основных компетенций, связанных с управлением жизненным циклом информационной системы.

Практические задания были проверены на занятиях в бакалавриате и магистратуре НИУ ВШЭ и Финансовом университете при Правительстве РФ.

В результате освоения материала учебника студент должен:

**знать**

- современные стандарты и методы реализации фаз жизненного цикла ИС;
- основные методы анализа предметной области и построения концептуальной модели информационной системы;
- современные методы совершенствования бизнес-процессов на основе информационных технологий;
- современные методологические основы проектирования информационных систем;
- современные методы проектирования ИС на физическом уровне;

**уметь**

- использовать полученные знания для выбора методологии и технологий проектирования ИС;
- использовать методы реализации фаз жизненного цикла ИС на практике;
- обоснованно принимать решения в части выбора инструментальных средств проектирования ИС;
- осуществлять анализ и моделирование предметной области;
- использовать методы совершенствования бизнес-процессов на основе информационных технологий;
- разрабатывать модели физического уровня проектируемой ИС;

**владеть**

- навыками обобщать, делать выводы и давать предложения, используемые для принятия решений в повышении эффективности проектирования ИС;
- техниками анализа предметной области;
- навыками практического использования методов совершенствования бизнес-процессов на основе информационных технологий;
- методами визуального проектирования информационной системы;
- структурными и объектно-ориентированными методами анализа требований и проектирования информационной системы;
- навыками разработки диаграмм компонентов и размещения UML;
- навыками использования на практике полученных знаний.

# **Тема 1**

## **ИНФОРМАЦИОННЫЕ СИСТЕМЫ В СОВРЕМЕННОМ МИРЕ**

---

В этой теме рассказывается, что такое информационные системы, для чего они создаются, из каких компонентов состоят и какие типы ИС существуют сегодня. В теме содержится история информационных систем, которые за достаточно короткий промежуток времени смогли серьезно изменить не только экономику, но и общество в целом. Прослеживается изменение ИС на протяжении всего их существования и приводится прогноз дальнейшего развития.

После изучения данной темы студент будет:

*знать*

- понятие информационной системы;
- историю развития информационных систем;
- существующие классификации информационных систем;
- функциональность информационных систем;

*уметь*

- классифицировать информационные системы на основе одного или нескольких критериев классификации;

*владеть навыками*

- выделения отличительных черт информационных систем для отнесения их к тому или иному классу.
- 

### **1.1. История эволюции информационных систем**

Современные информационные системы являются не просто средством автоматизации и повышения эффективности, но неотъемлемым элементом архитектуры компании. Организации все чаще вкладывают значительные средства в системы, способные помочь компании выжить в стремительно изменяющейся внешней среде и условиях конкуренции. Однако что же именно способны дать предприятию информационные системы, и как предприятие может в свою очередь способствовать развитию ИС и максимизации выгод от их применения? Для ответа на этот вопрос необходимо рассмотреть само понятие информационной системы.

Существуют сотни различных определений данного термина, однако мы рассмотрим только те, которые наиболее точно подходят

к излагаемому ниже материалу. Автор предлагает следующее определение информационной системы.

---

**Информационная система** — совокупность информационного, программного и технического обеспечения, а также персонала, за счет ИТ-процессов обеспечивающих информационную поддержку выполнения бизнес-процессов или информационные потребности заинтересованных лиц.

---

Наиболее часто цитируется определение М. Р. Когаловского<sup>1</sup>:

«Информационной системой называется комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал, обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей».

Соответственно, информационная система состоит из определенного набора частей, которые должны быть собраны вместе.

1. **Техническое обеспечение.** Все аппаратные и коммуникационные средства относятся к техническому обеспечению. Эта категория включает в себя сами компьютеры (как персональные места, так и компьютеры, используемые в качестве серверов) и все вспомогательное оборудование. Среди вспомогательного оборудования отметим устройства ввода/вывода, устройства хранения данных, устройства связи, инженерное оборудование и пр.

2. **Программное обеспечение.** Термин ПО относится к системным и прикладным компьютерным программам. Именно прикладные программы выполняют работу внутри аппаратных частей системы и должны производить полезную информацию из исходных данных.

3. **Информационное обеспечение.** Это все данные, которые используются в программах для получения полезной информации (прикладные данные, системные данные, первичные данные, расчетные данные, нормативно-справочная информация и т. д.). Как и программы, данные обычно хранятся в машиночитаемой форме на устройствах хранения, пока система не использует их. Для обеспечения ввода, хранения, обработки и представления данных широко используются различные виды информационных технологий.

4. **ИТ-процессы.** Это процедуры работы с ИС, которые являются направляющими для персонала на всех фазах ЖЦИС. Обычно такие

---

<sup>1</sup> Когаловский М. Р. Глоссарий по информационному обществу / М. Р. Когаловский [и др.] ; под общ. ред. Ю. Е. Хохлова. М. : Институт развития информационного общества, 2009. С. 160.

процедуры прописаны в регламентах, инструкциях на рабочих местах, руководствах, технической документации и прочих документах. Именно ИТ-процессы обеспечивают своевременность сбора, обработки, предоставления информации, развития ИС в соответствии с этапами жизненного цикла информационных систем и соответствия требованиям пользователей, а также на этапах создания, эксплуатации, модернизации и утилизации. ИТ-процессы на предприятии формируются на основе практики, стандартов, сводов знаний, рекомендаций поставщиков решений, особенностей технического и программного обеспечения и т. п. Процедуры, которые не formalизованы в виде документов, могут вызывать разнотечения у персонала, работающего с ИС.

**5. Персонал.** Каждая система нуждается в людях, которым она будет полезна, и отдельно — в людях, которые будут обеспечивать работу ИС в соответствии с требованиями. Именно наличие людей, которые по определенным регламентам вводят, получают и обрабатывают необходимую информацию, а также обеспечивают эксплуатацию системы, отличает ИС от набора программно-аппаратных средств.

На сегодняшний день на рынке действует достаточно большое число компаний с доступными и функциональными программными продуктами для недорогой и эффективной автоматизации хозяйственной деятельности. Критерии выбора продукта для внедрения будут рассматриваться в дальнейшем. При этом примем во внимание тот факт, что ИС по умолчанию должна удовлетворять потребности заказчика, иначе смысла в ее создании и использовании по-просту нет.

#### **1.1.1. Ключевые задачи и свойства информационной системы**

Информационная система призвана удовлетворять информационные потребности потребителей в пределах заранее определенной области (за счет хранения, сбора, обработки и представления информации). Для описания глобального результата использования ИС в компании чаще всего применяется словосочетание «единое информационное пространство». В пределах этого пространства обеспечивается доступ к необходимым данным для сотрудников каждой ступени управленческой иерархии.

**Какие средства и свойства ИС обеспечивают достижение необходимых результатов?**

- При помощи ИС оптимизируется использование математических методов и их применение к решению сложных управленческих задач.

- Конечный пользователь ИС не является специалистом в области информационных технологий и вычислительной техники.

Таким образом, информационная система обязана предоставлять в распоряжение пользователя клиентские приложения с интуитивно понятным и простым интерфейсом.

- В основании любой ИС находятся средства и инструменты хранения и доступа к собранным данным.
  - ИС позволяет регистрировать информацию в реальном времени.
  - ИС открыта и масштабируема, а значит, позволяет добавлять новых пользователей, расширять информационное покрытие на другие подразделения и компании и, разумеется, расширять функциональность.
  - ИС позволяет консолидировать данные для организационной структуры автоматически.
  - ИС упрощает процедуру регистрации входящих и собранных данных и их дальнейшую обработку, а также исключает ошибочную повторную регистрацию данных.
  - ИС содержат нормативно-справочную информацию, т. е. единые для многих функций и процессов организации справочники (коды и основные данные сотрудников, продукции, поставщиков, клиентов), позволяющие быстро находить необходимую информацию и избегать ее дублирования при создании новых объектов.

#### **Результаты использования ИС на предприятии:**

- ИС способствует оптимизации труда и избавляет сотрудников от значительной доли рутинных задач, а также снижает трудозатраты на решение управленческих вопросов и способствует их равномерному распределению на всех пользователей системы.
  - ИС минимизирует вероятности появления ошибки в процессе обработки или передачи информации.
  - ИС способствует уменьшению и совершенствованию бумажного документооборота.
  - ИС способствует снижению затрат на производство товаров и услуг, а также содействует их оптимизации.
  - ИС предоставляет оперативный, удобный, достоверный доступ к исчерпывающей информации, причем формат представления понятен руководителям предприятия.
  - ИС соответствует потребностям компаний, бизнесу компаний, а также согласована с организационной и финансовой структурой компаний, культурой компании и в целом интегрирована в деятельность предприятия.
  - ИС содействует информационным потокам между подразделениями внутри компании и поддержке связей с другими организациями.

#### **1.1.2. Автоматизированные системы управления**

Автоматизированные системы управления, ставшие основой информатизации и прообразами современных информационных

систем, в настоящее время уже не удовлетворяют потребности компаний. Многие АСУ сегодня подвержены трансформациям, которые необходимо отчетливо представлять для коррекции развития и преобразования АСУ в целях соответствия потребностям бизнеса.

В силу того что автоматизированные системы управления берут свое начало еще в середине XX в., приведем определение, данное во время существования Советского Союза.

---

**Автоматизированная система управления** — это система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных задач.

---

Иными словами, АСУ использовались для быстрого и качественного выполнения государственного плана, и этой потребности были подчинены все элементы АСУ. Многие АСУ были унифицированы и проектировались по общеотраслевым стандартам, однако сейчас каждое предприятие полностью автономно в создании собственной информационной системы.

### 1.1.3. Корпоративные информационные системы (КИС)

Логично говорить о том, что один из путей развития АСУ — это их преобразование в **корпоративные информационные системы** (КИС), ведь на первый взгляд КИС и АСУ имеют много общего. Однако на практике это далеко не так. Первоначально (современная формулировка будет приведена чуть ниже) определение КИС было сформулировано следующим образом.

---

**Корпоративная ИС** — информационная система, объединяющая современные информационные технологии и стратегию бизнес-развития предприятия для реализации управленческого процесса.

---

Бизнес начинает построение КИС в случаях, когда не остается иных способов управления материальными, финансовыми и иными потоками в организации и за ее пределами. Сохранение своего положения в конкурентной рыночной борьбе — основная цель, поэтому создание КИС зависит только от потребности менеджмента компании в создании эффективно работающей организации.

Назначение КИС — поддержка «регулярного менеджмента», и если таковой не распространен на предприятии, то внедрение КИС в его деятельность нецелесообразно. Создание информационной системы предприятия требует многих управленческих решений: необходимо сформировать специальную группу по созданию КИС, выделить для нее все необходимые ресурсы, предоставить полномочия и доступ к данным, а также обеспечить ее поддержку.

Уровень КИС определяется не только наличием определенной функциональности (например, по оперативному учету и аналитике состояния бизнеса), но и собственными научно-исследовательскими алгоритмами. При помощи новейших КИС зачастую можно решать задачи, невыполнимые традиционными «человеческими» средствами, и все чаще весомое преимущество дают решения, основанные на открытых платформах. При этом важно, что технологии для КИС должны подбираться таким образом, чтобы извлекать пользу из собранных исторических данных, а не уничтожать их. На практике реализация КИС не требует серьезных инноваций сферы ИТ. Куда важнее интеграция программно-аппаратных средств, а также надежность и качество их работы.

Современное определение КИС звучит следующим образом.

---

**Корпоративная информационная система (КИС)** — система, охватывающая все основные области хозяйственной деятельности предприятия, а также общий процесс управления.

---

Соответственно, среди пользователей КИС возможно выделить две группы сотрудников: первая использует получаемую информацию для принятия управленческих решений (а значит, основная функция КИС для них — консолидация и аналитика данных), а вторая использует данные при реализации бизнес-процессов предприятия.

Идеальной ситуацией является создание единой КИС, которая соответствовала бы интересам всех сотрудников, служб и подразделений предприятия. Однако это многие годы остается и останется теорией. На практике предприятия имеют множество разрозненных информационных систем, часто не связанных между собой и покрывающих лишь отдельные области деятельности предприятия (производство, продажи, логистику, финансы и пр.). Подобная ситуация получила название *лоскутной автоматизации*, когда часть задач избыточно покрывается несколькими системами или не автоматизируется совсем. Разумеется, это не означает необходимость повсеместной автоматизации, которая далеко не всегда оправдывает себя. Однако следует учитывать общие тенденции, которые будут рассмотрены далее и которые важны для принятия корректных бизнес-решений о выделении бюджета на автоматизацию. Более того, любые подобные решения должны приниматься совместно специалистами как со стороны бизнеса, так и со стороны ИТ-подразделений, в результате объективной оценки достоинств и недостатков каждого из вариантов ИС.

Говоря об истории информационных систем, следует рассмотреть два основных аспекта:

- предпосылки развития способов обработки данных и эволюцию автоматизированных систем управления;

- развитие функциональности и рост широты применения корпоративных ИС.

#### **1.1.4. Цифровая платформа**

Цифровая платформа — новый тип информационной системы, который в настоящее время набирает популярность и становится все более распространенным. Согласно отчету Московской школы управления Сколково, цифровая платформа — это «сложная информационная система, обеспечивающая специфический способ выполнения определенной функции и открытая для использования клиентами и партнерами, включая разработчиков приложений, мерчантов и агентов. Платформа может быть использована напрямую или же через приложения, созданные на ее основе ее владельцем или третьими лицами».

Существуют и другие определения цифровой платформы. Так, по версии компании Accenture, цифровая платформа — это «группа технологий, которые используются в качестве основы, обеспечивающей создание конкретизированной и специализированной системы цифрового взаимодействия».

Исследователи из Massachusetts Institute of Technology (MIT) считают, что цифровая платформа — это обеспеченная высокими технологиями бизнес-модель, которая создает стоимость, облегчая обмены между двумя и более группами участников.

Существует множество других определений, однако фактически цифровая платформа представляет собой некоторую совокупность цифровых технологий, продуктов и услуг, формирующие некоторый фундамент, на базе которого внешние компании могут создавать их собственные дополнительные продукты, технологии или услуги.

Понятие цифровой платформы может использоваться и для обозначения особого типа бизнес-модели, которая целиком базируется на информационных технологиях и получает прибыль за счет обмена между двумя или более независимыми группами участников. Платформа предоставляет единое информационное пространство, в рамках которого могут без посредников взаимодействовать различные производители и потребители. Также они дают возможность различным компаниям делиться информацией и таким образом существенно улучшать сотрудничество и создавать инновационные продукты или услуги.

### **1.2. Этапы развития информационных технологий**

Развитие информационных систем в том или ином виде можно проследить от начала XX в., причем общую логику развития ИС всегда определяли два тесно связанных между собой тренда:

1) недостаточная производительность технического обеспечения. К техническому обеспечению относится не только само вычислительное оборудование, но также многочисленные периферийные устройства, системы хранения данных, ЦОДы, сервисные площадки, локальные вычислительные сети, корпоративные сети передачи данных, телефония и пр.;

2) потребность сделать основные вычисления как можно более децентрализованными. По мере эволюции информационных систем увеличивается и число их пользователей, а вместе с тем снижаются требования к их квалификации.

Рассматривать развитие информационных систем в отрыве от развития аппаратного обеспечения было бы недостаточно информативно. При рассмотрении в совокупности можно выделить пять этапов развития, каждый из которых характеризуется особым восприятием технического обеспечения и информационных систем, а также некоторыми наиболее значимыми функциями.

Таблица 1.1

#### Этапы развития технического обеспечения и информационных систем

№ п/п	Этап	Восприятие	Функция
1	Майнфреймы и первые вычислители	Вычислитель	Программные вычисления (калькулятор)
2	Персональные компьютеры и «кусочная автоматизация»	Исполнитель	Автоматизация отдельных бизнес-функций
3	Интегрированные информационные системы и массовая автоматизация	Помощник и контроллер	Интегрированные ИС, учет, контроль, автоматизация
4	Повсеместная автоматизация деятельности, гибкие информационные системы и технологии	Партнер	Оптимизация, Реинжиниринг, Инновации
5	Цифровая трансформация бизнеса	Инноватор	Трансформация бизнеса, изменение бизнес-моделей

Далее подробно рассмотрим выделенные этапы.

1. *Первые вычислители и майнфреймы*. Слабые аппаратные возможности до конца XIX в. делали крайне затруднительной обработку больших объемов информации, однако уже в 1890 г. Герман Холлерит создал электрическую табуляционную систему, которая была успешно использована при переписи населения. Спустя шесть лет Г. Холлерит создал компанию TMC (Tabulating Machine Company), которая после продажи в 1911 г. вошла в конгломерат C-T-R Чарльза Флинта. А уже в 1924 г. компания была переименована в IBM.

В начале XX в. остро обозначилась потребность предприятий в получении информации в заданной форме. Несмотря на то, что полноценное техническое обеспечение, способное удовлетворить эту потребность, фактически отсутствовало, стали появляться многочисленные теоретические работы по управлению предприятиями. Наиболее значимым можно назвать труд Ф. У. Тейлора «Управление фабрикой». К 1924 г. некоторые виды технического обеспечения применялись для обработки потока расчетных документов.

Примерно до второй половины XX в., в силу неразвитости вычислительной техники, основной задачей было предоставление информации и обеспечение дистанционной коммуникации. Задачи решались вручную, написанием и отправкой писем. А к середине 1940-х гг. уже были разработаны первые машины электромеханического типа, благодаря которым информацию стало возможно создавать при помощи печатной машинки, записывать на диктофон, передавать по телефону.

Первый компьютер в современном понимании этого слова появился в начале 1940-х гг. В 1941 г. был разработан, а в 1944 г. после успешного прохождения тестов перенесен в Гарвардский университет компьютер Марк I от компании IBM. Производительность Марк I была не велика: фактически он заменял труд 10—30 операторов, считывая операции с перфорированной бумажной ленты. При этом Марк I весил около 4,5 тонн, состоял из порядка 800 тыс. деталей и занимал площадь в несколько десятков квадратных метров. Несмотря на революционный характер компьютера Марк I, и он, и большинство его последователей так и останутся в роли вычислителя.

В 1952 г. свет увидел первый в мире большой компьютер на лампах IBM 701. IBM 701 стал первым коммерческим компьютером, который нашел широкое применение в бизнесе. Применение компьютера в среднем обходилось компании в 12 000—15 000 долл. в месяц. Появившийся спустя два года IBM 704 стал первым массовым компьютером с аппаратной поддержкой вычислений с плавающей точкой. Именно для этой модели были разработаны языки программирования Фортран и Лисп, потому что производительность IBM 704 была существенно выше, чем у предшественников.

IBM 701 и его последователи использовались прежде всего для автоматизации рутинных функций и оптимизации подготовки отчетности. В качестве основных информационных систем этого периода использовались системы экономической отчетности и ИС инженерных данных.

К середине 1960-х гг. роль информации в представлении компаний возросла, и ее начали воспринимать как один из ключевых ресурсов предприятия, организации, региона и общества. Появившиеся первые операционные системы и дисковые технологии спо-

существовали популяризации информационных технологий. Начали стремительно развиваться языки программирования. В ответ на запросы бизнеса была представлена первая компьютерная система класса мейнфреймов, которой стал IBM System/360 в 1963 г. Разработка IBM System/360 стала самым дорогим и рискованным инвестиционным проектом в истории частного бизнеса, однако в результате компьютеры оказались доступными для организаций всех типов и размеров.

Первыми автоматизированными информационными системами стали системы обработки отчетных документов на базе электромеханических бухгалтерских машин. В первую очередь подобная программино-аппаратная обработка данных применялась военным и космическим комплексами для управления ракетами и космическими объектами, а также в масштабах предприятия для получения, обработки и передачи статистической информации и ведения учета. Так как все эти данные имели огромный для того времени объем, системам хранения и базам данных стало уделяться пристальное внимание, причем зачастую использовались термины «база целей» и «база знаний» для более точной постановки задач.

Позже на первый план вышла периодическая отчетность по выбранным параметрам, в частности, на «многофункциональном» компьютерном оборудовании широкого назначения. Однако подобная мультифункциональность также имела свои пределы, так как любое изменение предметной области приводило к изменению структуры данных — а значит, к изменению программ и значительным затратам.

В то время основное внимание уделялось двум основным направлениям: автоматизированным системам управления и системам поддержки математических расчетов.

При этом снова напомним, что речь идет о 1950—1960-х гг., а значит, подобный подход к обеспечению принимающих решения людей необходимой информацией немедленно принял системный и повсеместный характер.

Классификация автоматизированных систем управления включает:

- АСУП (уровень предприятий):
  - АСУ объединения,
  - АСУ предприятий/организаций,
  - АСУ производств,
- АСУ цехов/участков;
- ОАСУ (уровень отраслей);
- РАСУ (уровень регионов/республик);
- ОГАС (общегосударственный уровень).

Возможность ввода информации чаще всего предоставлялась пользователям в виде заполняемых форм, сгруппированных относи-

тельно прикладных задач, на решение которых была нацелена система. Иногда пользователь даже не имел прямого контакта с системой, так как и предварительная обработка данных, и их последующий ввод в систему производились специализированным персоналом. По сути, АИС того времени являлись документально-фактографическими информационно-поисковыми системами, однако подобная терминология не получила распространения.

Появление IBM System/360 ознаменовало собой начало эры мейнфреймов, многие из которых до сих пор используются предприятиями по всему миру. В конце 1980-х гг. компания Gartner прогнозировала отключения последнего мейнфрейма в 1993 г., однако в настоящий момент не только используются старые модели мейнфреймов, но и активно разрабатываются новые.

В настоящий момент мейнфреймы наиболее интересны с точки зрения потенциала виртуализации: создания виртуальных машин, виртуальных рабочих мест и пр. При этом мейнфрейм обладает множеством преимуществ, среди которых среднее время наработки на отказ более 12—15 лет, возможность горячей замены всех элементов, повышенная устойчивость, большие возможности масштабирования, рабочая нагрузка в 80—85 % от пиковой, возможность дублирования компонентов и пр.

2. *Персональные компьютеры и «лоскутная автоматизация».* Серьезный качественный сдвиг наметился в 1981 г., когда IBM развернула производство персональных компьютеров IBM PC, работающих под управлением операционной системы DOS, разработанной специалистами фирмы Microsoft. Компьютеры IBM PC пользовались коммерческим успехом, и многие фирмы-производители электронной техники наладили выпуск «клонов» IBM PC. Значительное влияние на становление и развитие периода персональных компьютеров оказало изобретение в 1971 г. первого гибкого диска 8" от IBM.

С появлением персональных ЭВМ в начале 1980-х гг. идея построения АСУ была скорректирована в сторону движения к распределению вычислительных ресурсов и децентрализации управления (в дальнейшем становясь новым этапом в ИТ организационного управления в системах поддержки принятия решений). ИС предприятий становятся средством управления производством, поддерживающим и ускоряющим процесс принятия решений (постановка которых, однако, проводилась на этапе создания системы и впоследствии практически не корректировалась). Соответственно, системы становились одновременно узкоспециализированными и более интеллектуальными. С инфраструктурной точки зрения нагрузка на централизованные вычислительные ресурсы снижалась, уступая место необходимости решения крупных долгосрочных задач.

В 1982 г. появился компьютер ZX Spectrum с графическим интерфейсом, что еще более повысило доступность компьютеров и сни-

зило порог требований к квалификации для их использования. Еще спустя два года свет увидел первый Apple Macintosh 128K с мышью и полностью графическим интерфейсом.

С появлением интерактивных дисплеев технологии становились более приближенными к рядовым пользователям, что снизило требования к их квалификации и предоставило дополнительные возможности применения в офисах компаний сразу на нескольких организационных уровнях.

Значительным прорывом стало распространение принципов дружественных (англ. user-friendly) интерфейсов: графических интерфейсов, всплывающих подсказок и разделов помощи, а также систем быстрой разработки RAD и CASE-средств автопроектирования ИС, что позволило относительно рядовым пользователям самостоятельно создавать системы. Тем не менее, подобные знания получить было не так просто, в то время как навыки работы с прикладными программами и знание основ ПК на уровне пользователя стало обязательным требованием для многих специалистов (что стало проблемой в силу общей инертности сотрудников и их нежелания менять стиль работы). Однако в любом случае технологии не стояли на месте, и возможности текстовой обработки значительно популяризовали работу с ПК, которые продолжили (и продолжают!) стремительно развиваться. Они облегчили многие рутинные ежедневные операции и позволили расширить функциональность вплоть до систем бизнес-аналитики, предоставлявших информацию по мере появления проблем.

На данном этапе производительности технического обеспечения стало достаточно для решения более сложных задач. Компьютеры стали применяться для информационного моделирования, управления, прогнозирования. Стали появляться системы бизнес-аналитики и первые офисные информационные системы. Информационные системы стали помощниками при выработке рационального решения.

Информационные системы данного этапа позволили автоматизировать задачи, которые ранее выполнялись в «бумажном» формате. Фактически бумажная работа была заменена на электронную, что серьезным образом сказалось на производительности труда и сроках выполнения целого ряда задач, однако в целом компьютеры и информационные системы по-прежнему не имели определяющего значения в развитии предприятия. При этом имела место «лоскутная автоматизация», при которой различные виды деятельности автоматизировались отдельно друг от друга. Для этого использовались не связанные или слабо связанные друг с другом компоненты программного и аппаратного обеспечения.

3. *Интегрированные информационные системы и массовая автоматизация.* На данном этапе программное и аппаратное обеспе-

чение перешло от роли исполнителя к роли помощника и контролера. Наметился сдвиг фокуса внимания с развития аппаратного обеспечения на развитие ПО, что в конечном счете закономерно привело к достижению нового уровня недостаточности вычислительной мощности технического обеспечения.

На данном этапе широкое распространение получила клиент-серверная архитектура, позволяющая распределить функции вычислительной системы между персональными компьютерами пользователей. При этом хранение данных обеспечивалось на защищенном сервере, поддерживающем многопользовательскую работу. Переход к клиент-серверной архитектуре стал важным шагом в децентрализации основных вычислений. В отличие от дорогих мейнфреймов, доступ к которым имело ограниченное число сотрудников, клиент-серверная архитектура вкупе с персональными компьютерами позволили обеспечить вычислительными мощностями огромное число сотрудников предприятий.

Несмотря на очевидные положительные стороны, двухуровневая архитектура «клиент-сервер» имела ряд недостатков, многие из которых потребовали дальнейшего развития технологий. Сложное администрирование, высокая стоимость оборудования, недостаточная функциональность клиентского ПО и потенциальная возможность «обрушить» вычислительную сеть привели к появлению многоуровневой архитектуры клиент-сервер.

Многоуровневая архитектура клиент-сервер позволила добиться высокой степени масштабирования и конфигурируемости, а также обеспечить высокую безопасность и надежность. Требования к автоматизированным рабочим местам существенно снизились, а администрирование клиентского ПО упростилось.

К недостаткам многоуровневой архитектуры клиент-сервер традиционно относятся высокие требования к производительности серверов приложений и серверам базы данных и высокие требования к скорости канала (сети) между сервером базы данных и сервисами приложений.

По мере развития и распространения локальных вычислительных сетей и Интернета популярность стала набирать архитектура веб-приложений. Архитектура веб-приложений вновь смягчила требования к рабочим местам пользователей, также обеспечило централизованное хранение данных и позволило сохранять данные при смене ПК. При этом архитектура не требовала наличия клиентского ПО, а обновление веб-приложений обычно не создавало серьезных проблем.

Из минусов архитектуры веб-приложений следует упомянуть возможную низкую защищенность за счет широких возможностей для злоумышленников по взлому браузера, недоступность работы браузера при отсутствии работоспособности сервера или каналов

связи, меньше возможностей по использованию элементов по визуализации информации. Также возможна низкая скорость работы при нестабильном сетевом подключении.

4. *Повсеместная автоматизация деятельности, гибкие информационные системы и технологии.* Недостаточная производительность технического обеспечения стала основным катализатором развития на данном этапе. Растущие требования к функциональности информационных систем требовали все более мощного аппаратного обеспечения, стоимость которого слишком высока для удовлетворения потребностей большинства компаний путем закупки аппаратного обеспечения подо все нужды.

В результате основным направлением развития на данном этапе стало постепенное распространение технологий виртуализации. При помощи технологий виртуализации стало возможным представлять вычислительные ресурсы или их логическое объединение, абстрагированное от аппаратной реализации и обеспечивающее логическую изоляцию вычислительных процессов.

К основным видам виртуализации в настоящий момент относятся:

- виртуализация платформ, позволяющая создавать виртуальные машины и эмуляторы платформ, а также обеспечивать виртуализацию операционных систем, виртуализацию приложений;
- виртуализация ресурсов, обеспечивающая объединение и агрегацию ресурсов, распределенные вычисления, кластеризацию компьютеров, разделение ресурсов.

Фактически виртуализация позволила разделить вычислительные процессы и ресурсы. Современные архитектуры виртуальных систем позволяют более эффективно использовать аппаратные ресурсы, обеспечивают совместимость, позволяют изолировать временноное окружение, создавать гибкие аппаратные конфигурации, виртуальные образы не имеющихся в наличии устройств, обеспечивают одновременный запуск виртуальных машин внутри виртуальной сети. Архитектура виртуальных систем кардинальным образом увеличила мобильность организации и ее сотрудников, а также повысила степень управляемости.

Тем не менее, не каждое устройство может быть эмулировано, поэтому не все устройства могут быть виртуализированы. При этом некоторые платформы виртуализации требовательны к конкретному аппаратному обеспечению, а наиболее популярные сегодня платформы виртуализации достаточно дорогие. Несмотря на это виртуализация позволяет удовлетворить потребности бизнеса в вычислительных мощностях в настоящий момент без закупки дорогостоящего оборудования.

Ответом на потребность в децентрализации на данном этапе стали облачные и, в меньшей степени за счет недостаточной распро-

страненности на текущий момент, туманные вычисления. Туманные вычисления позволяют предоставлять «по запросу» серверные мощности, обеспечивать «по запросу» проведение веб-конференций, обеспечивать хранение и резервное копирование данных, а при необходимости — и аварийное восстановление.

Облачные модели обслуживания, существующие в настоящий момент, позволяют получить «по запросу» не только вычислительные мощности. На сегодняшний день широко распространены сервисные модели «инфраструктура как сервис», «платформа как сервис», «программное обеспечение как сервис», «безопасность как сервис», «рабочий стол как сервис» и многие другие.

Туманные вычисления позволяют добиться еще большей степени децентрализации за счет обработки данных в непосредственной близости с источником данных без передачи в крупные дата-центры. Для обработки могут использоваться небольшие дата-центры, фактически представляющие собой вынос крупных дата-центров в устройство, что позволяет приблизиться к конечным пользователям и повысить мобильность.

### Цифровая трансформация бизнеса

Сегодня наблюдается становление и окончательное формирование пятого этапа — цифровой трансформации существующих организаций. На данном этапе информационные технологии становятся инноватором, обеспечивающим конкурентное развитие предприятия и получение добавленной стоимости. Предприятия, которые изначально не были цифровыми и применяли ИТ в минимальном объеме, производят цифровизацию собственной деятельности для максимально эффективного использования информационных технологий.

Серьезно изменяются бизнес-модели. Помимо цифровой трансформации, которая наблюдается внутри каждого сектора и затрагивает контрагентов предприятия, серьезно изменяются требования потребителей, их поведенческие модели, привычки и ценности. Предприятия вынуждены адаптироваться к потребностям гиперподключенного пользователя, имеющего несколько устройств с выходом в интернет и предъявляющего требования, немыслимые еще 10—20 лет назад.

Вместе с этим формируются цифровые платформы, позволяющие сформировать экосистему взаимодействия многочисленных участников цепочки создания ценности. В рамках единой цифровой платформы взаимодействуют поставщики, разработчики, производители, консультанты, аналитики, конечные потребители и иные участники рынка. Компания, владеющая цифровой платформой, становится модератором экосистемы — своеобразным центром виртуального предприятия, охватывающего полную цепочку создания

ценности, начиная с поставок и заканчивая производством, маркетингом и обслуживанием.

При этом возможности технологического обеспечения практически исчерпаны в настоящий момент. Виртуализация и облачные технологии позволили на время отодвинуть потолок их применения в интересах бизнеса, однако в ближайшее время технологии, основанные на полупроводниках, практически исчерпали себя. В качестве одного из возможных ответов на данный вызов является активная разработка на сегодняшний день квантовых компьютеров (IBM).

### **1.3. Развитие функциональности ИС КИС и управления процессами предприятия**

Ниже рассмотрим ключевые аспекты изменения функциональных возможностей корпоративных ИС в динамике за последние десятилетия для формирования более целостного понимания текущих процессов и тенденций их развития (табл. 1.2).

*Таблица 1.2  
История изменения функциональных возможностей КИС*

Этап	Период	Названия систем / модулей	Развитие функциональности
Мейнфреймы и первые вычислители	XIX в. — 1950-е гг.	—	Принципы организации производства, заложенные Ф. У. Тейлором (F. W. Taylor, 1856—1915), временно без аппаратно-программной реализации (в силу «неизобретенности» и неразвитости вычислительной техники). Принципы организации производства носили исключительно теоретический характер в силу того, что вычислительная техника тех времен была еще недостаточно развита. Однако именно на основе этих работ Тейлора, применимых для любой сферы деятельности, Генри Форд позднее на практике успешно организовал производство автомобилей. Первые работы стали основой систематизации знаний об управлении предприятиями
	1950—1964 гг.	IC / IM (MRP 0)	Оптимизация складских запасов (Inventory Control/Management, IC/IM), расчет и планирование

Продолжение табл. 1.2

Этап	Период	Названия систем/ модулей	Развитие функциональности
			потребностей в материалах (Material Requirements Planning, условно MRP 0) по Дж. Орлики (J. Orlicky) и О. Уайту (O. Wight). Они впервые затронули тему более эффективной работы склада, оптимизации его запасов за счет грамотного и своевременного планирования необходимости в сырье и материалах (соответственно, во избежание простаивания или переполнения склада)
	1965—1974 гг.	MRP	Планирование потребностей в материалах (Material Requirements Planning, MRP или MRP 1), в том числе по замкнутому циклу (Closed Loop MRP), включающее составление производственной программы и ее контроль на цеховом уровне по Дж. Дж. Миллеру (J. G. Miller) и Л. Дж. Спраг (L. G. Sprague)
	1975—1980 гг.	MRP II	Планирование производственных ресурсов на основе данных, полученных от поставщиков и потребителей, включая прогнозирование, планирование (в том числе загрузки производственных мощностей) и контроль за производством
Персональные компьютеры и «кусочная автоматизация»	1981—1985 гг.	CALS	Добавление к MRP 2 идеологии «точно в срок» (Just-In-Time, JIT), элементов системы «канбан» (kanji + ban — визуальных карточек) по Ш. Шинго (S. Shingo) и Т. Оно (T. Ohno), оптимальной технологии производства (Optimized Production Technology, OPT) и оптимизации «узких мест» по Э. Голдратту (E. Goldratt); автоматизированная поддержка поставок (Computer-Aided Logistic Support, CALS)

Продолжение табл. 1.2

Этап	Период	Названия систем / модулей	Развитие функциональности
Интегрированные информационные системы и массовая автоматизация	1986—1990 гг.	ERP	Планирование (всех) ресурсов предприятия (Enterprise Resources Planning, ERP), в том числе человеческих (Human Resources Management, HRM) и финансовых (Financial Resources Planning, FRP)
	1991—1996 гг.	SCM, CALS II	Добавление к ERP планирования ресурсов для распределения (Distribution Resources Planning, DRP); управление цепочками поставок (Supply Chain Management, SCM), позволяющее направлять и контролировать движение материальных и информационных потоков от поставщика к потребителю, по всей цепочке; непрерывную поддержку поставок и жизненного цикла (Continuous Acquisition and Lifecycle Support, CALS 2). Анализ рынка, поиск деловых партнеров, создание товаров и услуг, оптимизация производства — лишь некоторые из важных на данном этапе функций информационных систем, толчком к развитию которых стало стремительное распространение персональных компьютеров
Повсеместная автоматизация деятельности, гибкие информационные системы и технологии	1997—2000 гг.	CSRP	Планирование ресурсов, синхронизированное с потребителями (Customer-Synchronized Resources Planning, CSRP): интеграция потребителей и связанных с ними подразделений с основными плановыми и производственными подразделениями, интеграция собственных информационных систем с приложениями потребителей, планирование заказов потребителей
	2000 г. — н. в.	ERP II (SRM/SCM)	Управление внутренними ресурсами и внешними связями. Развитие модулей SCM (управление цепочками

Этап	Период	Названия систем / модулей	Развитие функциональности
			поставок) и CRM (управление взаимоотношениями с клиентами), отвечающими за оптимизацию внешних связей предприятия
Цифровая трансформация бизнеса	2010 — н. в.	Цифровая платформа	Охват всей цепочки создания ценности. Возможность совместного участия в создании ценности множества контрагентов. Возможность формирования виртуальных предприятий и виртуальных цепочек создания ценности. Единое информационное пространство для взаимодействия. Работа с едиными данными сразу нескольких предприятий.

## 1.4. Существующие классификации ИС

Для формирования высокоуровневого представления об охватываемых информационными системами областях рассмотрим их классификацию по различным критериям с указанием в некоторых случаях взаимосвязи между тем или иным классификационным критерием и особенностями жизненного цикла системы. Разумеется, число и последовательность этапов жизненного цикла всегда будут постоянными, однако длительность каждой фазы, трудоемкость внедрения и перспективы последующей эксплуатации могут значительно различаться.

### 1.4.1. Архитектура

Для использования подобной классификации необходимо определить функции трех основных структурных элементов ИС (табл. 1.3).

**1. Базы данных.** Поддержка хранения данных и обеспечения доступа к ним по запросу.

**2. Системы управления.** Определение правил хранения, выборки, изменения и удаления данных.

**3. Клиентские приложения.** Поддержка взаимодействия с пользователем, причем для всех платформ и устройств (мобильные телефоны, планшеты, ПК).

**Особенности ЖЦ.** В зависимости от архитектуры системы будут значительно различаться по процессу проектирования и принципам последующей интеграции с другими решениями.

Таблица 1.3

## Классификация ИС по архитектуре

Тип систем	Базы данных	Системы управления	Клиентские приложения
Настольные	Все элементы на одном компьютере		
Распределенные:	Все элементы на разных компьютерах		
файл-серверные	На файловом сервере		На рабочих станциях
Клиент-серверные:	На сервере		На рабочих станциях
двузвенные ( <i>two-tier</i> )	Сервер баз данных включает и БД, и системы управления (СУБД)		Клиентские приложения обращаются к серверу напрямую
многозвенные ( <i>multi-tier</i> )	Существуют серверы баз данных (поддерживают управление данными) и в качестве дополнительных звеньев серверы приложений (поддержка бизнес-логики). Пример — современные веб-приложения		Клиентские приложения обращаются к серверу через серверы приложений

## 1.4.2. Масштабность

В зависимости от числа потенциальных конечных пользователей выделяются три основные категории (табл. 1.4).

Таблица 1.4

## Классификация ИС по масштабу

Категория	Характеристика
Персональная	«Покрывает» информационные потребности одного пользователя
Групповая	Подходят для коллективного использования, однако недостаточно производительны для использования в масштабах компании
Корпоративная	Охватывают большую часть функций предприятия и разные уровни пользователей в рамках компании

**Особенности ЖЦ.** Жизненный цикл систем, в работе с которыми задействован не один человек, а несколько подразделений или даже филиалов компаний, часто будет более продолжительным в силу сложности отказа от их эксплуатации. С точки зрения фаз анализа и проектирования трудоемкость будет более высокой при обследовании всего предприятия и сборе требований многих подразделений, а при внедрении может потребоваться большее число настроек и последующих модификаций.

### 1.4.3. Степень структурированности задач и характер обработки данных

Сложность применяемых алгоритмов зависит от сложности задачи, которая определяется полнотой и степенью обработки имеющихся данных. Различные функции ИС требуют наличия алгоритмов разной степени сложности и, соответственно, с различной трудоемкостью их создания (табл. 1.5).

Таблица 1.5

**Классификация ИС по степени структурированности задач**

Алгоритмы	Функции и особенности ИС	Примеры
Для структурированных задач	Содержание задачи представляется в виде полностью автоматизирующего ее алгоритма, так как все необходимые для анализа данные и их взаимосвязи известны. Системы часто могут носить лишь информационно-справочный характер, и для них наиболее важна скорость и точность поиска и предоставления необходимых данных	Расчет заработной платы — стандартный алгоритм, выполняющийся ежемесячно. Регистратура больницы. База данных отдела управления персоналом
Для частично структурированных или неструктурированных задач	Основная цель — обработка информации и ее выдача в наиболее удобном для дальнейшего принятия решения человеком виде, так как полный набор исходных элементов и связи между ними неизвестны. В данном случае приоритет отдается возможностям аналитики данных и визуализации результатов в удобном для быстрого принятия решений формате	
Создающие управленческие отчеты (репортинговые)	Оrientированы в основном на поиск, сортировку, фильтрацию данных	Система предоставления аналитики по предыдущим периодам и прогнозирования будущих показателей на их основе
Разрабатывающие альтернативные решения:	Предоставляют несколько вариантов для выбора человеком одного из них	
модельные	Результат — статистические, финансовые модели, требующие участия человека для ввода данных	Проведение анализа чувствительности, динамического изменения параметров. Например, в микробиологии такая система может использоваться для выбора оптимальной концентрации реагента

Окончание табл. 1.5

Алгоритмы	Функции и особенности ИС	Примеры
экспертные	Реализация двух уровней: на первом создаются несколько классов управленческих решений, на втором генерируются альтернативы на основе базы знаний, содержащей правила соответствующей генерации альтернатив на основе имеющихся данных	Примером может быть система анализа геологических данных для определения местоположения залежей руды или система прогнозирования течения заболевания на основе анализа множества входных данных о пациенте, базы историй болезней и динамики их развития для других пациентов

С одной стороны, с внедрением семантической паутины и развитием алгоритмов экспертных систем границы между этими классами ИС станут размытыми<sup>1</sup>. Одна из основных тенденций будущего информационных систем — совершенствование алгоритмов. Более того, с развитием технологий растут и требования пользователя к информационным системам. Если раньше ИС строго разграничивались по функциональности, то теперь становится необходимой комбинация различных их видов, так как пользователям хочется получать максимальное количество полезной информации. С другой стороны, в ближайшее время будет сохраняться потребность в простых ИС, выполняющих простые операции. Потребность сохранится даже несмотря на постепенный уход от понятия «неструктурированных и частично структурированных ИС».

**Особенности ЖЦ.** Сложность алгоритмов обработки данных отражается как на сроках и стоимости разработки и внедрения проектного решения, так и на продолжительности этапов его эксплуатации и модернизации. Так, если функциональность ИС охватывает значительное количество областей бизнеса либо же наиболее критические для компании управленческие аспекты и предоставляет критичную для принятия решений информацию, с большой долей вероятности отказаться от ее дальнейшей эксплуатации и перейти на другое решение будет достаточно сложно. Однако даже в этом случае решающую роль будет играть соответствие решения потребностям бизнеса, и никакая сложность решаемых задач или трудоемкость внедрения не сможет служить мотивацией продолжения работы с системой, не выполняющей поставленных перед ней изначально целей.

<sup>1</sup> Thierauf R. J. Executive Information System: A Guide for Senior Management and MIS Professionals. N. Y. : Quorum Books, 1991.

#### 1.4.4. Режим работы

В зависимости от числа потенциальных конечных пользователей выделяются три основные категории<sup>1</sup> (табл. 1.6).

Таблица 1.6

Классификация ИС по режиму работы

Категория	Характеристика
Пакетные	Пользователь не имеет прямого общения с системой, и поставленные задачи не несут оперативный характер
Диалоговые	Быстрая реакция на запросы пользователя, но зато программа не работает автономно от человека, как это реализовано в пакетном режиме
Смешанные	Предоставление по возможности максимальной автономности, скорости и качества обработки данных за счет смены режимов в зависимости от исходных условий

**Особенности ЖЦ.** С точки зрения анализа и проектирования трудоемкость этапов жизненного цикла будет разной для различных режимов работы, так как в случае относительно автономных от пользователя систем необходимо еще на этапе проектирования предусмотреть большое число ситуаций для самостоятельного принятия решения системой. В то же время для пакетной обработки более важен интерфейс взаимодействия с пользователем и его удобство для оперативного ввода и получения данных. Также, разумеется, с возрастанием степени автономности и скорости работы процесс разработки может стать более длительным из-за применения более сложных алгоритмов/технологий.

#### 1.4.5. Процессы и уровни управления

В данной категории<sup>2</sup> в зависимости от уровней принятия решения и вида процессов, с ними сопряженных, выделяют шесть основных классов систем на четырех уровнях управления<sup>3</sup> (табл. 1.7).

При этом информационные системы в данной классификации можно и далее группировать по функциональности (табл. 1.8). К категориям такой классификации могут относиться продажи, маркетинг, производство, HR, финансы, бухгалтерский учет и т. п.

<sup>1</sup> Галактионов В. Системная архитектура и ее место в архитектуре предприятия // Директор Информационной Службы. 20.05.2002. URL: <http://www.osp.ru/cio/2002/05/172142> (дата обращения: 15.05.2020).

<sup>2</sup> Макарова Н. В., Волков В. Б. Информатика. СПб. : Питер, 2011. С. 576.

<sup>3</sup> URL: <http://dilbert.iiml.ac.in/~bhasker/mis/week2session4.pdf>.

Таблица 1.7

## Классификация ИС по уровням управления

Уровень принятия решений и основные пользователи	Класс	Задачи
Стратегические, топ-менеджмент	ESS (Executive Support Systems), исполнительные ИС	Визуализация, интерактивность, моделирование
Управленческие, менеджеры среднего звена	MIS (Management Information Systems), управляющие ИС	Big data-аналитика, представление итоговых данных
	DSS (Decision Support Systems), системы поддержки принятия решений (СППР)	Интерактивная бизнес-аналитика и моделирование
Системы уровня знаний, менеджеры нижнего звена и специалисты	KWS (Knowledge Work Systems), системы знаний	Моделирование, работа с базами знаний и техническими данными
	OAS (Office Automation Systems), системы автоматизации делопроизводства	Планирование и управление
Эксплуатационные/операционные, специалисты	TPS (Transaction Processing Systems), системы диалоговой обработки запросов	Исполнение транзакций и представление их результатов

Таблица 1.8

## Классификация ИС по категориям пользователей

Уровень принятия решений и основные пользователи	Тип ИС	Функциональность	
		производство	финансы и бухгалтерский учет
Стратегические, топ-менеджмент	ESS (Executive Support Systems)	Операционный план на 2—5 лет	Составление инвестиционного бюджета
Управленческие, менеджеры среднего звена	MIS (Management Information Systems)	Планирование, контроль инвентаря	Ежегодный бюджет
	DSS (Decision Support Systems)	Планирование производства	Анализ затрат и рентабельности
Системы уровня знаний, менеджеры нижнего звена, специалисты	KWS (Knowledge Work Systems)	APM проектировщика	Управленческие рабочие станции
	OAS (Office Automation Systems)	Текстовые редакторы, электронные таблицы	

Уровень принятия решений и основные пользователи	Тип ИС	Функциональность	
		производство	финансы и бухгалтерский учет
Эксплуатационные и операционные, специалисты	TPS (Transaction Processing Systems)	Движение материалов	Расчет кредиторской задолженности

Технологическому прогрессу еще далеко до того момента, когда менеджеры и руководители не будут востребованы. Машины сегодня выполняют большую часть рутинной работы, но принимать окончательные решения они не способны. Несмотря на то что семантическая паутина позволит упростить процесс принятия решений, потребность в менеджерах и специалистах упадет, но спрос на них не исчезнет. Следовательно, подобная классификация в отличие от многих других не потеряет своей актуальности, хотя и может видоизмениться.

**Особенности ЖЦ.** В целом для разных процессов и уровней управления систем жизненный цикл будет немного отличаться по длительности и содержанию работ каждого этапа, однако значительных, требующих отдельного детального рассмотрения в данной теме, закономерностей не наблюдается.

#### 1.4.6. Сфера применения

В зависимости от специфики процессов и задач организация может испытывать потребность в совершенно разных данных и механизмах их обработки. В то время как логистическим или автотранспортным предприятиям требуется отслеживать перемещение транспортных средств, интернет-магазину жизненно важна система приема и обработки платежей, а поликлинике — решение для автоматизации записи на прием и ведению историй болезни. Сфер применения информационных систем — бесчисленное множество, однако в качестве примера для иллюстрации их разнообразия приведем несколько категорий (табл. 1.9).

Таблица 1.9

Классификация ИС по сфере применения

Категория	Описание	Пример
Финансовая/учетная	Система, поддерживающая получение данных для финансового планирования, контроля бюджета, управления портфелем проектов на уровне финансового анализа эффективности	Стандартные функции финансовых и контроллинговых модулей основных ERP-систем

Категория	Описание	Пример
Медицинская	Существуют разные масштабы подобных ИС — от ведения учета в одном лечебном учреждении до консолидации данных больниц/государственных учреждений/пациентов	ЕМИАС — система, внедряемая в лечебных учреждениях Москвы в 2012—2013 гг., поддерживающая функции автоматического поиска врача и записи на прием как через онлайн-кабинет, так и через специально установленные в больницах терминалы
Географическая	Данный вид систем имеет название геоинформационных, работая в основном с пространственными данными, получая сведения о местоположении объекта и передавая динамику передвижений в реальном времени. Как правило, принцип действия ГИС заключается в наложении нескольких слоев карт (здания, ландшафт, трафик) друг на друга с последующим комбинированием необходимой информации	Может использоваться компанией для отслеживания грузового транспорта с товарами

Однако уже сейчас существуют социальные сети, где пользователям предлагается выкладывать свои фотографии или видеоклипы. Эти ИС не попадают ни под один пункт рассматриваемой классификации. Такой способ группировки информационных систем не рассчитан на постоянно растущий объем разнородной информации, а также на высокую популярность информационных систем личного пользования и распространение ИС в разных цифровых форматах. К сожалению, аналитики не смогли предугадать эту тенденцию, что сказывается на проработанности этой классификации ИС.

**Особенности ЖЦ.** Как и в случае с классификацией по процессам, для систем различных сфер применения и функциональности (см. ниже) жизненный цикл будет немного отличаться по длительности и содержанию работ каждого этапа, однако единых закономерностей выделять в данном случае не будем.

#### 1.4.7. Функциональность

Информационные системы могут работать в одной (!) сфере применения (например, в логистической компании), однако иметь абсолютно разные функции и технологии. Ниже приведем основные

типы внедряемых на сегодняшний день организациями прикладных программ.

**ERP, управление предприятием.** В первую очередь приведем обзор модулей ERP-системы управления ресурсами компании как одной из основных систем в любой компании независимо от масштабов, отрасли, сферы деятельности и других факторов. Подобные системы используют единую транзакционную систему обработки операций для всех модулей, поддерживают различные локализации, системы расчетов, учетные политики, различные схемы налогообложения и другие параметры.

Функциональность, предлагаемую ERP, рассмотрим на примере системы ведущего мирового поставщика — немецкой компании SAP, основные функции которой перечислены ниже:

- управление производством — планирование;
- управление основными средствами:

  - управление техническим обслуживанием и ремонт оборудования,
  - управление основными средствами,
  - управление инвестициями и капитальным строительством;

- управление финансами:

  - бухгалтерский учет,
  - учет основных средств,
  - финансовый менеджмент,
  - управление цепочкой создания стоимости,
  - управление инвестициями,
  - управление недвижимым имуществом;

- управление контроллингом/отчетностью:

  - планирование/управление экономикой предприятия,
  - учет затрат,
  - контроль прибыльности и деятельности предприятия;

- управление логистикой:

  - управление материальными потоками: снабжение и управление запасами,
  - управление сбытом: распределение, поставки и выставление счетов;

- управление персоналом:

  - кадровый учет,
  - организационный менеджмент,

- управление временными данными и расчетом заработной платы;
- управление проектами;
- управление информационными потоками/потоками операций;
- управление качеством — планирование, проверка и контроль качества производства и закупок.

Отдельного внимания заслуживает класс систем Enterprise Resource and Relationship Processing, замыкающий предшествующую историю развития систем управления процессами организаций. Предложенная для их описания концепция Gartner получила имя ERP II. Принято считать, что второе поколение ERP включает в себя такие компоненты, как управление ресурсами предприятия, а также управление взаимоотношениями с клиентами и поставщиками, т. е.  $ERP\text{ II} = ERP + CRM + SCM$ . Таким образом, ERP выходит за рамки задач оптимизации исключительно внутренних ресурсов предприятия, фокусируясь также на внешних связях и активностях. Среди ее основных особенностей — использование понятий *Value chain* и *E-commerce*, в отличие от простой оптимизации предприятия; покрытие не только производства, финансов, продаж и дистрибуции, но и остальных межпроизводственных и индустриальных функций, как внешних, так и внутренних. Архитектура подобных систем гораздо более интернет-ориентирована и открывает возможности сотрудничества с другими организациями, в том числе при использовании одной ИС несколькими предприятиями. Совместное предпринимательство, в свою очередь, способствует повышению скорости обработки заказов и доставки продукции потребителю.

Далее рассмотрим другие классы систем, которые частично могут пересекаться с функциональностью ERP-систем, однако чаще используются в качестве независимых решений поставщиков, отличных от поставщиков ERP. Описания и примеры ПО этих классов приведены в табл. 1.10.

Таблица 1.10

**Классификация ИС по функциональности**

Описание	Примеры программных продуктов
<i>BI (аналитика и отчетность)</i>	
Планирование, управление эффективностью, финансовая и операционная аналитика, а также анализ продаж, подача отчетности в регулирующие органы и формирование квартальных и годовых фактических показателей деятельности.	Oracle Business Intelligence IBM Cognos BI SAP BW, BO SAS Performance Management
<i>MDM / управление нормативно-справочной информацией</i>	
Получение, обработка, хранение и предоставление данных для использования различными корпоративными системами, поддержание информации в корректном состоянии (устранение дублирования, контроль целостности и непротиворечивости, поддержка распределенного ввода), упрощение миграции данных	SAP MDM IBM Information Server Microsoft MDM SAS Dataflux MDM Siemens PSS/ODMS SISCO UIB Store Oracle Master Data Management Suite

Продолжение табл. 1.10

Описание	Примеры программных продуктов
<i>SCM / управление цепочками поставок</i>	
Возможности автоматизации всех этапов поставки товара — закупки сырья, их поставки, производства и транспортировки товаров клиентам, а также контроля местонахождения каждого объекта, а также поддержка моделирования различных ситуаций в рамках сети поставок, расчет оптимальных запасов, прогнозы и учет по контрольным показателям	SAP SCM Oracle SCM JDA Software Manhattan Associates
<i>CRM / управление взаимоотношениями с потребителями</i>	
Автоматизация продаж (включает счета, контакты, управление привлечением и работой с клиентом). Управление анализом эффективности продаж и расчетом скидок/премий, анализ рынка, прогнозирование объемов операций. Управление прямыми маркетинговыми организациями и маркетинговыми ресурсами	Salesforce.com Terrasoft Microsoft Axapta/Navision/ CRM Sales Logix Oracle e-Business Suite SAP CRM
<i>BPM / управление бизнес-процессами</i>	
Системы управления бизнес-процессами предоставляют возможность повысить степень контроля над исполнением процессов, назначать показатели эффективности, владельцев процессов, контролировать результаты деятельности. Системы BPM предоставляют инструменты редактирования графического представления процессов, а также инструменты анализа для определения потенциальных способов их оптимизации	Oracle BPEL Manager Tibco BPM IBM Lombradi
<i>ECM / управление корпоративной информацией и документооборотом</i>	
Управление информационными ресурсами предприятия, в том числе электронными документами, образами бумажных документов, медиаконтентом и файлами различных форматов. Регистрация файлов в системе, тегирование, обработка и передача, механизмы согласования и утверждения, назначения ответственных	EMC Documentum Microsoft SharePoint SAP Open Text Oracle Content Manager IBM FileNet Hyland Software ECM
<i>KM / управление знаниями</i>	
Построение классификатора знаний, организация их первичного извлечения, а также доступа и актуализации информации	WebSphere Portal Microsoft SharePoint IBM OmniFind IBM Lotus (Notes, Domino,

Продолжение табл. 1.10

Описание	Примеры программных продуктов
	Sametime, Quickr, Connections) IBM Content Manager IQMen DocsVision
<i>PLM / управление жизненным циклом изделий</i>	
Управление информацией об изделии (цифровым макетом) от этапа проектирования до этапа снятия с эксплуатации, с проведением цифровой сборки, интеграции материалов и оборудования в данный процесс, контроль качества и, разумеется, визуализации продукта на всех этапах процесса	SAP PLM Oracle PLM Siemens A&D UGS Software Teamcenter 1C: Предприятие 8. PDM Управление инженерными данными ЛОЦМАН: PLM
<i>HRM / управление персоналом</i>	
Комплексные системы HRM позволяют автоматизировать активности по поиску/подбору персонала, его адаптации, оценке, обучению и развитию, формированию должностных инструкций	Oracle PeopleSoft HRMS mySAP HRM QUINYX WorkForce Infor HRM Workday HRM Ceridian Dayforce HRM Компас: Управление персоналом БОСС-Кадровик
<i>PPM / управление портфелем проектов</i>	
Интегрированные системы управления портфелями проектов предприятия позволяют автоматизировать весь процесс анализа, планирования, расстановки приоритетов, выбора проектов, и разумеется, дальнейшего мониторинга хода проектов и балансировки загрузки ресурсов и выполнения объема проекта. Неотъемлемыми функциями являются также распределение задач и проектов, включая детальный сбор статистики хода проекта для дальнейшего анализа и совершенствования процесса	Oracle Primavera PPM Oracle Fusion PPM Microsoft Project CA Clarity PPM Planview PPM HP PPM
<i>EP / корпоративный портал</i>	
Обеспечение доступа сотрудников к корпоративным приложениям и информации. Заходя на корпоративный портал, пользователь может в одном интерфейсе видеть свои проекты/назначения, сообщения почты, корпоративные новости, каталог контактов сотрудников,	IBM Websphere SAP Enterprise Portal Microsoft SharePoint BEA WebLogic Oracle Portal

Продолжение табл. 1.10

Описание	Примеры программных продуктов
а также скачать необходимые шаблоны документов (например, заявление на отпуск) либо получить доступ к файлам из базы знаний.	
<i>ESB / сервисная шина</i>	
Для обеспечения взаимодействия сервисов предоставляется единая точка, обеспечивающая не только передачу информации и доступ многих приложений к ней, но и преобразование данных, и транзакционность. Благодаря единойшине легко подключаются новые системы и проводится оперативный перенос данных при сохранении их целостности	BEA WebLogic IBM WebSphere Informatica PowerExchange Microsoft BizTalk SAP PI Oracle Fusion Middleware SISCO Utility Bus Sonic MQ, ESB Sybase Real-Time Events
<i>SEARCH / корпоративный поиск</i>	
Сервис поиска как по внешним, так и по внутренним источникам данных. Индексация и обработка текстовых и медиафайлов из множества структурированных и неструктурированных баз с предоставлением информации при обязательном соблюдении корпоративных политик безопасности (выдача пользователям только доступного им содержимого в зависимости от определенных прав доступа и ролей)	Autonomy IDOL Endeca Google Search Appliance IBM OmniFind Microsoft Enterprise Search Oracle Secure Enterprise Search FAST ESP
<i>GIS / геоинформационная система</i>	
Интегрированное решение по работе с пространственными данными. Так, ГИС предоставляет широкие возможности визуализации информации и создания нескольких слоев объектов на картах, выводя только необходимые категории объектов в любой момент времени. Мониторинг положения объектов по GPS/ГЛОНАСС делает ГИС незаменимой системой для контроля автотранспорта/перемещения грузов/персонала, расчета оптимальных маршрутов, визуализации размещения магазинов/рекламных щитов и для многочисленных прочих целей бизнеса	Autodesk AutoCAD MAP, World, MAPGuide Bentley MicroStation ERDAS IMAGINE Esri ArcGIS, ArcFM Intergraph MapInfo GeoGraph SICAD Overwatch RemoteView GE Smallworld
<i>MES / управление производством</i>	
Программные решения данного класса поддерживают координацию, анализ и оптимизацию выпуска продукции на уровне отдельной организационной единицы предприятия	SAP MES Oracle MES Honeywell OptiVision Invensys Wonderware MES

Окончание табл. 1.10

Описание	Примеры программных продуктов
(например, цеха), в частности в области оперативно-календарного детального планирования и диспетчеризации производственных процессов (также на основе анализа производительности)	iBASEt Solumina Paperless Manufacturing Execution System 1C: MES Оперативное управление производством
<i>CAE / управление функциональным проектированием</i>	
Анализ, моделирование, оптимизация проектных решений, включая анализ прочности и расчет процессов на макроуровне (при определении их взаимовлияния, обусловленного различной природой). Для систем данного класса также важна возможность применения имитационного моделирования	Dassault Systems CATIA UGS NX PTC CAE
<i>CAD / управление проектированием изделий</i>	
Системы автоматизированного проектирования позволяют определять геометрию конструкций/изделий, получая двух- и трехмерные модели, метрические расчеты, необходимую визуализацию и конструкторскую документацию	Dassault Systems CATIA Siemens PLM Software Autodesk AutoCAD, Mechanical Desktop, Inventor, Streamline PTC CAD UGS NX
<i>CAM / управление технологическим проектированием</i>	
Разработка технологических процессов, их моделирование (с построением траекторий движения заготовок/материалов/инструмента в процессе производства, расчет необходимого для операции времени и условий)	Dassault Systems CATIA Siemens PLM Software Autodesk CAM PTC CAM UGS NX Siemens PLM Software

#### 1.4.8. Прошлое и будущее в классификациях ИС

Стремительное развитие технологий приводит к постоянному размыванию границ между существующими категориями и созданию абсолютно новых. Рассмотрим классификации, уходящие в прошлое, и способствующие этому тенденции.

Классификация по характеру использования вычислительных ресурсов — распространение Интернета и совершенствование систем информационной безопасности. Существует деление ИС на локальные и распределенные<sup>1</sup>. Основная идея локальных ИС

<sup>1</sup> Когаловский М. Р. Энциклопедия технологий баз данных. М. : Финансы и статистика, 2002.

заключается в том, что они используют единственную ЭВМ и применяются для автоматизации отдельных функций управления. В свою очередь, распределенные ИС используют несколько ЭВМ, связанных сетью. На данный момент опыт «дистанционной» работы еще не слишком значителен, и остаются локальные ИС, но с дальнейшим развитием Интернета подобные ИС исчезнут. Таким образом, эта классификация выйдет из использования.

**Классификация по степени автоматизации — повсеместная автоматизация и исчезновение «ручных» ИС.** По степени автоматизации ИС делятся на ручные, автоматические и автоматизированные. Но уже сейчас можно поставить под сомнение важность выделения ручных информационных систем. Несмотря на то что они все еще существуют, их роль в современном обществе невелика. Объем информации в неэлектронном виде незначителен, ведь, не используя технические средства, невозможно обрабатывать большие объемы информации. На данный момент ручные информационные системы существуют в виде городских библиотек, которые представляют в большей степени историческую ценность, нежели информационную. Их также можно встретить в небольших провинциальных магазинах, где учет товаров до сих пор не автоматизирован. Внедрение таких систем является неоптимальным, тем более сейчас, когда идет активное развитие ИС, которые работают без вмешательства человека, и автоматизируются немногие оставшиеся ручные ИС. Вследствие этого можно предположить, что в ближайшем времени все системы станут смешанного типа, а ручная работа будет рассматриваться в виде части большой автоматизированной системы.

**Классификация по классу реализуемых технологических операций — появление новых форматов данных, в том числе от органов чувств.** В данной категории<sup>1</sup> выделяют системы с текстовыми редакторами, системы с табличными редакторами, СУБД, СУБЗ, системы с графикой, мультимедиа и гипертекстом. В будущем эти параметры перестанут характеризовать информационную систему. Если сейчас разбиение ИС на данные категории является достаточно четким, то в скором времени границы между пунктами этой классификации станут размытыми из-за использования семантической паутины. Человек хочет работать с привычной для него информацией, т. е. с такой, которая воспринимается органами чувств. К ним относятся изображение, звук, запах. Уже сейчас начинают появляться ИС смешанного типа. К примеру, в бюро патентования США при работе с базой данных патентов используется технологии, работающие с разными типами операций, будь то графические ре-

---

<sup>1</sup> Salmeron J. L. EIS Success: Keys and difficulties in major companies // Technovation. Vol. 23, Iss. 1. 2003. P. 35–38.

дакторы и редакторы гипертекста<sup>1</sup>. Также при обработке запросов поисковых машин проводится извлечение информации из БД запрашиваемого пользователем контента в разных форматах данных (например, видео с YouTube и статей из Wikipedia).

**Классификация по оперативности обработки данных — распространяющая потребность в моментальном получении информации.** В данной категории выделяют оперативные ИС и ИС пакетной обработки. Сейчас обе эти разновидности информационных систем широко используются. Оперативные ИС необходимы в банковских, справочных системах, в бронировании билетов. Пакетная обработка, в свою очередь, применяется научно-техническими центрами при проведении статистических и аналитических исследований. При этом для первой группы характерны высокая скорость взаимодействия между системой и пользователем и малые объемы информации. Для второй — наоборот. По причине того что люди обладают доступом к практически неограниченной коллекции данных, а скорость жизни постоянно растет, необходимой становится возможность получения большого объема информации, качественно отсортированной, подобранный и моментально полученной из Интернета или из другой информационной сети. Исходя из этого, имеет место предположение о том, что в скором времени ИС будут обладать всеми этими качествами одновременно, так что данная классификация потеряет актуальность.

**По сферам деятельности — развитие «облачных» и collaborative technologies.** Несмотря на то что появление ИС относят к далеким 1950-м гг., их развитие все еще не набрало полную силу, люди только начинают привыкать к новым возможностям. В обозримом будущем существенная часть личных данных пользователей может переместиться из компьютеров в «облака», из чего следует появление еще одной ветви классификации по сфере деятельности — персональные ИС. Все эти факты говорят о скорой смене приоритетов и взглядов на настоящее, выраженных в виде классификаций.

## Контрольные вопросы и задания

1. Дайте определение понятия «информационная система».
2. Каковы основные части информационной системы?
3. Каковы ключевые задачи и свойства информационной системы?
4. Каковы результаты использования ИС на предприятии?
5. В чем заключаются различия между автоматизированными системами управления и корпоративными информационными системами?
6. Что такое цифровая платформа?

---

<sup>1</sup> Из личного интервью главного экзаменатора патентного бюро США (Alexander Markoff).

7. Что такое корпоративная информационная система?
8. Каковы основные этапы создания корпоративных информационных систем?
9. Как изменялась функциональность ИС по мере развития технологий?
10. Какие системы и модули ИС возникали в различные исторические периоды?
11. Каковы основные классификации информационных систем?
12. Какие классы ИС выделяются при проведении классификации по архитектуре?
13. Какие типы ИС выделяются при классификации по степени структурированности задач и по характеру обработки данных?
14. Как классифицируют ИС по процессам и уровням управления?
15. Какие классы информационных систем выделяют по функциональности?
16. Какие функции обычно включает в себя ERP-система (на примере SAP)?
17. Как может измениться классификация ИС в обозримом будущем?

### **Рекомендуемая литература**

1. Боронов, В. В. Информационные технологии и управление предприятием / В. В. Боронов [и др.]. — Москва : ДМК Пресс, 2004.
2. Григорьев, Л. Менеджмент по нотам. Технология построения эффективных компаний / Л. Григорьев [и др.] ; под общей редакцией Л. Григорьева. — Москва : Альпина Паблишерз, 2010.
3. Ермошкин, Н. Стратегия информационных технологий предприятия: Как Cisco Systems и ведущие компании мира используют интернет-решения для бизнеса / Н. Ермошкин, А. Тарасов. — Москва : Издательство Московского гуманитарного университета, 2003.
4. Зараменских, Е. П. Введение в бизнес-информатику / Е. П. Зараменских. — Москва : Издательство Юрайт, 2016.
5. Минцберг, Г. Структура в кулаке. Создание эффективной организации / Г. Минцберг. — Санкт-Петербург : Питер, 2004.

## **Тема 2**

# **ЖИЗНЕННЫЙ ЦИКЛ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

---

В этой теме подробно рассматривается понятие жизненного цикла информационной системы. Рассмотрены основные модели, стандарты и методологии, которые используются для описания жизненного цикла информационной системы. Современное развитие данной области знаний иллюстрируется международными и отечественными стандартами, которые сегодня широко применяются на практике. Отдельно рассмотрены модели жизненного цикла программного обеспечения.

После изучения данной темы студент будет:

**знатъ**

- понятие жизненного цикла информационной системы;
- историю развития концепции жизненного цикла ИС и соответствующие модели ЖЦИС;

• модели жизненного цикла программного обеспечения;

**уметь**

- сравнивать модели жизненного цикла ИС, оценивая их содержание;
- применять различные модели, стандарты и методологии жизненного цикла ИС;
- применять различные модели, стандарты и методологии жизненного цикла ПО;

**владеть навыками**

- эффективного поиска необходимых референтных процессов в рассмотренных стандартах и методологиях;
  - практического применения моделей, стандартов и методологий жизненного цикла ИС и ПО.
- 

Жизненный цикл информационных систем, как уже было сказано, представляет собой непрерывный процесс, включающий ряд определенных этапов.

---

**Жизненный цикл информационной системы** — непрерывный процесс, началом которого становится момент принятия решения о необходимости системы, а завершением — ее изъятие из эксплуатации. Этапы создания системы до момента ввода в эксплуатацию могут рассматриваться как самостоятельные проекты, каждый из которых имеет конкретный результат и ограничения.

---

А так как процессы создания и конфигурирования различных информационных систем включают в себя один и тот же набор этапов, то можно говорить о **моделях жизненного цикла**.

В общем виде жизненный цикл ИС включает в себя следующие этапы:

- планирование проекта;
- анализ и постановка задачи;
- проектирование;
- разработка;
- развертывание и внедрение;
- эксплуатация;
- поддержка;
- модернизация;
- утилизация.

Средняя продолжительность ЖЦИС сегодня составляет порядка 10—15 лет, но срок эксплуатации создаваемых ИС постепенно снижается. В наше время ИС нередко перестает соответствовать требованиям бизнеса еще в процессе внедрения. Компании стремятся отказаться от использования устаревающих ИС из-за высоких расходов, связанных с их поддержкой. Кроме того, ИС может устареть преждевременно из-за реорганизации бизнес-процессов компаний, перехода на другой рынок, появления новых технологий и пр.

## **2.1. История развития концепции ЖЦ информационных систем**

Концепция жизненных циклов в сфере работы с информационными системами применяется достаточно давно. Одна из первых моделей ЖЦИС была опубликована У. У. Ройсом в 1970 г. В данной модели процесс создания ИС и дальнейшей работы с ней представлялся в качестве различных фаз, которые проходит ИС в процессе своего существования. В дальнейшем, по мере возрастания популярности, модель получила название *водопадной* (*waterfall model*). Также различные вариации модели У. У. Ройса зачастую называют *каскадными*.

В первоначальной версии модели выделялись семь этапов ЖЦИС:

- планирование проекта;
- определение требований;
- проектирование;
- конструирование;
- воплощение;
- тестирование и отладка;
- инсталляция;
- поддержка.

Несмотря на кажущуюся простоту, модель У. У. Ройса оказала серьезное влияние как на развитие подходов к разработке ИС, так и на развитие концепции жизненных циклов ИС. На протяжении следующих десятилетий регулярно появлялись различные вариации каскадной модели ЖЦИС, каждая из которых содержала различные стадии и способы перехода между ними. Однако базовые принципы, предложенные У. У. Ройсом, не изменились: движение между стадиями было односторонним, а их выполнение оставалось исчерпывающим.

В качестве примера адаптированной каскадной модели можно привести каскадную модель ЖЦИС Марри Кантора, предложенную в 2002 г. Отдельного внимания заслуживает факт детализации каждой из фаз ЖЦ (рис. 2.1). Так, по мнению Кантора, на каждом этапе ЖЦИС происходят следующие операции:

- составление плана действий;
- планирование работ для каждого действия;
- применение операции отслеживания хода выполнения действия (включая контрольные этапы).

Помимо детализации этапов (по-прежнему достаточно укрупненной), Кантор перечислял и промежуточные результаты для каждого из этапов ЖЦИС в рамках каскадной модели.

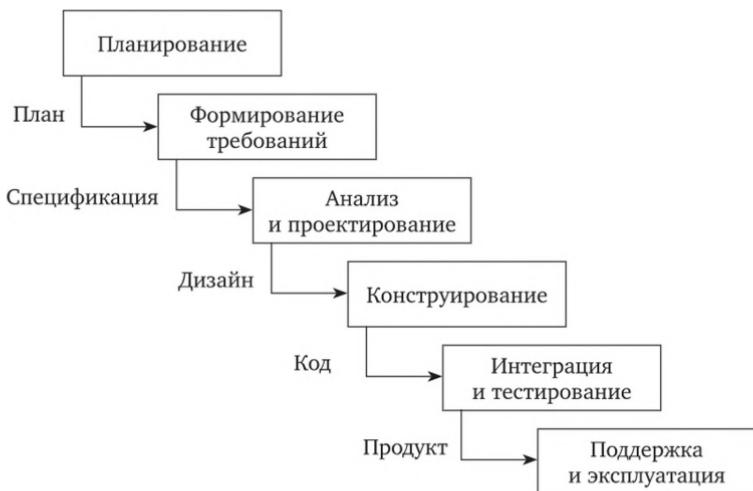


Рис. 2.1. Каскадная модель жизненного цикла ИС  
(по М. Кантору)

Тем не менее Кантор поддержал выводы другого исследователя — Ф. Брукса. Был сделан вывод о том, что применение каскадной модели ЖЦИС в реальном мире практически невозможно, поскольку

современная экономика — слишком динамичная среда, в рамках которой не представляется возможным осуществлять столь жесткие по рамкам проекты. Требования к ИС регулярно корректируются и уточняются, они не могут быть четко и однозначно определены до начала реализации проекта по созданию информационной системы. Соответственно, прохождение таких четко ограниченных стадий ЖЦИС не представляется возможным.

В 1988 г. Барри Боэмом была предложена *спиральная модель ЖЦИС*, которая частично устранила недостатки каскадной модели ЖЦИС (рис. 2.2).

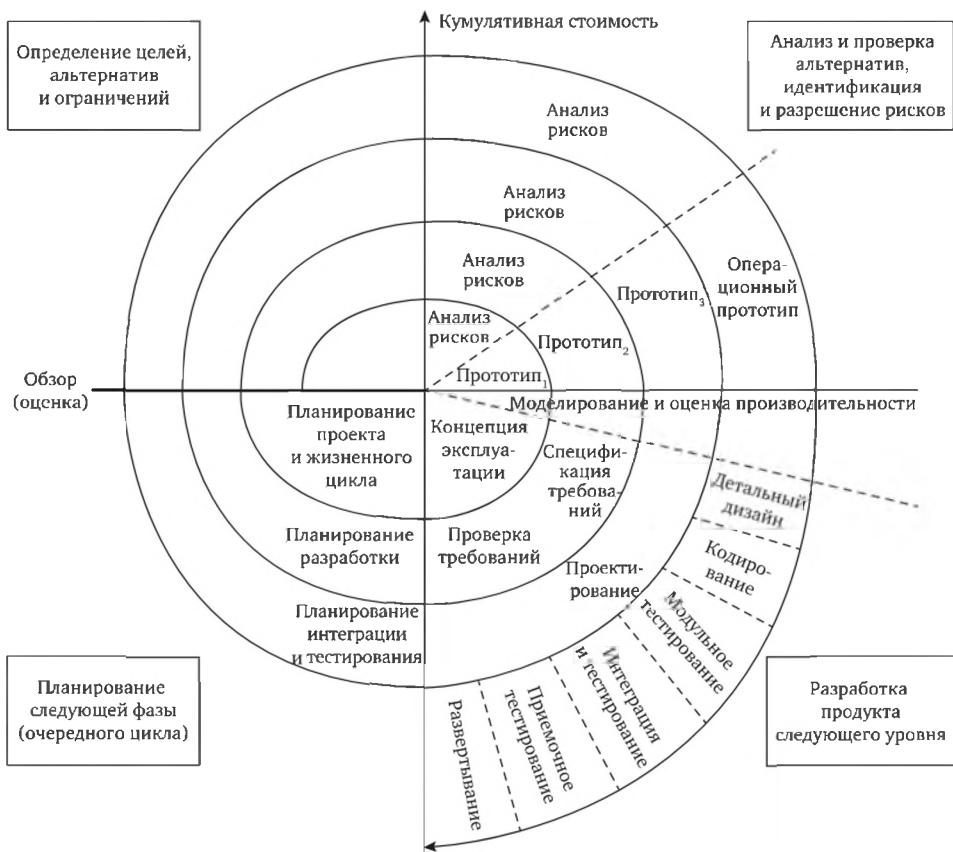


Рис. 2.2. Спиральная модель ЖЦ разработки информационной системы (по Б. Боэмму)<sup>1</sup>

Одним из основных качественных отличий модели является пристальное внимание к рискам, работа с которыми является составной частью ЖЦИС. Боэм выделял следующие типы рисков:

- нехватка специалистов;
- сжатые сроки и низкий бюджет;

<sup>1</sup> Модели жизненного цикла программного обеспечения // ХАБРАХАБР. 2011. 11 янв. URL: <http://habrahabr.ru/post/111674> (дата обращения: 15.06.2020).

- создание ненужной функциональности;
- создание неправильного интерфейса для пользователей;
- «золотая сервировка» (излишний перфекционизм, излишняя оптимизация и слишком пристальное внимание к второстепенным деталям);
  - слишком сильный поток изменений;
  - недостаток информации об окружении системы и субъектах, вовлеченных в итерации;
  - проблемы в работах, которые выполнялись при помощи внешних (относительно проекта) ресурсов;
  - низкая производительность итоговой информационной системы;
  - разрыв между уровнями знаний специалистов с разными квалификациями.

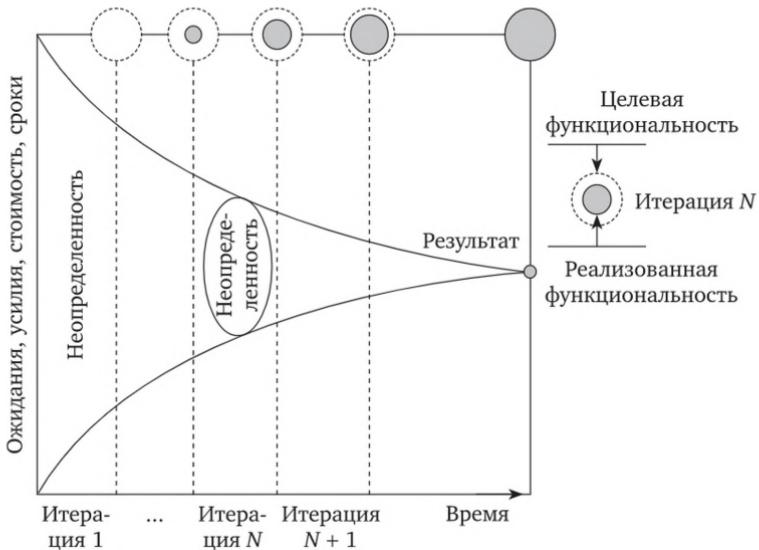
В дальнейшем Боэм исследовал практики применения предложенной модели ЖЦИС. В 2000 г. исследователь пришел к выводу, что успешное применение спиральной модели зависит от пяти ключевых условий:

- параллельное определение активностей проекта;
- уделение внимания на каждом этапе:
  - целям и ограничениям,
  - альтернативам организации процесса и технологическим решениям,
  - идентификации рисков и предотвращению их наступления,
  - оценке со стороны заказчика,
  - достижению согласия по поводу дальнейших действий;
- определение уровня усилий (на основе рисков) для каждого уровня ЖЦИС;
- управление жизненным циклом на основе контрольных точек;
- уделение особого внимания проектным работам как фазам ЖЦ.

В дальнейшем на основе спиральной модели ЖЦИС возникла и получила широкое распространение методология быстрой разработки RAD, которая рассматривается в одном из последующих параграфов. Также в рамках данной модели была создана методология SADD (Spiral Architecture Driven Development).

В дальнейшем оказалось, что спиральная модель Б. Боэма и другие версии спиральных моделей оптимально подходят для разработки крупных и дорогостоящих проектов. Спиральная модель позволяет вносить серьезные изменения в ИС на всех этапах ее ЖЦ.

И, наконец, отдельного внимания заслуживает эволюционная модель ЖЦИС М. Фаулера и С. Амблера. Эволюционная модель основана на множестве последовательных операций, по мере выполнения которых увеличиваются функциональные возможности информационной системы (рис. 2.3).



*Рис. 2.3. Уменьшение неопределенности и расширение функциональности при итеративной организации ЖЦ информационной системы<sup>1</sup>*

На сегодняшний день понимание циклической природы информационных систем фиксируется практически во всех подходах к разработке, созданию, внедрению и сопровождению ИС. Концепцию ЖЦИС содержат наиболее авторитетные документы в области информационных систем, включая ISO 12207, 15288.

### 2.1.1. ГОСТ 34.601—90

**Национальный стандарт ГОСТ 34.601—90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.**

Отличается высокой степенью формализации и по умолчанию предполагает каскадный подход. На сегодняшний день ГОСТ много-кратно становился основой для доработок и частичного использования в других стандартах и методологиях, и в целом в исходном виде не является исчерпывающим источником информации для выполнения проекта разработки и внедрения.

Благодаря своей структурированности ГОСТ 34.601—90 до сих пор служит самодостаточной основой, которую можно адаптировать к конкретным условиям деятельности предприятия. Приложе-

<sup>1</sup> Орлик С. Модели жизненного цикла программного обеспечения. URL: [ftp://ftp.asu.ru/incoming/alex/SWE/software\\_lifecycle\\_models.pdf](ftp://ftp.asu.ru/incoming/alex/SWE/software_lifecycle_models.pdf) (дата обращения: 15.06.2020).

ние к стандарту содержит детальное описание работ, включая списки формируемых по завершении этапа документов.

Таблица 2.1

**Стадии создания ИС по ГОСТ 34.601—90**

Стадия	Этапы
Формирование требований к АС	Обследование объекта и обоснование необходимости создания АС. Формирование требований пользователя к АС. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания)
Разработка концепции АС	Изучение объекта. Проведение необходимых научно-исследовательских работ. Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя. Оформление отчета о выполненной работе
Техническое задание	Разработка и утверждение технического задания на создание АС
Эскизный проект	Разработка предварительных проектных решений по системе и ее частям. Разработка документации на АС и ее части
Технический проект	Разработка проектных решений по системе и ее частям. Разработка документации на АС и ее части. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации
Рабочая документация	Разработка рабочей документации на систему и ее части. Разработка или адаптация программ
Ввод в действие	Подготовка объекта автоматизации к вводу АС в действие. Подготовка персонала. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями). Строительно-монтажные работы. Пусконаладочные работы. Проведение предварительных испытаний. Проведение опытной эксплуатации. Проведение приемочных испытаний
Сопровождение АС	Выполнение работ в соответствии с гарантийными обязательствами. Послегарантийное обслуживание

## Пример

На этапе «Проведение предварительных испытаний» (см. табл. 2.1) осуществляют:

- испытания АС на работоспособность и соответствие техническому заданию в соответствии с программой и методикой предварительных испытаний;
- устранение неисправностей и внесение изменений в документацию на АС, в том числе эксплуатационную в соответствии с протоколом испытаний;
- оформление акта о приемке АС в опытную эксплуатацию.

Также в стандарте приведен список основных типов организаций, участвующих в работах по созданию АС, что позволяет сформировать понимание сути процесса.

## Пример

Организация-заказчик (пользователь), разработчик, поставщик, генпроектировщик, ...

### 2.1.2. ISO/IEC 15288 (ГОСТ Р ИСО/МЭК 15288—2005)

**Международный стандарт:** ISO/IEC 15288:2005 Systems engineering. System life cycle processes (Системотехника. Процессы жизненного цикла системы).

**Российский аналог:** ГОСТ Р ИСО/МЭК 15288—2005 Информационная технология. Системная инженерия. Процессы жизненного цикла систем.

Достаточно «молодой» стандарт системной инженерии (впервые представленный в 2002 г.), ISO/IEC 15288 фокусируется на вопросах жизненного цикла системного уровня, в особенности тейлоринге (*tailoring*) — по сути, настройке и адаптации процессов ЖЦ к конкретным требованиям и ограничениям.

В отличие от рассмотренного ранее стандарта, ISO 15288 распространяется на системы в целом, охватывая такие их элементы, как: «технические средства, программные средства, люди, процессы (например, процесс оценки), процедуры (например, инструкции оператора), основные средства и природные ресурсы (например, вода, объекты живой природы, минералы)»<sup>1</sup>. Согласно данному стандарту, любой процесс ЖЦ может быть начат в любой момент, без ограничения порядка использования и последовательности (в том числе при параллельном выполнении нескольких процессов).

Важно также отметить высокий уровень абстракции ISO 15288 в сравнении с ISO 12207, рассматриваемом ниже, так как данный стандарт не приводит ролей, состава работ и конечных результатов

<sup>1</sup> ISO/IEC 15288:2005 Systems engineering. System life cycle processes.

в виде списка выходных документов, оставаясь лишь на уровне концепции.

### Пример

В процессе управления ресурсами (п. 5.3.5 Стандарта) в качестве основных пунктов деятельности приводятся следующие:

- определять и обеспечивать поддержку инфраструктуры ресурсов, необходимой для выполнения организацией требований настоящего стандарта и осуществления поддержки проекта;
- получать ресурсы, за исключением персонала, необходимые для внедрения и осуществления проектов;
- проявлять заботу о персонале, занятом в осуществлении текущих проектов;
- стимулировать персонал, например, посредством предоставления возможности карьерного роста или при помощи системы поощрений;
- контролировать области взаимодействия нескольких проектов для разрешения связанных с графиками их реализации конфликтов.

Очевидно, что формулировки «...обеспечивать поддержку инфраструктуры ресурсов...» или «проявлять заботу о персонале...» не являются конкретными и могут толковаться по-разному. В связи с этим для использования данного стандарта в проекте создания и эксплуатации системы важно иметь поддержку высшего руководства. Особенность ISO 15288 состоит в том, что он может использоваться как со стороны заказчика, так и со стороны исполнителя.

Стандарт содержит четыре основные группы процессов (предприятия, соглашения, проекта и технические, рис. 2.4), описывающие соответственно вспомогательные корпоративные процессы, взаимодействие с контрагентами, управление проектом и саму реализацию системы.

Важной положительной чертой стандарта является его связь с бизнес-стороной проекта создания системы за счет групп процессов предприятия и соглашения. Благодаря наличию подобных разделов стандарта появляется связь с соответствующими корпоративными функциями, и для бизнеса становится более понятным место процессов ЖЦ в процессах организации в целом.

## 2.2. Жизненный цикл программного обеспечения

Жизненный цикл программного обеспечения — это период времени, который начинается в момент принятия решения о создании программного продукта и заканчивается прекращением его эксплуатации и утилизацией. В ряде случаев ЖЦПО рассматривается отдельно, хотя ввиду высокой значимости ПО как основного компонента ИС жизненный цикл ПО может совпадать с ЖЦИС.



Рис. 2.4. Группы процессов по стандарту ISO 15288

Тем не менее в ряде случаев требуется отдельно рассматривать ЖЦПО, поскольку только так удастся добиться эффективного создания, внедрения и использования нового программного продукта или модернизации имеющегося.

### **2.2.1. SWEBOK (ISO /IEC TR 19759:2015)**

**Документ** SWEBOK, полностью называющийся IEEE Guide to the Software Engineering Body of Knowledge, — свод знаний по программной инженерии. Документ имеет достаточно близкое отношение к концепции ЖЦИС. Области знаний, выделяемые в SWEBOK, используются на различных этапах ЖЦИС. Кроме того, области знаний в данном документе тесно пересекаются с основными фазами, которые выделяются в различных моделях ЖЦИС.

**История** SWEBOK. Термин «программная инженерия» впервые был использован в научном обиходе в 1968 г. на конференции НАТО в области компьютерных наук. Уже спустя четыре года организация IEEE Computer Society опубликована сборник Transaction on Software Engineering.

В 1976 г. та же организация сформировала комитет по разработке стандартов в данной области. Первым стандартом программной инженерии стал стандарт IEEE Std 730, в котором прописывались критерии качества ПО. В дальнейшем IEEE стала инициатором выпуска различных стандартов, которые охватывали многие области программной инженерии, включая управление конфигурациями, тестирование, управление требованиями и т. п. Первой попыткой обобщить предмет программной инженерии стало в 1986 г. создание стандарта IEEE Std 1002 «Taxonomy of Software Engineering Standards».

В 1993 г. был создан комитет SWECC, ставший результатом совместной работы IEEE Computer Society и Association for Computing Machinery. Рабочие группы, сформированные внутри SWECC, работали по трем направлениям. Первая и наиболее крупная группа, занимающаяся созданием свода необходимых знаний и рекомендуемых практических навыков, создала основную базу для создания SWEBOK.

SWEBOK как таковой был инициирован только в 1998 г. На первом этапе был создан прототип организации SWEBOK. На втором этапе в 2001 г. была опубликована пробная версия SWEBOK, для создания которой пришлось учесть порядка 9 тыс. замечаний, предложенных 500 специалистами из 42 стран. В 2004 г. завершилась очередная часть создания SWEBOK.

SWEBOK V3 вышел в 2014 г. Обновленная версия включает гораздо больше областей знаний.

По мнению авторов документа, при создании SWEBOK им удалось добиться консенсуса за счет того, что в разработке, анализе

и доработке документа приняли участие сотни специалистов из разных стран и организаций — как коммерческих, так и некоммерческих. Сейчас этот стандарт известен также как международный стандарт ISO/IEC TR 19759:2015 и был обновлен в 2015 г.

**Содержание SWEBOK.** В рамках SWEBOK дается определение программной инженерии, которая трактуется как применение систематизированного, дисциплинированного и оцениваемого по количественным параметрам подхода к разработке, функционированию и сопровождению ПО. Иными словами, о программной инженерии можно говорить, если для создания ПО применяются инженерные методы.

Выделяются следующие цели создания SWEBOK:

1) продвижение в мире единого представления о программной инженерии за счет достижения консенсуса во время создания документа;

2) определение границ программной инженерии и ее места относительно других областей знаний;

3) характеристика содержания программной инженерии как научной дисциплины;

4) предоставление доступа к знаниям по программной инженерии;

5) создание базы для разработки учебных планов и программ сертификации.

Для определения границ программной инженерии в SWEBOK подробно рассматриваются 10 областей знаний. В качестве смежных дисциплин SWEBOK выделяет математику, компьютерные науки, менеджмент, управление проектами, системную инженерию и др.

SWEBOK содержит в себе только общепринятые знания по программной инженерии. К таковым знаниям относят все знания и навыки, владение которыми является обязательным условием профессиональной деятельности квалифицированного работника. В противовес общепринятым знаниям в SWEBOK используется категория исследовательских знаний, которые используются учеными, исследовательскими центрами или отдельными организациями.

**Области знаний SWEBOK V3 (2014).** В документе выделяются 15 областей знаний, относящихся к программной инженерии:

1) требования к ПО (Software Requirements);

2) проектирование ПО (Software Design);

3) конструирование ПО (Software Construction);

4) тестирование ПО (Software Testing);

5) сопровождение ПО (Software Maintenance);

6) управление конфигурацией ПО (Software Configuration Management);

7) управление в программной инженерии (Software Engineering Management);

- 8) процесс программной инженерии (Software Engineering Process);
- 9) инструменты и методы программной инженерии (Software Engineering Tools and Methods);
- 10) качество ПО (Software Quality);
- 11) профессиональные практики программного инжиниринга (Software Engineering Professional Practice);
- 12) экономика программного инжиниринга (Software Engineering Economics);
- 13) основы вычислений (Computing Foundation);
- 14) математические основы (Mathematical Foundation);
- 15) основы инжиниринга (Engineering Foundation).

Первые пять областей отражают традиционный жизненный цикл ИС, который отражается в водопадной модели разработки. Следующие пять областей перечислены в алфавитном порядке, причем каждая область описывается на основе подобластей, которые разбиваются на темы и подтемы.

Раздел «Требования к программному обеспечению» дает определение требованиям. Требование — это свойство, которое обязательно должно быть реализовано в создаваемом программном продукте для решения какой-либо реальной проблемы. Эта область знаний SWEBOK рассматривает все типы требований, а также проводит границу между системными требованиями и требованиями к ПО. Подобласть, описывающая процесс определения требований, также рассматривает связи между этапами процесса. Демонстрируется связь определения требований с другими областями программной инженерии.

Строго говоря, выявление требований — это первый этап, с которого начинается работа над любым программным продуктом. SWEBOK в данном разделе рассматривает, какие существуют источники требований и как разработчик должен собирать их.

После сбора требований начинается этап анализа требований, который должен обнаружить конфликты и противоречия между требованиями, а также выделить границы создаваемого ПО или ИС.

Вслед за этим наступает время спецификации требований. Для этого создается документ с требованиями, которые необходимо рассмотреть, пересмотреть, оценить и утвердить. Каждое требование из такого документа должно проверяться и утверждаться перед тем, как на его реализацию будут выделены какие-либо ресурсы. Требования приобретают силу закона только после утверждения их со стороны заказчика и исполнителя.

И, наконец, в завершение рассматриваются темы, которые приходится учитывать на практике во время работы с требованиями. К таким темам относятся: итеративность самого процесса, потребность в управлении вносимыми изменениями, отслеживание тре-

бований и способы, которые применяются для измерения требований.

Раздел «Проектирование программного обеспечения» дает основное определение. Проектирование — это «процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или ее компонента», завершающийся получением конкретных «результатов этого процесса». В процессе проектирования все требования превращаются в формализованные описания, на основе которых команда разработчиков выявляет путь решения проблемы. В процессе проектирования производится декомпозиция системы с целью обеспечения необходимой детализации для каждого из компонентов.

Подобласти проектирования ПО в SWEBOK рассматривают, какие существуют концепции в области проектирования программных проектов, а также иллюстрируют основные проблемы, возникающие в процессе проектирования. Для структуры и архитектуры ПО существует отдельная подобласть, которая также посвящена архитектурным стилям, шаблонам проектирования, программным семействам и т. п.

Кроме того, второй раздел SWEBOK фокусирует внимание на анализе качества создаваемого программного продукта. В рамках раздела обсуждаются существующие и наиболее популярные нотации, применяющиеся для документирования проектов на различных уровнях абстракции. Также в разделе описываются общие стратегии и особые методы проектирования.

Раздел «Конструирование программного обеспечения» рассматривает все этапы процесса создания функционирующего ПО. В данном разделе рассматриваются основные принципы, соответствие которым определяет полезность созданного ПО и эффективность его разработки.

Раздел «Тестирование программного обеспечения» рассматривает динамичную проверку работающей программы на соответствие ожидаемому поведению. Раздел описывает виды тестов и области тестирования. Вводятся различные подобласти тестирования и метрики, на основе которых выполняется тестирование.

Раздел «Сопровождение программного обеспечения» решает проблемы, связанные с возникновением незапланированных ситуаций, изменениями во внешней среде, появлением новых пользователей или ролей, появлением новых требований и т. п. Область знаний описывает концепцию сопровождения ПО на базовом уровне. Подобласти раздела рассматривают технические проблемы данной области и специфические вопросы управления.

Раздел «Управление конфигурацией программного обеспечения» идентифицирует конфигурации ПО в различные моменты времени для систематического контроля за изменениями конфигурации. Это

необходимо для обеспечения целостности конфигурации на протяжении ЖЦИС и ЖЦПО. Раздел рассматривает все действия, которые связаны с управлением конфигурациями ПО.

Раздел «Управление в программной инженерии» рассматривает планирование, координацию, измерение, мониторинг, контроль и отчетность программного обеспечения. Выделяются три уровня управления ПО: организационный, проектный и уровень планирования и контроля программы измерений.

Раздел «Процесс программной инженерии» рассматривает жизненный цикл программного обеспечения как таковой. Раздел содержит модели и процессы жизненных циклов, нотации для определения процессов, вопросы адаптации и автоматизации. Также подробно рассматриваются различные модели и измерения, связанные с жизненными циклами и процессами ПО.

Отдельного внимания заслуживает раздел «Инструменты и методы программной инженерии», который состоит из двух подобластей: описание программных инструментов для поддержки других областей SWEBOK и различные методы разработки. В документе рассмотрено несколько категорий методов, включая эвристические, математические, методы прототипирования и др.

Десятая область знаний SWEBOK посвящена «Качеству программного обеспечения». Область знаний описывает факторы, влияющие на качество ПО на всех этапах жизненного цикла.

Раздел «Профессиональные практики программного инжиниринга» посвящен различным профессиональным вопросам, будь то вопросы законодательства, этический код, возможности профессиональной сертификации, психология работы в группе и т. п. Отдельное внимание уделяется коммуникационным навыкам специалиста по программному инжинирингу, поскольку работа в команде — реальность, с которой сталкивается каждый профессионал.

Двенадцатый раздел носит название «Экономика программного инжиниринга». В нем освещаются фундаментальные экономические основы программного инжиниринга как сферы деятельности. Раздел включает рассмотрение ключевых финансово-экономических понятий, экономических жизненных циклов, рисков, методов экономического анализа, а также включает ряд практических рекомендаций.

Тринадцатый раздел — это «Основы вычислений». Здесь рассматриваются техники решения соответствующих проблем, изучение уровней абстракции, основы программирования, базовые знания языков программирования, инструменты и техники устранения ошибок, структурирование и представление данных, организация компьютеров, основы ОС, основы БД и управления данными, основы сетей, параллельные вычисления, безопасность и ключевые человеческие факторы.

Раздел «Математические основы» содержит информацию о методах и техниках доказательства, базовой логике, основе вычислений, теории графов и деревьев принятия решений, дискретной вероятности, теории цифр, алгебраической структуре и общей математической грамматике.

И, наконец, в последнем разделе, «Основы инжиниринга», содержатся эмпирические методы и экспериментальные техники, основы статистического анализа в инжиниринге, вопросы разработки, моделирования, симуляции, прототипирования и измерения.

#### 2.2.2. ISO/IEC 12207:2008 (ГОСТ Р ИСО/МЭК 12207—2010)

**Международный стандарт:** ISO/IEC 12207:2008 Information Technology — Software life cycle processes (Информационные технологии. Процессы жизненного цикла программного обеспечения).

**Российский аналог:** ГОСТ Р ИСО/МЭК 12207—2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств».

Базируясь на процессном подходе, ISO 12207 определяет необходимость документирования основных результатов процесса, но не ограничивает их содержание и последовательность, а также не противоречит применению итераций в разработке. Данный стандарт стал основой для дальнейшей детализации в некоторых методологиях разработки ПО (в частности, Rational Unified Process). Однако сам по себе он лишь устанавливает структуру основных, вспомогательных и организационных процессов ЖЦ программных средств, определяя необходимые в их рамках работы и задачи. Таким образом формируется единое понимание жизненного цикла (и единая терминология) между заказчиком, разработчиком/подрядчиком и другими стейкхолдерами. ISO 12207:2008 рассматривает лишь программные средства и соответствующие организационные процессы (рис. 2.5), не рассматривая аппаратную составляющую. В некоторых случаях количество непрограммных частей системы может быть настолько мало, что процессы ИС и ПО могут совпадать.

В РФ был разработан и принят идентичный ISO 12207 стандарт ГОСТ Р ИСО/МЭК 12207—2010. Основной идеей разработчиков ГОСТ 12207 являлось создание единого общекорпоративного стандарта, которым было бы возможно воспользоваться при возникновении любой задачи из тех, которые описаны в документе (будь это обучение пользователей, поставка ПО или любая другая активность в рамках ЖЦ ПО).

Стандарт предполагает, что процессы состоят из работ, для которых определены задачи (а также цели и результаты). Тем не менее, допускается адаптация процессов к особенностям организации (например, при больших масштабах проекта можно изменять состав

определенных задач или работ). Как правило, это возможно сделать в рамках существующих на предприятии процессов.

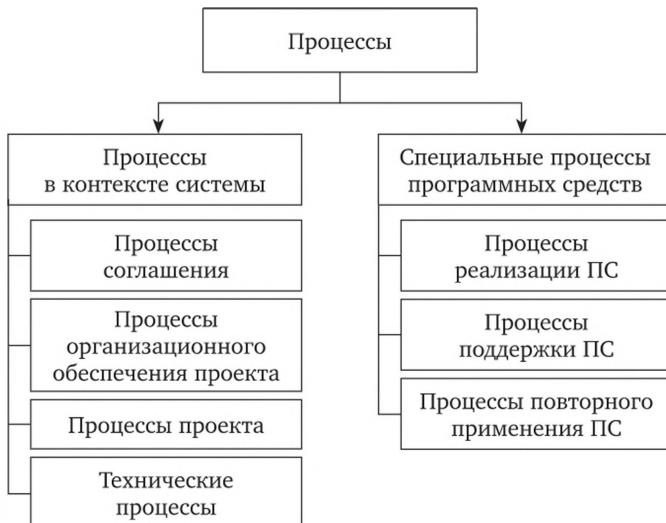


Рис. 2.5. Процессы по ISO 12207:2008

### Пример

В процессе поставки ПО (п. 6.1.2 стандарта) предполагаются следующие виды работ:

- 6.1.2.3.1. Идентификация возможностей
- 6.1.2.3.2. Предоставление заявки поставщикам
- 6.1.2.3.3. Согласование проекта
- 6.1.2.3.4. Выполнение контракта
- 6.1.2.3.5. Поставка и поддержка продукта (услуги)
- 6.1.2.3.6. Закрытие

Несомненно, что у организации-заказчика существуют свои корпоративные регламенты проведения закупок (например, согласования заявок, формирования годовой программы закупок по подразделению ИТ, определения критериев выбора поставщиков услуг и т. д.). Соответственно, организация будет осуществлять приведенные в ГОСТ активности, но на операционном уровне она будет это делать в соответствии со своими внутренними регламентами и процессами.

Приложения стандарта содержат ( помимо эталонной модели процессов, их описаний и видов, а также истории разработки) отдельно выделенный процесс адаптации. Приведенные в нем рекомендации по переходу от стандарта к реалиям определенного предприятия в основном концентрируются на выборе из всего приве-

денного множества процессов тех работ, которые необходимы для реализации конкретного программного проекта. Однако практические рекомендации по организации внедрения ГОСТ 12207—2010 остаются за границами самого документа.

## 2.3. Модели жизненного цикла программного обеспечения

### 2.3.1. Каскадная модель

Повторимся, что каскадная модель жизненного цикла, также называемая моделью «водопада» (англ. waterfall), была разработана еще в 1980-х гг. и на протяжении многих лет считалась стандартом для разработки ПО. Данная модель характеризуется тем, что этапы строго последовательны и переход между ними необратим (рис. 2.6). Это означает, что в рамках каскадной модели переход к следующему этапу (например, от проектирования и сбора требований к разработке и развертыванию) может произойти только по завершении предыдущего этапа. Модель «водопада» была применена одной из первых, и одно из ее основных достоинств в возможности планирования сроков и стоимости каждого этапа. Однако на практике разработка системы почти никогда не проходит строго в соответствии с жесткой, заранее продуманной схемой. В частности, это касается сбора требований, так как реально при старте проекта требования бывают определены только частично и в дальнейшем уточняются, изменяются и дополняются. К тому же, если изначально требования были определены неточно, высока вероятность того, что система не будет удовлетворять потребностям заказчика.

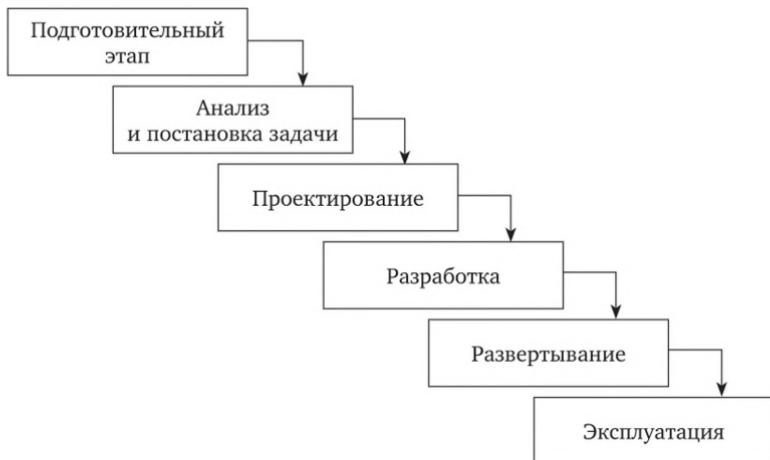


Рис. 2.6. Каскадная модель разработки ПО

Далее сравним преимущества и недостатки каскадных моделей разработки ПО.

Преимущества	Недостатки
Последовательное выполнение этапов проекта в фиксированном порядке. Наличие полной и согласованной документации на каждом этапе	Задержка при получении значимого для заказчика результата. Слишком высокая цена ошибки (даже в модели с промежуточным контролем)

### 2.3.2. Каскадная модель с промежуточным контролем

Чтобы предусмотреть возможность возвращения к предыдущим этапам для внесения определенных изменений и пересмотра отдельных вопросов, в качестве одной из вариаций каскадной модели была создана **каскадная модель с промежуточным контролем**. Она предполагает увеличенное время, отведенное на разработку, за счет проведения промежуточных корректировок между фазами жизненного цикла (рис. 2.7). Это снижает риски получения некачественного продукта на выходе и повышает надежность системы в целом.

Важно отметить, что согласование результатов в двух описанных моделях происходит только по окончании внедрения, а значит, повышается вероятность получения программного продукта, который морально устареет либо не будет востребован рынком. Еще больше увеличиваются риски возможные неточности в исходном техническом задании. В итоге можно говорить о проблеме задержки в получении результата, которая не может быть решена в «каскадном» варианте разработки и внедрения системы.

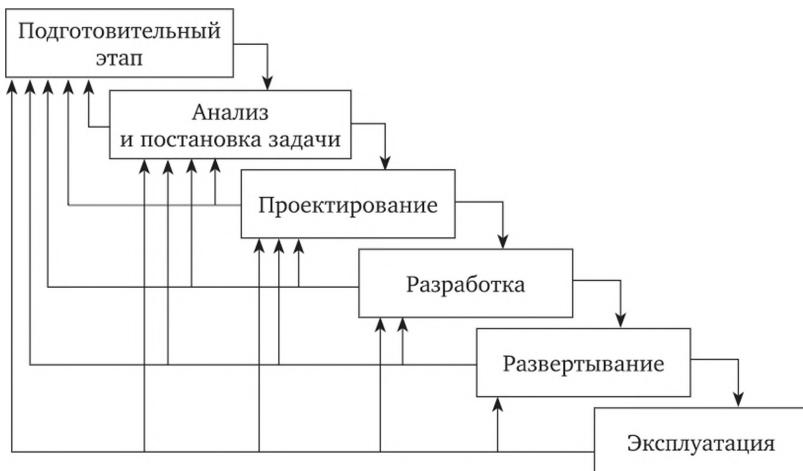


Рис. 2.7. Каскадная модель разработки ПО с промежуточным контролем

### 2.3.3. Метод разработки через тестирование (V-модель)

Для установления связей между этапами каскадной модели иногда используют метод разработки, получивший название V-модель. Приближенная по своей сути к практикам PRINCE2 (широко применяющаяся методология управления программами и проектами в организациях), V-модель разработки через тестирование была разработана еще в конце 1980-х гг. ведомствами Германии и США и до сих пор является стандартом немецких правительственные и оборонных проектов. Основной ее принцип состоит в постепенном возрастании степени детализации проекта с течением времени и одновременном проведении «горизонтальных» итераций. Таким образом (рис. 2.8), результаты каждой из фаз левой стороны буквы V влияют на тестирование и компоновку проекта с правой стороны буквы V: приемо-сдаточные испытания основываются на проведенном анализе требований, интеграционное тестирование — на высокоуровневом описании архитектуры, модульное тестирование — на архитектуре, интерфейсах, алгоритмах и прочих элементах детализированных требований к системе.



Рис. 2.8. V-модель разработки ПО

Важна гибкость данной модели, так как по сути она адаптируема под любой тип организаций. Промежуточные результаты проверяются на ранних стадиях, и минимизация рисков достигается благодаря простому соотнесению фаз. V-модель не рассматривает непосредственно стадию обслуживания и утилизации, учитывая лишь активности по подготовке к ним. Сам по себе этот метод часто в литературе не рассматривается как отдельная модель жизненного цикла, но удобен в установлении соответствия между фазами.

Далее сравним преимущества и недостатки V-модели разработки ПО:

Преимущества	Недостатки
Модель предусматривает верификацию и валидацию всех внешних и внутренних данных. Подходит для проектов, в которых надежность и безопасность важнее стоимости	Модель не поддерживает внесение динамических изменений при разработке. Модель не предназначена для управления параллельными событиями

### 2.3.4. Спиральная модель

Для нивелирования рисков, связанных с вышеописанной проблемой, была создана **спиральная модель** (еще называемая **итерационной**). Фазы жизненного цикла не последовательны, т. е. допустимо (но не обязательно!) начало работ над следующим этапом до завершения предыдущего (рис. 2.9). Таким образом, суть спиральной модели состоит в возможности прохождения всех этапов жизненно-го цикла системы в несколько итераций. При этом на каждой ите-рации создается новый прототип и проверяется актуальность требо-ваний, по которым он создавался, вносятся технические доработки в интерфейс и функциональность. Подобная гибкость позволяет ис-пользовать модель на предприятиях любого масштаба.

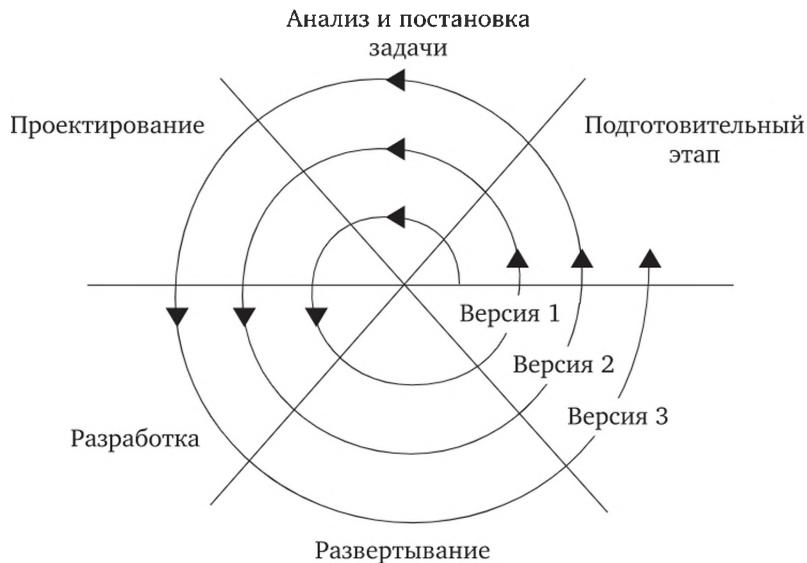


Рис. 2.9. Спиральная модель разработки ПО

Спиральную модель можно применять в областях, где тре-буется завершенность разработки в выпускаемых версиях. В част-

ности, это разработки ПО в области оборонной промышленности, авиастроения, атомной энергетики и другие отрасли, где любая ошибка может стоить человеческих жизней. На практике предполагается, что применение спиральной модели предполагает создание единственной работоспособной версии только в конце всего цикла.

В российской практике часто название спиральной модели приравнивается также к итерационной и эволюционным моделям, хотя западная практика приводит различия в определениях и применении этих моделей. В отличие от спиральной модели, итерационная и эволюционная модель предполагают наличие промежуточных версий, которые может использовать конечный пользователь. Эти модели будут рассмотрены ниже.

Сравним преимущества и недостатки спиральной модели разработки:

Преимущества	Недостатки
Высокое внимание к рискам и анализ рисков позволяют избежать дополнительных затрат. Все фазы разработки взаимосвязаны, предусмотрены итерации	Не может эффективно применяться в сложных проектах с ожиданием работоспособных промежуточных версий. Сложная структура модели для реализации

### 2.3.5. Итерационная модель

Итерационная и инкрементная модели, которые будут рассмотрены далее, на первый взгляд кажутся похожими. Однако существует ряд принципов, которые отличают их друг от друга.

Итерационная модель предполагает, что в начале проекта реализуется часть функциональности, которая становится базой для определения дальнейших требований. Затем процесс повторяется, на каждой фазе к продукту добавляется новая функциональность, а вместе с этим определяются и дальнейшие требования. Не требуется, чтобы каждая версия была идеальной, однако по мере разработки ее соответствие потребностям заказчика должно неуклонно повышаться.

Инкрементная модель ведет команду разработчиков другим путем. Продукт разбивается на набор взаимосвязанных модулей, каждый из которых разрабатывается отдельно. По мере разработки каждого нового модуля он включается в единый продукт.

На рис. 2.10 показаны итерационный и инкрементный процесс «создания» Моны Лизы. В случае использования итерационной модели сначала делается карандашный набросок картины. Затем, на второй итерации, к наброску добавляются базовые цвета. И, наконец, на третьей итерации добавляются детали, картина становится насыщенной и принимает завершенный вид.

Итерационная модель



Инкрементная модель



Рис. 2.10. Различие между итерационной и инкрементной моделями

А в случае использования инкрементной модели картина условно разбивается на несколько частей. Каждая часть — это целостный элемент, который в дальнейшем объединяется с другими элементами для получения конечного результата. Каждая из этих моделей предполагает создание новых версий продукта, которые могут передаваться в эксплуатацию.

Итерационная модель основана на базовой идее разработки ПО через повторяющиеся фазы, каждая из которых длится относительно небольшой промежуток времени. Это позволяет использовать опыт, полученный в ходе выполнения ранних фаз, для более продуктивного выполнения последующих фаз. Результаты выполнения каждой фазы тщательно анализируются для корректировки последующих фаз.

Каждая фаза содержит набор повторяющихся активностей, причем в зависимости от фазы тому или иному виду активностей уделяется разное внимание. К примеру, на ранних фазах большое внимание уделяется сбору требований, тогда как на поздних — реализации, тестированию и развертыванию.

Как правило, итерационная модель применяется при разработке нетиповых систем. Она позволяет быстро подготовить прототип. Часто наличие прототипа крайне помогает в ситуациях, когда требуется разъяснить и уточнить требования, выбрать концептуальное решение или даже в целом определить целесообразность реализации проекта. При этом сам прототип может моделировать либо исключительно пользовательский интерфейс, либо архитектурную сторону системы (логику обработки и хранения данных).

В первом случае важно принимать во внимание возможное сопротивление со стороны пользователей, которые хотят видеть требующуюся именно им функциональность в первом же прототипе, а затем — в первой же предоставленной им версии, а не второй или третьей. Соответственно, на первый план выходит необходимость грамотного управления ожиданиями пользователей.

Это означает, что процесс создания системы и само управление проектом становится более гибким и управляемым, с совершенствованием системы на каждой новой фазе. Уменьшаются риски (в том числе финансовые) для заказчика системы, которые могут отказаться от проекта еще на этапе показа первого прототипа в случае абсолютного его несоответствия ожиданиям и потребностям (либо в случае изменения рыночной ситуации).

Как пример, итерационная модель разработки ПО применяется в корпоративной методологии Rational Unified Process (RUP). На рис. 2.11 отражена итерационная модель с фазами и дисциплинами RUP.

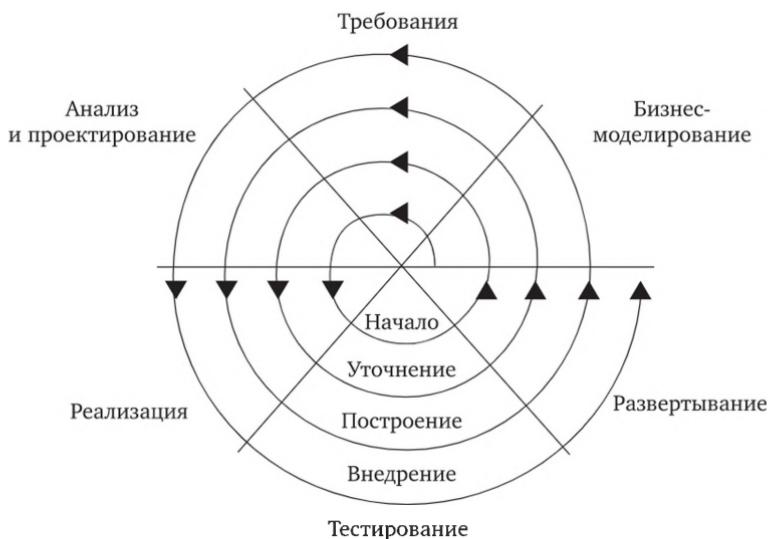


Рис. 2.11. Итерационная модель разработки ПО на примере RUP

Важно отметить, что набирающие сегодня популярность модели гибкой разработки построены на основе итерационной модели. Особенностью этих моделей является то, что команда разработчиков находится в условиях, когда они должны как можно чаще осуществлять сборку версии продукта, которая может быть передана в эксплуатацию. В некоторых методологиях, таких как Scrum, команда разработки должны формировать версию работающего продукта через определенные промежутки времени (например, каждые две недели).

Сравним преимущества и недостатки итерационной модели разработки ПО:

Преимущества	Недостатки
Эффективная обратная связь команды разработчиков с заказчиками. Раннее обнаружение конфликтов между требованиями, моделями и реализацией продукта. Затраты равномерно распределены по всему проекту, а нагрузка — по всем участникам	Требуется сильная вовлеченность заказчика в процесс разработки ПО. Нестабильный список требований. Высокая сложность планирования и повышенная нагрузка на менеджера проекта

### 2.3.6. Инкрементная модель

Инкрементная модель разработки ПО — это метод разработки программного обеспечения, в котором продукт разрабатывается, реализуется и тестируется эволюционно (с небольшими добавлениями за промежуток времени). Продукт считается завершенным в случае, когда он соответствует всем предъявляемым к нему требованиям.

Продукт декомпозируется на набор модулей, каждый из которых разрабатывается отдельно. Каждый модуль предоставляется клиенту по завершению работ над ним. Подобный подход позволяет сделать разработку наиболее удобной и полезной для заказчика, а также избежать излишних затрат времени. Также инкрементная модель может снизить «травматический» эффект для предприятия от внедрения готового продукта целиком.

Первыми разрабатываются модули с критически важной функциональностью, набор которых согласован с заказчиком. Когда некоторая совокупность модулей разработана, формируется первая (и последующие) версии, а вместо нереализованной функциональности в текущей версии используются так называемые «заглушки». Далее «заглушки» заменяются готовыми модулями. Подобный подход позволяет совершать поставки каждого готового модуля заказчику.

Сравним преимущества и недостатки инкрементной модели разработки ПО:

Преимущества	Недостатки
Ускоряет вывод новой версии продукта на рынок или предоставление новой версии продукта заказчику. Функциональность продукта расширяется при выполнении каждого инкремента	Не предусматриваются итерации внутри каждого инкремента. Для определения инкрементов функциональность продукта должна быть определена в начале жизненного цикла

### 2.3.7. Эволюционная модель развития

В изложенном ранее материале были рассмотрены модели разработки программного обеспечения. На практике, если речь идет не только о создании, но и развитии ПО, а также информационной системы в целом, такой подход часто называется эволюционной моделью развития. В работах Тома Гилба в 1976 г. был впервые использован этот термин. «Эволюция — прием, предназначенный для создания видимости стабильности. Шансы успешного создания сложной системы будут максимальными, если она реализуется в серии небольших шагов и, если каждый шаг заключает в себе четко определенный успех, а также возможность отката к предыдущему успешному этапу в случае неудачи. Перед тем, как пустить в дело все ресурсы, предназначенные для создания системы, разработчик имеет возможность получать из реального мира сигналы обратной связи и исправлять возможные ошибки в проекте...» Эволюционная модель может сочетать в себе элементы каскадной и итерационной моделей на отдельных этапах, а также может применять метод прототипирования. В процессе эксплуатации первоначальной функциональности программных продуктов возникают новые и новые требования, которые реализуются как в существующем программном обеспечении, так и в создании новых программных продуктов. В дальнейшем осуществляется интеграция вновь созданных программных продуктов в общую функциональность информационной системы.

Отличительной особенностью такой модели является прохождение информационной системы через фазу эксплуатации, что будет отличать такую модель от модели разработки. Версии, которые обозначены на этой модели, являются законченными версиями, которые переданы в эксплуатацию.

Важно отметить, что в международной и российской практике все рассмотренные модели очень часто сводят к двум типам моделей: каскадной и итерационной, которую в российской практике часто называют спиральной. В разных организациях специалисты используют различный профессиональный сленг, который иногда вносит путаницу в названия этих моделей. В заключение автор хотел бы процитировать слова Мартина Фаулера, написанные им в 2004 г.: «Итеративную разработку называют по-разному: инкре-

ментальной, спиральной, эволюционной и постепенной. Разные люди вкладывают в эти термины разный смысл, но эти различия не имеют широкого признания и не так важны, как противостояние итеративного метода и метода водопада».

## **Контрольные вопросы и задания**

1. Назовите этапы, которые включает в себя жизненный цикл информационной системы.
2. Каковы основные модели ЖЦИС?
3. Как выглядит жизненный цикл ИС в COBIT?
4. Какие области знаний рассматриваются в SWEBOK?
5. Какие существуют стандарты жизненного цикла ИС?
6. Какие существуют основные модели ЖЦПО?
7. В чем различие между каскадной и спиральной моделями ЖЦПО?
8. В каких случаях и почему применяется каскадная модель ЖЦПО с промежуточным контролем?
9. Когда эффективнее всего может быть применена V-модель разработки ПО?

## **Рекомендуемая литература**

1. Адизес, И. К. Управление жизненным циклом корпораций / И. К. Адизес. — Санкт-Петербург : Питер, 2008.
2. COBIT 5.0. — ISACA, 2012.
3. Guide to Software Engineering Body of Knowledge (SWEBOK): Version 3.0. — IEEE Computer Society, 2014.

# **Тема 3**

## **ФАЗЫ ЖИЗНЕННОГО ЦИКЛА ИНФОРМАЦИОННЫХ СИСТЕМ И СПЕЦИФИКА КАЖДОЙ ИЗ НИХ**

---

В этой теме рассмотрены наиболее общие фазы, характерные для жизненного цикла информационных систем. Изложение содержания фаз выполнено на примере каскадной модели, но содержание фаз также может быть отнесено к спиральной модели. Каждый этап содержит типовые виды деятельности, которые также отражены в этой теме.

После изучения данной темы студент будет:

**знатъ**

- содержание основных этапов жизненного цикла информационных систем;
- специфику каждого этапа ЖЦИС;
- типовые виды деятельности для каждого этапа ЖЦИС;

**уметь**

- производить планирование этапов ЖЦИС и типовых видов деятельности;

**владеть навыками**

- выполнения типовых видов деятельности для каждого этапа ЖЦИС.
- 

### **3.1. Подготовительный этап**

На самом первом этапе в обязательном порядке проводится обследование для сбора информации о целях и задачах компании в целом и внедряемой системы в частности, а также для получения данных об основных бизнес-процессах и покрывающем их ландшафте информационных систем (для определения места внедряемого решения в нем).

Важно верно определить:

- организационный объем проекта (затрагиваемые реализацией проекта подразделения);
- наличие зависимостей от других проектов. Например, если в этот же период внедряются другие системы и в обоих проектах задействованы одни и те же специалисты ИТ-департамента, одно обо-

рудование, или если они оба требуют незапланированных в исходном бюджете дополнительных расходов;

- планируемые финансовые, инфраструктурные и человеческие ресурсы, бюджет проекта (включая необходимые закупки, например, лицензии от поставщика ПО, и планируемые затраты при необходимости оплаты услуг подрядчика по сопровождению, обслуживанию серверов и т. д.).

Принято различать две категории обследования:

- экспресс-обследование, имеющее своим результатом:
  - краткое описание текущей бизнес-модели предприятия,
  - коммерческое предложение со сформулированными проблемами, а также ориентировочными сроками и бюджетом работ по дальнейшему информационному обследованию и непосредственно внедрению системы;
- информационное обследование, по результатам проведения которого должны быть подготовлены:
  - полное описание текущей бизнес-модели предприятия заказчика,
  - утвержденный план работ по разработке и внедрению (включая состав группы внедрения с обеих сторон).

По результатам информационного обследования заказчик должен принять все организационные меры для обеспечения реализации проекта. В частности, это относится к внутренним приказам и документам предприятия заказчика (для назначения ответственных за предоставление информации и взаимодействие с проектной командой в рамках внедрения).

### 3.1.1. Экспресс-обследование

Основными целями экспресс-обследования являются:

- выявление основных проблем и «узких мест» по участкам бизнес-процессов;
- изучение и укрупненное описание бизнес-процессов в привязке к подразделениям предприятия с распределением ответственности между ними;
- изучение и описание базы нормативно-справочной документации, используемой на предприятии;
- изучение пожеланий по совершенствованию системы управления предприятием и формирование предложений по решению проблем, стоящих перед предприятием;
- подготовка предложений по срокам и стоимости проведения полномасштабного информационного обследования требуемых бизнес-процессов предприятия.

Таким образом, экспресс-обследование — это своеобразное знакомство потенциального заказчика и исполнителя с работой друг

- документооборот;
- бизнес-процессы основной деятельности,
- автоматизация и пр.;
- актуальные вопросы развития;
- дополнительные пожелания заказчика, примечания;
- данные о сотруднике, ответственном за заполнение анкеты.

В результате основного анкетирования какие-то моменты могут быть до конца не выяснены, где-то могут возникнуть противоречия. Поэтому необходима обратная связь с анкетируемыми для уточнения ранее полученных данных. При этом крайне важно, что консультантам должны предоставляться исключительно точные данные в пределах предварительно оговоренных руководством предприятия информационных границ.

Продолжительность обратной связи по анкетам — 3—7 дней. Основным результатом проведения экспресс-обследования является отчет, формируемый консультантами в течение 1—2 рабочих недель.

В итоге отчет о результатах проведенного экспресс-обследования будет описывать следующие направления:

- 1) краткая характеристика объекта исследования; основные виды деятельности;
- 2) схема и краткое описание организационной структуры предприятия;
- 3) основные функции обследуемых структурных подразделений с группировкой по бизнес-процессам;
- 4) краткое описание существующих на предприятии бизнес-процессов, принципов взаимодействия между подразделениями предприятия заказчика;
- 5) описание информационного взаимодействия между подразделениями;
- 6) описание основных проблем и слабых мест на обследуемых участках бизнес-процессов;
- 7) пожелания заказчика по совершенствованию системы управления предприятием;
- 8) предложения о дальнейшем сотрудничестве, рекомендации по решению выявленных проблем и ориентировочная стоимость работ.

В результате экспресс-обследования консультанты получают знания о предприятии заказчика, включая представление о его бизнес-модели. В свою очередь, заказчик вместе с отчетом получает краткое описание бизнес-модели с ценной информацией о выявленных возможностях для улучшения и, конечно, коммерческое предложение с указанием ориентировочных сроков и стоимости работ по полному информационному обследованию и проведению работ по созданию и внедрению информационной системы на предприятии.

### 3.1.2. Технико-экономическое обоснование

Ключевая проблема, с которой сталкиваются как обосновывающие необходимость внедрения ИС, так и принимающие решение — невозможность объективно предсказать и рассчитать совокупный эффект от самой системы и от ее взаимодействия с другими программными решениями, входящими в архитектуру предприятия. Именно поэтому ТЭО чаще всего превращается в декларативный документ, содержащий цели и задачи внедрения системы, ожидаемые результаты и возможные риски, по сути представляя собой краткую «бизнес-версию» технического задания.

---

**ТЭО, технико-экономическое обоснование** (англ. *business case*) — экономическое, бизнес-ориентированное обоснование выгод от ИТ-инвестиций (например, в информационную систему) и прогнозирование объема этих выгод.

---

Содержание ТЭО в разных организациях различается, однако в классическом случае оно состоит из следующих разделов.

1. Резюме для руководства.

Основные показатели эффективности проекта.

#### Пример

- Чистый денежный поток (млн руб.).
- Суммарный денежный поток (млн руб.).
- NPV, чистая приведенная стоимость (млн руб.).
- IRR, внутренняя норма доходности, %.
- Срок окупаемости, лет.

2. Описание проекта.

2.1. Предпосылки реализации проекта.

Документы, нормативно-правовые акты, события.

2.2. Мировой и отраслевой опыт реализации подобных проектов.

Аналогичные проекты, выполненные в схожих предметных областях, их достижения и ошибки.

2.3. Ключевые заинтересованные стороны.

Заказчики проекта: функциональные подразделения различных уровней иерархии и конкретные сотрудники (в том числе акционеры и топ-менеджмент), в чьих интересах (с учетом чьих потребностей) планируется реализация проекта.

2.4. Организационный масштаб проекта.

Охватываемые проектом подразделения и сотрудники.

2.5. Проектные зависимости.

Другие проекты, результаты которых используются при реализации данного проекта. И напротив, проекты, реализация которых возможна только после окончания текущего.

## 2.6 Основные области достигаемых бизнес-выгод.

### Пример

- Сокращение рисков несоблюдения установленных сроков и объемов продукции.
- Исключение рисков несоответствия местных нормативов требованиям корпоративных и отраслевых регламентирующих актов.
- Повышение качества производимой продукции.
- Сокращение расходов на создание, развитие и сопровождение ИТ-инфраструктуры.
- Обеспечение качества и достоверности информации.

---

### 3. Оценка коммерческой целесообразности реализации проекта.

#### 3.1. Затраты и экономика проекта.

Покупка аппаратного и программного обеспечения, оплата услуг подрядчиков, операционные расходы, обслуживание оборудования и прочие статьи затрат.

#### 3.2. Основные области выгод. Экономические выгоды.

Оценка чистого денежного потока, внутренней нормы доходности, срока окупаемости.

#### 3.3. Структура финансирования проекта.

Определение источников финансирования (внешние инвесторы, бюджет подразделения, ...).

#### 4. Матрица основных проектных рисков.

Ключевые ассоциированные с проектом риски, сгруппированные по категориям (например, вероятность реализации риска и размер возможных убытков).

#### 3.1.3. Оценка целесообразности проекта (TELOS)

Прежде чем приступить к разработке, важно еще раз убедиться, что проведенных работ по анализу, обоснованию и проектированию системы достаточно для перехода к следующему этапу.

Аббревиатура TELOS расшифровывается как Technological, Economical, Legal, Operational, Scheduling.

Рассмотрение TELOS позволяет удостовериться, что проект по реализации системы реалистичен и имеет значительный потенциал.

- Технические (*Technological*) — проверка наличия инфраструктурных ресурсов организации, а также квалифицированного персонала и проектного опыта для создания и поддержки системы. Описание потенциальных сложностей и возможностей их разрешения.
- Экономические (*Economical*) — определение ожидаемых экономических выгод от реализации информационной системы.

### Пример

---

Снижение времени обработки транзакции на 18 %.

- Юридические (*Legal*) — определение, насколько все технологические спецификации и существующая функциональность обеспечивают требования к безопасности данных и их предоставлению в случае необходимости.

### Пример

Любые системы обработки персональных данных должны обеспечивать минимальный (закрепленный регламентами и законодательством) уровень защиты информации.

- Операционные (*Operational*) — покрытие сформированных на этапе анализа требований заинтересованных сторон подготовленным проектом. Определение, насколько система соответствует возможностям и потребностям бизнеса с точки зрения сроков проекта, встраивания в бизнес-процессы в текущих рыночных условиях к моменту создания системы. Рассмотрение критериев надежности, возможности поддержки, удобства интерфейса, продуктивности, стабильности и прочих факторов создания системы.

- Сроки реализации (*Scheduling*) — оценка сроков создания системы для определения возможных изменений во внешней среде за время реализации проекта. Оценка, насколько реалистичны поставленные крайние сроки сдачи системы в эксплуатацию, а также промежуточные вехи.

#### 3.1.4. Выбор программного решения

После сбора требований, на подходе к этапу проектирования, проектной команде предстоит определиться, каким образом подойти к выбору программного решения: необходимо ли его разрабатывать с нуля либо приобрести уже существующий на рынке и зарекомендовавший себя продукт.

Заказные КИС являются уникальными, создаются исключительно для нужд и бизнес-целей конкретной компании (чаще всего для конкретной узкой специализации или автоматизации одной функции). Как следует из названия, такие системы не имеют аналогов и не подлежат дальнейшему тиражированию (а часто и масштабированию). Обычно ИС такого класса не имеют прототипов. Иногда использование прототипов возможно, однако требуется внесение серьезных изменений качественного характера.

К системам данного типа мы будем относить как самостоятельную разработку внутренними ИТ-подразделениями компаний, так и действительно заказную разработку, осуществляемую силами подрядчиков или внешних компаний.

Компании, рассматривающие разработку бизнес-приложений с нуля как подходящий для себя вариант, часто формулируют обоснование своего выбора следующим образом.

- Юридические (*Legal*) — определение, насколько все технологические спецификации и существующая функциональность обеспечивают требования к безопасности данных и их предоставлению в случае необходимости.

### Пример

Любые системы обработки персональных данных должны обеспечивать минимальный (закрепленный регламентами и законодательством) уровень защиты информации.

- Операционные (*Operational*) — покрытие сформированных на этапе анализа требований заинтересованных сторон подготовленным проектом. Определение, насколько система соответствует возможностям и потребностям бизнеса с точки зрения сроков проекта, встраивания в бизнес-процессы в текущих рыночных условиях к моменту создания системы. Рассмотрение критериев надежности, возможности поддержки, удобства интерфейса, продуктивности, стабильности и прочих факторов создания системы.

- Сроки реализации (*Scheduling*) — оценка сроков создания системы для определения возможных изменений во внешней среде за время реализации проекта. Оценка, насколько реалистичны поставленные крайние сроки сдачи системы в эксплуатацию, а также промежуточные вехи.

#### 3.1.4. Выбор программного решения

После сбора требований, на подходе к этапу проектирования, проектной команде предстоит определиться, каким образом подойти к выбору программного решения: необходимо ли его разрабатывать с нуля либо приобрести уже существующий на рынке и зарекомендовавший себя продукт.

Заказные КИС являются уникальными, создаются исключительно для нужд и бизнес-целей конкретной компании (чаще всего для конкретной узкой специализации или автоматизации одной функции). Как следует из названия, такие системы не имеют аналогов и не подлежат дальнейшему тиражированию (а часто и масштабированию). Обычно ИС такого класса не имеют прототипов. Иногда использование прототипов возможно, однако требуется внесение серьезных изменений качественного характера.

К системам данного типа мы будем относить как самостоятельную разработку внутренними ИТ-подразделениями компаний, так и действительно заказную разработку, осуществляемую силами подрядчиков или внешних компаний.

Компании, рассматривающие разработку бизнес-приложений с нуля как подходящий для себя вариант, часто формулируют обоснование своего выбора следующим образом.

1. «Не существует двух одинаковых компаний, а значит, стандартные решения не смогут целиком отразить специфику компаний».

2. «Стандартные системы содержат избыточный набор функций, за которые приходится платить».

3. «Стоимость самостоятельной разработки ниже стоимости лицензий и услуг по системной интеграции и консалтингу».

Разработка таких КИС сопряжена с дополнительными рисками в получении необходимых результатов.

В случае самостоятельной разработки важно помнить о том, что компетенции для работы с подобными системами (и (или) их создания) часто являются уникальными, а значит, риск ухода ключевых сотрудников, работавших с ними, тоже нужно принимать во внимание. В разрабатываемых «для себя» приложениях часто обнаруживаются ошибки, вызванные недостаточно тщательно проведенным тестированием. В силу того что в большинстве компаний в штате ИТ-специалистов нет отдельных ответственных за версионность, бета-тестирование и другие аспекты процесса внедрения, основная часть разработчиков фокусируется исключительно на реализации функциональных возможностей и производительности системы, не уделяя должного внимания «чистоте» и безопасности кода.

В случае заказной разработки часто встречаются случаи неверного толкования технического задания (или ошибок при его составлении), высокая оценка стоимости реализации проекта, коммуникационные проблемы, неполное документирование системы.

По всем вышеописанным причинам, а также в силу развитости современной индустрии решений для автоматизации деятельности компании на сегодняшний момент заказная разработка уже далеко не столь распространена, как 15 лет назад. Ранее популярность подобных решений объяснялась тем, что адаптируемых или «коробочных» КИС для отраслей/бизнесов определенного масштаба почти не существовало. Сейчас почти любой проект разработки и внедрения предполагает лишь доработку стандартной или отраслевой конфигурации под согласованные с заказчиком требования.

Тиражируемые КИС иногда еще называют коробочными или шаблонными решениями. Однако это не совсем корректно в силу адаптации подобных систем к нуждам конкретного предприятия. Как правило, в основу подобных систем заложены общие процессы и свойства предприятий (примерно одного размера и работающих в одной отрасли). Производитель ИС такого типа при разработке использует опыт собственных проектов и «лучшие практики» (*best practices*) отрасли. Соответственно, одним из критериев при выборе ИС становится репутация и опыт производителя системы, а также общий масштаб системы и ее соответствие специфике предприя-

тия, несмотря на то что способность к адаптации так или иначе предусмотрена в большинстве информационных систем.

Процесс адаптации тиражируемых КИС может быть как одноголовневым, так и двухуровневым. В первом случае настройку и изменение параметров исходной системы, купленной у предприятия-производителя, проводит непосредственно проектная команда по внедрению — либо со стороны исполнителя (консалтинговых компаний и системных интеграторов), либо ИТ-подразделение компании собственными силами.

#### Пример

---

Закупка пакета лицензий SAP Business One с услугами по интеграции непосредственно у российского офиса SAP AG.

---

Во втором случае (особенно в случае зарубежных КИС) российские организации — системные интеграторы на основе собственного опыта в национальной отраслевой специфике с учетом особенностей российского законодательства вносят модификации в исходное решение. Они же занимаются дистрибуцией созданного отраслевого решения (вместе с услугами по интеграции).

#### Пример

---

Партнерство NaviCon Group и Microsoft, в рамках которого на базе решения Microsoft Dynamics NAV разработан локализированный продукт NaviCon Trade, автоматизирующий управление сбытом, заказами и взаиморасчетами с клиентами в торговых предприятиях.

---

Резюмируя сказанное, самостоятельная (заказная) разработка предполагает равномерное распределение внимания между всеми фазами жизненного цикла (при написании кода с нуля), в то время как доработки и настройка тиражируемых информационных систем концентрируются на сборе требований, сравнении их с имеющимся решением и последующем развертывании ИС.

## 3.2. Анализ и постановка задачи

Существует определенный набор ключевых мероприятий, обязательное проведение которых может повысить вероятность успеха проекта. Уже на стадии постановки задачи должны быть определены цели этапов проекта и самого проекта, включая критерии их выполнения. При этом данные цели должны быть сформулированы предельно четко и ясно (по возможности удовлетворяя критериям SMART: конкретность, измеримость, достижимость, актуальность, установка конкретного срока).

## Пример

---

Установка бухгалтерской программы (просто ради ее установки) или разворачивание технических средств не может быть самодостаточной целью внедрения системы, а будет являться лишь инструментом для достижения настоящей цели. Таковой, к примеру, может являться автоматизация бухгалтерского учета (для учета операций производственно-экономической деятельности).

---

Прежде чем перейти к рассмотрению основных активностей, подчеркнем, что следующая часть документа основывается на опыте внедрения различных программных систем, в частности — интегрированной системы управления, о которой уже было сказано ранее, и рекомендациях стандартов и сводов знаний.

---

**Под интегрированной системой управления** понимается система управления, использующая совокупность современных программно-аппаратных средств, информационных технологий и экономико-математических методов для регулярного решения задач управления производственно-экономической деятельностью предприятий и коммерческих организаций.

---

Интегрированная система управления является сложной системой управления, включающей в свой состав отдельные модули (подсистемы).

---

**Модуль** — часть системы, выделенная по определенному признаку, отвечающему конкретным целям и задачам управления. В рамках этих задач он может рассматриваться как самостоятельная система.

---

Перейдем к основным этапам, составу, содержанию и порядку выполнения работ по внедрению программных продуктов в рамках реализуемых организациями ИТ-проектов.

### 3.2.1. Информационное обследование предприятия

**Анкетирование.** Основными целями информационного обследования являются:

- детальное описание требуемых бизнес-процессов;
- подготовка предприятия заказчика к внедрению системы;
- уточнение графика выполнения работ по внедрению системы по срокам и видам работ.

Информационное обследование осуществляется по определенной методике, предусматривающей три ключевых этапа:

- сбор первичной информации (анкетирование, интервью и др.);
- систематизация и анализ информации;

- представление собранных данных (в формате отчета) для согласования результатов обследования с заказчиком проекта.

Важно отметить, что именно эти виды деятельности тесно пересекаются с получением информации на всех стадиях ЖЦИС, которое будет тем быстрее и эффективнее, чем более тщательная работа будет проведена на первом этапе.

Основные действия этапа:

- определение даты начала работ, назначение руководителя работ и ответственных представителей заказчика по каждому подразделению предприятия, в которых проводятся работы по обследованию бизнес-процессов, издание приказа о начале работ;
- определение основных характеристик предприятия командой исполнителя;
- утверждение приказа об информационном обследовании бизнес-процессов предприятия с назначением руководителя проекта от заказчика;
- формирование графика проведения собеседований по анкетам с указанием даты, места и участников встречи;
- проведение интервьюирования для описания бизнес-процессов (включая сквозные) с указанием ответственных;
- анкетирование основных подразделений;
- подготовка отчета для заказчика о результатах обследования с его последующим согласованием и утверждением.

Среди предоставляемых заказчиком анкет информационного обследования:

- опросники для высшего руководства;
- опросники руководителей подразделений;
- опросники специалистов: ведущих специалистов, сотрудников отделов, служб, главного бухгалтера, сотрудников бухгалтерии (в случае внедрения финансовых или интегрированных с ними систем).

Анкета для высшего руководства предназначена для определения проблемных зон в функциональном взаимодействии подразделений и информационном обмене, затрудняющих процесс принятия управленческих решений руководством предприятия. Данная анкета может быть заполнена генеральным директором, финансовым директором, коммерческим директором, техническим директором и другими руководителями аналогичного ранга. Внимательное заполнение анкеты, а также предоставление полной и точной информации поможет консультантам лучше структурировать те проблемы предприятия, которые могут быть решены с их помощью.

Анкеты руководителей бизнес-подразделений создаются в целях определения проблемных зон в информационном обмене и функциональном взаимодействии подразделений. С точки зрения определения текущей ситуации и выявления потенциальных областей для улучшения они являются наиболее показательными.

## Пример

---

Руководители департаментов или функциональных направлений могут обозначить основные типы проблемных ситуаций, связанных с механизмом коммуникаций между подразделениями:

- дублирование функций;
- перегруженность работой;
- неотрегулированный механизм обмена информацией, прохождения документов, заказов;
- недостаток информации для принятия оперативных решений по ряду вопросов;
- сложности при запросе дополнительных данных;
- отсутствие оперативной информации;
- задержка в предоставлении отчетов, различных данных; предоставление некачественной информации (неточности, произвольная форма) и т. п.

Могут быть также определены некоторые *проблемные участки работы*, например составление отчета о работе подразделения за месяц. Если данная функция не автоматизирована, информация собирается вручную и часть информации поступает с задержками и неточностями. Это повышает напряженность во взаимоотношениях между сотрудниками, отвечающими за разные участки работы подразделения, и требует много времени на обработку поступившей информации.

Помимо анкет и интервью с ответственными от подразделений, в качестве источников информации используются также следующие передаваемые заказчиком документы:

- информационный буклет (рекламный проспект) об истории и основных направлениях деятельности предприятия;
- схема организационной структуры;
- положения о подразделениях, отделах;
- учетная политика предприятия;
- стандарты качества;
- стратегия развития;
- документация по используемому ПО.

---

И наконец, анкеты для специалистов призваны выявлять зоны ответственности сотрудников, основные «узкие места» процессов, с которыми им приходится сталкиваться при работе. Определяется, каким образом специалисты организуют свою ежедневную деятельность, с какими системами и данными работают. Также вопросы анкеты помогают сформировать точку зрения сотрудников на потенциальные возможности улучшения процессов как в области их ответственности, так и в масштабах всей организации.

**Прочие методы сбора информации.** Как и в случае экспресс-обследования, помимо интервью с ответственными от подразделений и анкет на этом этапе используется значительный объем документальных источников информации:

- все запрашиваемые первичные документы предприятия;
- стандарты предприятия;
- положения о подразделениях, должностные инструкции, штатное расписание, стандарты предприятия;
- нормативно-справочная информация;
- бизнес-план и (или) стратегия развития предприятия;
- имеющиеся на предприятии аналитические отчеты, результаты маркетинговых исследований и т. п.;
- финансовые отчеты, балансы и т. п.

В результате интервью должны быть:

- выявлены все внешние объекты, с которыми предприятие взаимодействует, технологии взаимодействия предприятия с этими объектами, а также информационные и материальные потоки, обеспечивающие эти взаимодействия;
- выявлены реальные технологии работы предприятия;
- определены реальные функции подразделений и их взаимосвязи и взаимозависимости, информационные потоки;
- идентифицированы и специфицированы все информационные хранилища, в том числе и бумажные (карточки, архивы);
- оценены используемые на момент проведения обследования средства автоматизации и ПО как в структурных подразделениях, так и на предприятии в целом.

По результатам проведенных работ исполнителем должна быть подготовлена и передана заказчику бизнес-модель предприятия (с диаграммами сквозных бизнес-процессов), а также сформирован план мероприятий по внедрению системы. В типовом виде он содержит:

- цели и задачи внедрения;
- организационные мероприятия;
- рекомендации по реорганизации необходимой организационной структуры для поддержки системы;
- распределение функционально-рабочих мест системы у предприятия-заказчика;
- требования к аппаратному и системному ПО локальной сети предприятия-заказчика;
- пуско-наладочные работы (состав, сроки);
- этапы ввода системы в эксплуатацию;
- ориентировочный план-график проведения работ по внедрению системы на предприятии заказчика (в относительных датах);
- ориентировочную стоимость поставки и внедрения системы.

И конечно, переход к следующему этапу невозможен, если не определены цели внедрения системы. Независимо от того, приобретается ли «коробочное» решение или начинается разработка с нуля, очень важно на данном этапе уточнить с заказчиком и спонсором

проекта список основных стейкхолдеров<sup>1</sup> и достичь в переговорах с ними компромисса, так как у всех заинтересованных лиц могут быть очень разные представления как о назначении и целевом использовании системы, так и о ее масштабах. В дальнейшем, в ходе фазы проектирования, полученная информация уточняется и становится основой для формирования технического задания. ТЗ должно быть максимально проработанным со всеми участниками проекта, ведь именно на первых двух стадиях совершается большинство ошибок, которых можно избежать при применении более формальных методов анализа.

### 3.2.2. Описание бизнес-процессов

На этом этапе формируются описания законченных бизнес-процессов, участие в которых могут принимать одно или несколько подразделений.

#### Пример

Определяются основные составляющие процесса.

##### Производство:

- планирование выпуска и производственных мощностей;
- планирование обеспечения ресурсами (ТМЦ, услуги и работы, трудовые ресурсы, энергия и пр.);
- организация производственного процесса (подготовка документов для производства);
- учет затрат на производство (основное, вспомогательное);
- учет выпуска готовой продукции и незавершенного производства;
- контроль над выполнением производственной программы (диспетчеризация производства);
- анализ производства (отчетность).

Основная цель моделирования процессов — их документирование и последующее осуществление функционального анализа на предмет поиска «узких» мест процессов и возможностей для их совершенствования. Важно, что эта деятельность ни в коем случае не является «моделированием ради моделирования». В дальнейшем полученные результаты будут использоваться при формировании функциональных требований к системам, а также при реинжиниринге бизнес-процессов.

Далее отображаются основные действующие лица, документооборот, поддерживающие информационные системы и прочие элементы процессов, уже в графическом виде (в том числе UML-диаграммы, нотации BPMN и EPC). Подобное моделирование отдель-

<sup>1</sup> Стейкхолдер (англ. *stakeholder*), заинтересованная сторона, причастная сторона — организация или физическое лицо, имеющая права, долю или интересы относительно системы или ее свойств.

ных предметных областей и процессов позволяет сформировать системное представление о деятельности организации.

### 3.2.3. Сбор требований

Данная стадия подразумевает несколько аспектов — от анализа задачи и специфики деятельности предприятия до анализа требований к самой системе. Проектные спецификации и первые наброски технического задания на данном этапе составляются по результатам проведенных интервью и в дальнейшем уточняются для проведения оценки сроков, стоимости и содержания проекта в части функциональности системы.

Чтобы правильно сформировать стоящие перед целевой информационной системой задачи и определить ее ключевые функции, необходимо проведение объективного и всестороннего сбора требований.

В IEEE Standard Glossary of Software Engineering Terminology (1990) понятие «требование» трактуется следующим образом.

---

**Требование** — это:

- 1) условия или возможности, необходимые пользователю для решения проблем или достижения целей;
  - 2) условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;
  - 3) документированное представление условий или возможностей для пунктов 1 и 2.
- 

По сути, требования представляют собой определенный набор входных данных, на основании которых проводится проектирование ИС. Существует множество источников подобных данных (ведь в конечном итоге необходимо создать систему для большого числа пользователей с различными задачами), в связи с чем исходная информация является достаточно противоречивой, неструктурированной и изменяющейся во времени. Однако без этого этапа начать работу невозможно, так как именно согласие заказчика и разработчика-исполнителя по вопросам объема и содержания работ, временных и финансовых ограничений является критическим для выполнения проекта автоматизации.

**FURPS+**. Для охвата всех аспектов проектируемой ИС используется специальная классификация уровней требований (FURPS+, изначально предложена в начале 1990-х гг. специалистом Hewlett Packard Робертом Грейди)<sup>1</sup>.

---

<sup>1</sup> Академия Microsoft. Анализ требований к автоматизированным информационным системам // INTUIT.RU. 2007. URL: <http://www.intuit.ru/studies/courses/2188/174/lecture/4714?page=2>.

Сокращение *FURPS* является аббревиатурой и в переводе на русский включает пять следующих аспектов: Функциональность, Удобство использования, Надежность, Производительность и Поддержку.

Рассмотрим каждый из уровней более детально.

**Functionality** (функциональность):

- функциональные возможности системы (это основные функции системы, которые необходимы пользователю).

**Usability** (удобство использования — характеристики пользовательского интерфейса):

- интуитивность;
- эстетичность;
- мультимедийность (использование анимации, графики, видео и других форматов контента);
- защита от ошибок (человеческого фактора);
- удобство обучения;
- полнота системной документации и руководств пользователя;
- легкость получения справочной информации.

**Reliability** (надежность):

- доступность (минимально необходимое время бесперебойной работы системы — к примеру, 24 × 7);
- точность расчетов;
- отказоустойчивость;
- восстанавливаемость системы (частота резервного копирования данных).

**Performance** (производительность):

- пропускная способность (число одновременно работающих пользователей);
- время отклика системы (время реакции системы на совершенное пользователем действие);
- время восстановления;
- масштабируемость (к примеру, при увеличении числа пользователей в случае внедрения в других филиалах);
- потребление ресурсов (к примеру, оперативной памяти).

**Supportability** (поддерживаемость):

- ремонтопригодность;
- адаптируемость (скорость приспособления к условиям работы в конкретной среде);
- совместимость;
- легкость установки;
- конфигурируемость (гибкость и оперативность изменения параметров системы, добавления новых ролей, ограничения доступа, изменения функций);
- локализация (поддержка языков, валюты, настроек времени и дат выбранной страны);

- поддержка стандартов и требований регуляторов (например, поддержка SCORM — международного стандарта обмена электронными курсами как требование к системе дистанционного обучения) и т. д.

#### **Дополнительные ограничения (+):**

- ограничения проектирования, *design*, например использование реляционной БД в качестве основной;
- ограничения разработки, *implementation*, например ориентация на выбранную методологию разработки (RUP/MSF), языки программирования (C#, Python, ...);
- ограничения на интерфейсы, *interface* (накладываются пользователями или другими системами), например использование определенных форматов данных;
- физические ограничения, *physical* (например, к аппаратным средствам и окружающей среде).

**Требования по Карлу Вигерсу.** Все вышеописанные требования (кроме первого, функциональности) по сути являются нефункциональными (URPS+). Эти две категории, в свою очередь, легли в основу другой классификации требований (предложенной Карлом Вигерсом, известным лектором и консультантом в области сбора требований и совершенствования процесса разработки ПО). Среди функциональных требований он выделяет три аспекта.

**1. Бизнес-требования**, формулируемые чаще всего спонсором или заказчиком проекта. Это цели, которые преследует создание системы, какие преимущества необходимо таким образом достичь и какие задачи решить. Таким образом, формируются общий образ и границы проекта, которые также могут быть закреплены в уставе проекта.

#### **Пример**

На уровне технического задания могут быть сформированы требования к поддержанию бизнес-процессов. Так, в CRM-системе можно выделить коммуникационные процессы (информирование и регистрация и обработка обращений клиентов), отчетные и управленические процессы.

**2. Пользовательские требования**, задачи, решаемые системой для различных категорий пользователей. В данном случае требования могут быть представлены в виде сценариев (*user journey*), алгоритмов (для учета всех возможных вариантов действий), таблиц «событие — отклик».

#### **Пример**

Могут также описываться ключевые роли, такие как: «Оператор в банке», «Инвестор», «Партнер», «Клиент — физическое лицо», «Клиент — юридическое лицо». В зависимости от выделяемых групп пользователей будут различаться и их возможности в системе.

**3. Функциональные требования**, определяемая функциональность системы, описываемая затем детально в техническом задании для реализации разработчиком.

Выделяемые Вигерсом нефункциональные требования являются дополнительными аспектами, которые необходимо принимать во внимание при разработке.

**Бизнес-правила:** требования регуляторов (например, экологические нормы для производственных компаний), промышленные стандарты, корпоративные стандарты и политики как налагаемые внешней средой ограничения.

**Атрибуты качества:** дополнительные требования к программному продукту, не относящиеся к задаваемой классом систем функциональности (к примеру, для систем документооборота это прием, хранение, пересылка, контроль исполнения документов), но являющиеся необходимыми для эффективности системы характеристиками (целостность, интероперабельность с другими системами, интегрируемость).

**Ограничения:** преимущественно технические ограничения, накладываемые условиями, в которых система должна быть реализована, например, диктуемые целевыми параметрами производительности протоколы.

Источниками информации для функциональных требований могут служить описания бизнес-объектов (подразделений, должностей, ролей, процессов, систем, носителей информации) с указанием их взаимосвязей, а вид моделей и требования к ним определяются целями моделирования. Нефункциональные требования создаются по результатам обследований предприятия (интервью, анкетирование, наблюдение за типовым рабочим днем сотрудника).

Для описания необходимых шагов корректного формирования требований обратимся к мировым практикам, описанным в сборнике SWEBOK. Данный документ объединяет знания в области разработки ПО и, будучи разработанным комитетом сообщества IEEE Computer Society, призван определить набор знаний и рекомендуемые практики по пяти основным стадиям ЖЦ, а также по управлению конфигурацией, качеством, ИТ-проектом, в том числе с описанием процесса разработки ПО и используемых при этом методов и инструментов.

**Этапы формирования требований по SWEBOK.** Согласно SWEBOK работа над требованиями предполагает три основные активности: их сбор (по результатам интервью с представителями заказчика и обследования предприятия), анализ (в том числе на полноту, непротиворечивость) и документирование (в виде сценариев использования, *user journeys*, спецификаций процессов).

Как правило, в сборе требований со стороны заказчика принимают участие ключевые специалисты, наиболее четко и ясно пред-

ставляющие бизнес-деятельность своего подразделения и грамотно формулирующие требования для дальнейшей корректной их интерпретации группой внедрения. Важно отметить, что практически в любой организации можно встретить пользователей-экспертов, которые не просто готовы пользоваться системой или даже принимать участие в ее улучшении, но хотят полностью подстроить ее под свои нужды. Поэтому важно соблюдать баланс и реалистично представлять целесообразность добавления той или иной функции.

### Пример

В организации АВС с 300 сотрудниками до внедрения единой системы учет всех проектов вели вручную. Каждый проектный менеджер отвечал за 10—15 видов активностей и либо заносил их в Excel в свободном формате, либо пользовался онлайн-сервисами, либо вообще просто все важные вещи записывал в свой ежедневник. После того как руководством было принято решение об интеграции всех данных в единую систему и подключении к работе с ними филиала предприятия, эта идея была встречена достаточно негативно. Помимо того что сотрудники чувствовали, что их хотят контролировать, мало кто представлял, что делать с теми данными, которых будет не хватать для корректной работы с системой, или с той информацией, которую никто кроме них не собирал (например, датами промежуточных проектных встреч). Более того, одно и то же поле (например, «перспективность проекта») не всегда заполнялось. К примеру, 60% пользователей заполняли его без единого формата ввода:

- менеджер *A*ставил цифры от 1 до 5 (5 — высшая оценка, самый перспективный);
- менеджер *B*ставил цифры от 1 до 5 (но 5 — самый низкий приоритет, 1 — высший приоритет);
- менеджер *C*выбирал из вариантов «Перспективен для рынка Европы», «Перспективен для рынка США и Китая», «Перспективен в России», «Под вопросом», «Не перспективен», вариант «Другое»;
- менеджер *D*же просто писал текстовые комментарии, когда, по его мнению, конечный продукт, разрабатываемый в ходе проекта, может выйти на рынок.

Однако из 50 сотрудников пяти основных подразделений двое специалистов в свое время разработали сложную систему ведения проектов, использующую макросы для объединения данных нескольких Excel-файлов, и, естественно, хотели, чтобы именно их принцип организации данных был взят за основу. Так, если они ввели собственные развернутые классификации зрелости и перспективности проектов с dropdown-меню и 15 вариантами ответа, именно ее они предлагали взять за основу при определении формата соответствующих полей. Их мнение, безусловно, важно — и если они начинают пользоваться системой (а не бойкотировать ее), « рядовые » пользователи их подразделений, несомненно, присоединятся. Но насколько такая система будет удобна для них, насколько уровни детализации и классификации, «навязанные» коллегами, соответствуют потребностям массовых пользователей, в особенности других подразделений и филиалов?

Следует помнить, что, несмотря на важность получения поддержки ключевых пользователей, в конечном итоге система внедряется именно в интересах всех пользователей: именно их регулярная работа в системе импортированные ими данные являются базой для достижения эффективности системы.

Если, как уже было сказано, предприятие-заказчик пришло к решению о самостоятельном сопровождении системы на этапе эксплуатации, именно эти ключевые сотрудники будут принимать систему, для чего необходимо отчетливо понимать, как те или иные действия пользователей отразятся на работе системы и каким образом происходит движение информации в ней. И говоря о движении информации, вновь вернемся к этапу анализа и постановки задачи, на котором проводится описание автоматизируемых бизнес-процессов и актуальных для них потоков данных.

### 3.2.4. Подготовка технического задания

Техническое задание является достаточно регламентированным документом. Рекомендации по его формированию приведены в российском ГОСТ 19.201—78 «Техническое задание, требования к содержанию и оформлению» и ГОСТ 34.602—89 «Техническое задание на создание автоматизированной системы» (ТЗ на АС, которое подробнее рассмотрено ниже).

---

**Техническое задание (проектное решение)** на выполнение работ является основным документом, определяющим порядок создания, настройки, доработки и внедрения модуля системы (системы в целом).

---

Техническое задание должно включать в себя следующие пункты.

1. Общие сведения о модуле (системе).
2. Назначение и цели настройки (создания, доработки) модуля/системы (вид автоматизируемой деятельности (управление, проектирование и т. п.) и перечень объектов автоматизации).
3. Характеристики объекта автоматизации (наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть выполнены в результате создания системы, критерии оценки достижения целей ее создания).

4. Требования:

- к модулю/системе:
  - структуре и функционированию,
  - численности и квалификации персонала заказчика для работы с системой,
  - надежности, безопасности и отказоустойчивости,
  - эксплуатации и техническому обслуживанию,

- безопасности информации,
- непрерывности бизнеса и сохранности информации при сбоях или авариях,
- функциональности:
  - перечень функций и задач, подлежащих автоматизации;
  - сроки реализации каждой функции (задачи);
  - форма представления выходной информации (и ее характеристики);
- к проекту: информационное, программное, техническое, организационное обеспечение, включая рекомендации по техническим характеристикам стенда предприятия и рабочим местам пользователей;
- к составу и содержанию работ:
- по созданию и настройке системы (включая порядок и критерии ее приемки в эксплуатацию),
- подготовке объекта автоматизации к вводу системы в опытно-промышленную и промышленную эксплуатацию;
- к проектной документации.

Техническое задание систематизирует все функциональные и нефункциональные требования, тем самым обеспечивая базу для дальнейших активностей по созданию и внедрению системы.

### 3.3. Проектирование

На этапе проектирования формируется описание проектного решения. В зависимости от выбранного подхода к проектированию необходимо подготовить различный набор документов и отчетов. При этом порядок их подготовки во многом зависит от выбранной технологии проектирования.

Каноническое проектирование основано на пошаговом выполнении определенного порядка действий, при этом большинство процессов выполняются вручную или с минимальной автоматизацией. При этом проектирование зачастую осуществляется непосредственными исполнителями проекта.

Проектирование с использованием CASE-средств предполагает широкое применение программных инструментов, способных облегчить работу. Фокус внимания смешается на анализ и непосредственное проектирование, а инструменты позволяют автоматизировать создание документов, отчетности и генерацию кода.

Типовое проектирование применяется, когда речь идет об имеющихся типовых решениях, которые требуется адаптировать под потребности конкретной компании. В этом случае уникальный продукт не создается; напротив, происходит адаптация типового решения под реальные нужды бизнеса. Несмотря на кажущуюся про-

стоту, типовое проектирование в сложных случаях немногим легче предыдущих методов.

Документы и отчеты обычно генерируются автоматически, при этом вендор зачастую обладает шаблонами необходимых документов и дорабатывает их в соответствии с конкретным проектным решением.

При описании проектного решения на этапе проектирования обычно формируются следующие документы в том или ином виде и формате:

- список модулей и функций системы, необходимых для поддержки определенных на этапе анализа автоматизируемых бизнес-процессов;
- список справочников систем (будущей и, если применимо, текущей) для дальнейшего переноса и обновления данных;
- сценарий работы системы по категориям пользователей для формирования необходимого набора диалогов и процедур проектируемой системы (включая реакции на все возможные и даже очень маловероятные действия со стороны пользователя);
- элементы интерфейса пользователей (в случае гибкого или разрабатываемого «с нуля» решения) для достижения удобства работы с системой;
- список отчетов и панелей мониторинга (включая их формы и обязательные для реализации элементы); эти отчеты в дальнейшем будут использоваться как для учетных целей, так и для целей мониторинга системы ее администратором (в части формирования и сбора статистики по нагрузке на ее отдельные модули, свободным ресурсам, активности пользователей и т. п.);
- перечень настроек функциональных компонентов системы в соответствии с выделенными на предыдущем этапе требованиями;
- решение о необходимости, возможности и пути интеграции с существующими и планируемыми к реализации системами на предприятии заказчика, чтобы своевременно предусмотреть технические средства для интеграции.

На этапе детального проектирования важно крайне четко определить все функциональные возможности системы и определить ее место в общей программной архитектуре предприятия.

### Пример

Если предполагается, что система взаимодействия с клиентами CRM будет получать данные из личных кабинетов клиентов на портале компании, необходимо предусмотреть подобную интеграцию. Если в компании уже внедрена и работает ERP-система, CRM должна получать и предоставлять ей данные.

С организационной точки зрения к этому моменту важна подготовка иерархической структуры работ, базового календарного плана и дорожной карты, которые позволили бы управлять проектом на протяжении всей его реализации.

### 3.3.1. Техническое проектирование

В том или ином виде фаза проектирования всегда содержит этап технического проектирования. Цель данного этапа — подготовка к этапу настройки модулей системы в целом. Данный этап предусматривает разработку документации, определяющей:

- организационную и функциональную структуры модуля системы;
- уровни автоматизации информационного процесса предприятия;
- структуры информационного и технического обеспечения и требований к их элементам;
- задачи обработки данных и их алгоритмизацию;
- порядок разработки системы ведения нормативно-справочной базы.

Результаты работ оформляются в виде технического проекта (проектного решения на автоматизацию модуля системы) и представляют собой задание на программирование. Техническое проектирование, создание, настройка, доработка и внедрение модуля системы осуществляются только на основании соответствующих утвержденных технических заданий (проектных решений).

---

**Технический проект** (проектное решение по автоматизации модуля) — документ, содержащий описание функциональных компонентов системы, необходимых для реализации бизнес-процессов, описанных на этапе подготовки проектного решения, а также необходимых доработок системы и (при необходимости) процедур интеграции с модулями (или внешними системами), внедренными ранее.

---

Перечень документов, создаваемых на стадии «Технический проект», может определяться документом ГОСТ 34.201—89. Основной задачей этой стадии является разработка архитектуры системы и технических решений по ее реализации. Перечень документов, служащих базой технического проекта, определяется условиями договора и технического задания, но чаще всего в него включаются следующие документы.

1. Пояснительная записка:

- общие положения, например, стадии и сроки, цели и задачи, используемые при разработке системы объекты ИС;
- описание процесса деятельности;

- основные технические решения, например, структура системы, основные функции, режимы функционирования, требования к персоналу;

- мероприятия по подготовке объекта автоматизации к вводу системы в действие, например, подготовка информации к импорту, создание рабочих мест.

2. Входные и выходные данные системы, например, импортируемые и вводимые вручную пользователями данные (названия и источники документов), формируемые системой документы.

3. Схема функциональной структуры:

- описание функций подсистем, например, описание функций по отдельным модулям (например, модуль расчета заработной платы, модуль учета труда);

- информационные связи между элементами и с внешней средой.

4. Описание автоматизируемых функций:

- исходные данные;

- цели АС и автоматизируемые функции;

- характеристика функциональной структуры, например, список подсистем, описание процесса реализации функций, требования (к надежности, защите информации от несанкционированного доступа, сохранности информации);

- типовые решения.

5. Постановка задач и алгоритмы решения:

- характеристики комплекса задач;

- используемая информация;

- результаты решения, например, информация для выдачи выходных сообщений и данные для использования исключительно внутри системы;

- алгоритм решения.

6. Программное обеспечение:

- структура ПО;

- функции частей ПО;

- методы и средства разработки ПО, например, инструментальные средства проектирования модели предметной области, структуры баз данных, разработки, применяемые методологии разработки, проектирования системы или интерфейсов;

- операционная система, например, совместимые или предпочтительные ОС для клиентской части ПО и серверов (баз данных);

- средства, расширяющие возможности ОС.

7. Информационное обеспечение:

- состав информационного обеспечения:

- **внутри машинная** информационная база — набор электронных таблиц и других объектов баз данных для обработки и хранения данных,

— **внемашинная** информационная база — информация, поступающая в бумажном виде (например, нормативно-справочная информация, годовые бюджеты);

- организация информационного обеспечения;

— принципы организации информационного обеспечения, например, БД должна представлять собой взаимосвязанные реляционные таблицы,

— обоснование выбора носителей данных, например, расчет числа запросов, которые необходимо обрабатывать, на основе статистики предыдущих периодов по числу пользователей, объему информации, частоте обращения к ней,

— принципы и методы контроля в маршрутах обработки данных, например, контроль форматов данных, контроль заполнения обязательных полей,

— описание решений, обеспечивающих информационную совместимость АС с другими системами;

- организация сбора и передачи информации;

— основные источники информации, например, информация из внешних систем, бумажные носители информации, передаваемые по электронной почте файлы,

— общие требования к организации сбора, передачи, контроля и корректировки информации;

- построение системы классификации и кодирования;

— классификации объектов, принятые для применения в АС, например, Общероссийский классификатор валют (ОК 014—94), Коды представления названий стран (ИСО 3166—93), внутриорганизационный справочник сотрудников,

— классификации объектов, принятые для применения в АС;

- организация внутримашинной информационной базы;

— принципы построения,

— структура, например, физическая структура БД системы (с полями, типами данных);

• организация внемашинной информационной базы (информации в бумажном виде от внешних организаций).

#### 8. Комплекс технических средств системы:

• структура комплекса технических средств, например, принципы построения кластеров;

• вычислительный комплекс, например, серверы БД, приложений, веб-сервер, персональные компьютеры;

- абонентские пункты;

- аппаратура передачи данных.

#### 9. Ведомость документов.

В результате подготовки технического проекта должен быть определен перечень компонентов модуля системы, необходимых доработок, процедур интеграции с существующей системой (при

ее наличии и необходимости), необходимый для реализации бизнес-модели предприятия То-Ве. Так же однозначно утверждается перечень работ по настройке и доработке модуля системы, а также перечень и формы отчетов и первичных документов, получаемых из модуля системы. Помимо отчетов должны быть определены процедуры для справочников системы (при ее наличии), которые необходимо использовать для импорта, и тех справочников, для которых необходимо разработать механизмы двустороннего обновления информации.

Важно отметить, что после детального определения и согласования всех вышеописанных параметров необходимо к моменту начала следующего этапа выбрать и подготовить также среду разработки и тестирования для проведения развертывания системы.

### 3.3.2. Рабочее проектирование и прототипирование

В целях снижения риска создания не соответствующей требованиям и техническому проекту системы перед разработкой возможно создать ее прототип (макет) для проверки предлагаемых архитектурных/функциональных/интерфейсных решений на практике с будущими пользователями.

---

**Прототип** — «черновая» реализация интерфейса и базовой функциональности системы для анализа принципов ее работы и тестирования совместно с будущими пользователями в целях дальнейшей доработки и совершенствования.

---

Обратная связь в рамках прототипирования, организованная на первых этапах жизненного цикла, очень важна для устранения замечаний по итогам проектирования и внесения доработок в интересах пользователей и заказчиков системы. В первую очередь создается макет, а уже потом проводятся тестирование, обсуждение и внесение корректировок в спецификации и прототип. Рассмотрим основные виды прототипов.

В качестве первой классификации можно выделить «горизонтальное» и «вертикальное» прототипирование. Первый случай («горизонтальное» прототипирование) предполагает моделирование исключительно пользовательского интерфейса и форм без фокусирования на логике обработки информации. Имитируются результаты действий или запросов при нажатии на элементы управления и осуществляются переходы между формами системы для формирования представления о реакции системы на те или иные действия пользователя (и какие не предусмотренные в текущей реализации действия потенциальный пользователь может совершить). Это позволяет выявить и устраниить противоречия между сформированными на этапе анализа требованиями и их реализацией в техническом

проекте. А в рамках «вертикального» прототипа, в отличие от предыдущего варианта, фокус переходит на саму структуру системы для проверки корректности и работоспособности сформированного архитектурного решения.

Второй (и более известной) классификацией является «быстрое» и «эволюционное» прототипирование. В первом случае (как и следует из названия) по технологии RAD, более подробно рассматриваемой в теме 6, посвященной методологиям разработки, создается макет определенных компонентов системы для более предметного и эффективного диалога между разработчиками кода (интерфейсов) и пользователями. Важно, что подобный прототип не призван в дальнейшем дорабатываться и становиться частью системы, соответственно, достигается значительная экономия времени и ресурсов, так как нет необходимости фокусироваться на технических деталях быстродействия или энергоэффективности системы.

Напротив, эволюционное прототипирование, по сути являясь итерационной разработкой, подразумевает создание в несколько итераций кода и тестирование полноценной рабочей системы с реализованными алгоритмами и целевым уровнем производительности.

### 3.3.3. Объектно-ориентированный подход к проектированию

Свою историю объектно-ориентированный подход начинает в далеких 1960-х гг. Именно в этот период термин «объект» был впервые использован применительно к разработке программных продуктов. О. Дж. Даль, Б. Мюрхог и К. Ньюгард (Норвежский вычислительный центр, Осло) разработали язык Simula 67. В основе языка лежал известный и популярный на тот момент язык Algol-60. Предполагалось, что новый язык будет успешно применяться для моделирования и реализации сложных информационных систем.

Однако создание языка Simula 67 так и не привело к расцвету объектно-ориентированного подхода. Отчасти это связано с высокой на тот момент популярностью структурного подхода, который эффективно решал задачи, возникающие при создании сложных крупномасштабных ИС.

По-настоящему широкое применение объектно-ориентированного подхода началось с 1990 г., когда был создан язык SmallTalk. Автором языка стал А. Кей из исследовательского центра фирмы Xerox. Язык был интересен тем, что использовал в своей работе исключительно объектно-ориентированные конструкции.

Таким образом, объектно-ориентированный подход достаточно «молод» по общенаучным меркам. Во многом это может трактоваться как дополнительное преимущество, которое позволяет абстрагироваться и дистанцироваться от отработанных десятилетиями схем

и методов разработки ИС, но в то же время взять у них все самое лучшее.

Как было сказано выше, понятие объекта является базовым для объектно-ориентированного анализа и проектирования. В качестве объекта в данном случае подразумевается абстракция реальной или воображаемой сущности, которая характеризуется индивидуальностью, четкими границами, состоянием и поведением.

Определение объекта нуждается в ряде пояснений. Так, в качестве абстракции в данном случае рассматривается мысленное (или информационное) представление сущности реального мира, причем сущность не обязательно должна быть физической. Очень часто речь идет об информационных сущностях. Иными словами, абстракция — это модель некоей сущности, отражающая окружающий мир.

Объектно-ориентированный анализ и проектирование предполагают работу с огромным количеством разнообразных сущностей. Самый простой случай — это реальная сущность, которая является частью физического мира. В качестве реальных сущностей рассматриваются автомобили, сотрудники предприятия, аппаратные ресурсы компаний и т. п. Воображаемую сущность представить несколько сложнее, потому что физически она не существует. Как правило, воображаемая сущность существует в виде правил (регламенты, документация и пр.), показателей (KPI) и иных информационных характеристик объекта.

Далее необходимо сказать несколько слов об индивидуальности. Индивидуальность позволяет отличить одну сущность от другой. Это набор атрибутов, которые способны однозначно определить сущность. К примеру, общая сущность «Сотрудник предприятия» содержит разнообразные атрибуты, например: Ф. И. О., номера телефонов, дату рождения, информацию о форме занятости, размер заработной платы, начальников и подчиненных сотрудника и т. п.

В качестве состояния рассматриваются конкретные значения атрибутов для объекта в отдельно взятый момент времени. Так, сущность «Сотрудник предприятия» может быть охарактеризована следующими значениями атрибутов: Иван Петрович Сидоров, дата рождения: 3 сентября 1989 г., начальник отдела маркетинга, заработка плата 48 800 руб. и т. п. Иными словами, данный сотрудник может рассматриваться как экземпляр класса «Сотрудники предприятия».

И, наконец, следует обратить внимание на поведение объектов. Под поведением подразумевается некий набор алгоритмов или действий, присущих объекту. Иногда их число невероятно велико и даже неисчислимо (к примеру, если речь идет о поведении человека). Иногда, напротив, поведение ограничивается несколькими простейшими действиями (простейший пример — выключатель люстры, имеющий два положения: «включить» и «выключить»).

**Главные принципы объектно-ориентированного анализа и проектирования.** Несколько параграфами ранее были рассмотрены общие принципы проектирования информационных систем. Объектно-ориентированный анализ и проектирование предполагают использование нескольких специфических принципов.

Первый из этих принципов — наследование. Согласно данному принципу, знание об общей категории может быть применено для более узкой категории. Иными словами, дочерний класс наследует все атрибуты родительского класса. Соответственно, экземпляры дочернего класса имеют все атрибуты как родительского, так и дочернего классов. К примеру, родительский класс «Сотрудники предприятия» характеризуется атрибутами «Ф. И. О.», «Индивидуальный пропуск», «Рабочий номер телефона» и «Форма занятости». Дочерний класс «Сотрудник отдела логистики» будет иметь все эти атрибуты, но вместе с ними — и свои собственные, к примеру, «Непосредственный начальник», «Непосредственные подчиненные», «Сфера ответственности» и т. п. А конкретный сотрудник, в данном случае — экземпляр класса, будет характеризоваться конкретными значениями перечисленных атрибутов.

Согласно принципу инкапсуляции внутреннее содержание объекта скрыто от других объектов и является независимым (или как можно более независимым). В соответствии с этим принципом информационный обмен между объектами минимизируется для их упрощения и увеличения степени автономности.

Третий принцип объектно-ориентированного анализа и проектирования — полиморфизм. Согласно принципу полиморфизма все модели объектно-ориентированного подхода должны строиться так, чтобы в зависимости от реальных обстоятельств они могли принимать различные формы.

**«Сильные стороны» объектно-ориентированного анализа и проектирования.** Во-первых, следует отметить, что с точки зрения общей логики описание системы, основанное на объектах, позволяет приблизиться к реальному состоянию предметной области. Опытная команда разработчиков, использующая объектно-ориентированный подход, сможет проанализировать предметную область организации и спроектировать информационную систему, соответствующую ей.

Во-вторых, большинство сущностей, с которыми приходится сталкиваться при разработке ИС, обычно обладают каким-либо поведением. Объектно-ориентированный подход позволяет эффективно описывать это поведение. Для сравнения, структурный подход предполагает раздельное описание поведения (в виде, к примеру, бизнес-процессов) и атрибутов (в виде создания инфологических моделей). Объектно-ориентированный подход, напротив, описывает их совместно.

В-третьих, принцип инкапсуляции позволяет легко адаптировать информационную систему к любым изменениям, а также повторно использовать объекты. Вместе с этим облегчается и дальнейшее сопровождение ИС.

В-четвертых, объектно-ориентированный анализ и проектирование позволяют наиболее эффективно использовать преимущества параллельных вычислений. Автономная работа каждого объекта, которая постулируется принципом инкапсуляции, позволяет добиться высокого уровня автономности в целом для информационной системы.

И, наконец, в-пятых, объектно-ориентированный подход в современном виде характеризуется наличием широкого спектра программных средств разработки, проектирования, анализа, моделирования и т. п. Отдельного внимания заслуживают коммерческие продукты IBM, которые будут рассмотрены в дальнейшем.

### **3.3.4. Основы Unified Modeling Language**

Унифицированный язык моделирования UML является методологией объектно-ориентированного подхода, представляющей набор диаграмм для проектирования информационных систем.

При изучении UML следует принимать во внимание тот факт, что унифицированный язык визуального моделирования не имеет обязательной привязки ни к конкретным инструментам (так, многие UML-диаграммы можно построить даже при помощи MS Visio), ни к конкретным методологиям создания ИС. Иными словами, применение UML не обязывает разработчиков к использованию заранее определенных программных средств или методологий. Однако UML часто применяется в паре с унифицированным процессом (Unified Process, UP).

На сегодняшний день UML является наиболее простым, понятным и повсеместно употребляемым объектно-ориентированным языком визуального моделирования, который был принят в качестве промышленного стандарта организацией OMG. Фактически сегодня UML объединяет лучшие визуальные приемы моделирования и разработки ИС/ПО.

**История UML.** Применение объектно-ориентированных методов долгое время было серьезно затруднено из-за слабой проработки данного сегмента. Такая ситуация сохранялась вплоть до середины 1990-х гг. В данный период среди лидеров объектно-ориентированных методологий выделялись метод Буча (Booch Method), а также техника объектного моделирования. Первый язык моделирования был предложен Гради Бучем, а второй разработан Джеймсом Рамбо. Суммарно обе методологии занимали практически половину рынка языков визуального моделирования.

Также на данном этапе возникла первая методология визуального моделирования, созданная Айваром Джекобсоном. Интересно, что в рамках данного периода неоднократно появлялись сообщения о создании полноценной методологии визуального моделирования, однако на деле оказывалось, что очередная новинка является не более чем очередным языком визуального моделирования.

Первые шаги в сторону унификации были предприняты Колеманом в 1994 г. Его метод был основан на использовании визуального языка Fusion, однако не получил широкого распространения сразу по нескольким причинам. Во-первых, исследование вышло в свет слишком поздно — к моменту его появления методологии и языки моделирования успели шагнуть вперед. Во-вторых, Г. Буч, А. Джекобсон и Дж. Рамбо, передовые эксперты в данной области, не приняли участие в работе Колемана.

В том же году Г. Буч и Дж. Рамбо начали работу над UML в совместно созданной Rational Corporation. Все участники рынка, которые так или иначе были заинтересованы в появлении языка визуального моделирования, были не на шутку встревожены: новая организация обладала фактически монополистическим положением за счет контроля более половины рыночного сегмента. Как оказалось в дальнейшем, беспокойства участников рынка оказались напрасными.

Уже спустя три года, в 1997 г., Группа объектного управления (Object Management Group, OMG) приняла UML в качестве промышленного стандарта. Фактически UML стал первым открытым стандартом визуального моделирования. Вместе с этим был положен конец другим методам визуального моделирования, поскольку все существующие языки оказались бессильны конкурировать с удобным и понятным UML.

В 2001 г. появился UML версии 1.4. Основным отличием стала добавленные семантика и язык действий, благодаря чему модели стали полными в вычислительном смысле. В результате появилась возможность сделать UML-модели исполняемыми. В 2003 г. была выпущена версия UML 1.5, которая содержала некоторые элементы ожидаемой версии UML 2.0.

2005 г. ознаменовался появлением UML 2.0. Новая версия унифицированного языка моделирования обладала новыми синтаксическими конструкциями, однако основные принципы остались прежними. UML стал по-настоящему сформировавшимся языком, эффективность которого была подтверждена на практике десятками компаний в сотнях проектов разного масштаба.

В 2007 г. появился UML 2.1.2, а два года спустя был создан UML 2.2.

UML версии 2.4.1 был также принят в качестве международного стандарта моделирования. Речь идет о стандарте «ISO/IEC

19505:2012. Информационные технологии. Унифицированный язык моделирования группы по управлению объектами (OMG UML)».

### 3.3.5. Общая характеристика UML

**«Унифицированность» UML.** Очевидно, что слово «унифицированный» содержится в названии языка не просто так, и на это есть достаточно много причин. На сегодняшний день UML успешно унифицирует различные области визуального моделирования и проектирования ИС и ПО. Прежде всего, унификация касается того факта, что UML позволяет выполнять моделирование на протяжении всего жизненного цикла программного обеспечения или информационной системы. Фактически сегодня UML применяется с самых первых этапов (формирование требований) вплоть до заключительных (реализация, развертывание). Таким образом, UML действительно унифицирован, поскольку подходит для применения независимо от конкретной фазы ЖЦ и не нуждается в дополнительных языках.

Второй аспект унификации UML связан с областью его применения. Последние версии UML могут использоваться для моделирования любых областей ИС и ПО, будь то аппаратная составляющая или сложные — и даже интеллектуальные — алгоритмы поддержки принятия решений. Благодаря этому практики объектно-ориентированного подхода получают возможность использования одного инструмента, применимого в разных областях деятельность по созданию ИС и ПО.

Третий аспект заключается в том, что UML не имеет привязки ни к одному конкретному языку программирования. Безусловно, очень часто UML используется в связке с объектно-ориентированными языками программирования, однако их применение является скорее желательным, чем обязательным. Модели, созданные при помощи UML, могут быть легко реализованы на C, C#, C++, Java, Visual Basic и других языках программирования.

Следует отметить отсутствие привязки к методологиям создания ИС/ПО, о котором говорилось ранее. Действительно, UML может одинаково эффективно применяться с различными методологиями, хотя зачастую UML используется в паре с UP или RUP.

**Статика и динамика в UML.** Возможность моделировать поведение объекта — одна из отличительных черт объектно-ориентированного подхода в целом. UML успешно справляется с этой задачей за счет того, что любая UML-модель условно включает в себя два аспекта: статический и динамический.

Статическая структура модели демонстрирует, какие типы объектов наиболее важны для моделирования системы и каким образом они взаимодействуют. Динамическая структура модели иллюстрирует поведение объектов, их жизненные циклы и взаимодействие

друг с другом, в результате чего ИС или ПО получают требуемую функциональность.

В UML существуют как статические, так и динамические модели. Как правило, различные типы моделей создаются на разных этапах жизненного цикла ИС или ПО. Так, на ранних этапах ЖЦ для формирования требований наибольшее значение имеют статические модели (диаграмма прецедентов и др.). Однако чем больше фаз пройдено, тем выше значение динамических моделей, поскольку они описывают реальное взаимодействие компонентов системы.

Унифицированный язык визуального моделирования может применяться в любом подходе к разработке информационной системы, будь то классические «жесткие» методы или гибкие подходы. Тем не менее, очень часто UML используется в паре с унифицированным процессом разработки программного обеспечения, созданным компанией Rational Software.

Вместе с UML используется как открытая версия унифицированного процесса (UP), так и коммерческая версия, получившая название RUP. На сегодняшний день окружение RUP состоит из множества прикладных программных решений, выпускаемых и поддерживаемых компанией IBM, поглотившей Rational Software в 2005 г.

### 3.3.6. Структура UML

Структура UML достаточно проста (рис. 3.1). Во многом именно поэтому язык стал таким популярным. Она включает в себя строительные блоки, общие механизмы и архитектуру (рис. 3.2).



Рис. 3.1. Структура UML

В качестве *строительных блоков* рассматриваются ключевые элементы, отношения и диаграммы UML. Условно говоря, строительные блоки — это камни, из которых складывается UML-модель.

*Общие механизмы* — это направления, следуя которым можно добиться поставленных целей. Общие механизмы — это UML-логика, которая позволяет решить возникающие задачи.

И, наконец, *архитектура* — это общее представление архитектуры информационной системы при помощи UML. Каждый элемент структуры UML необходимо подробно рассмотреть.

**Строительные блоки UML.** Строительные блоки — это фактическая основа UML, с которой регулярно сталкивается разработчик,

использующий унифицированный язык моделирования. Согласно общим руководствам принято выделять три ключевых строительных блока UML: *сущности, отношения и диаграммы*.

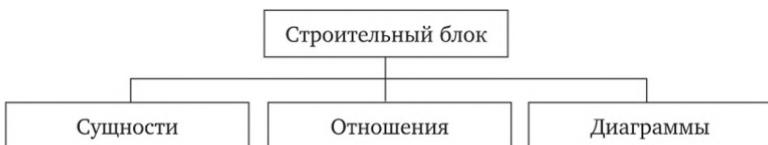


Рис. 3.2. Строительные блоки в UML

*Сущности* — это основные элементы UML-модели. Выделяют следующие типы сущностей:

- структурная сущность (класс, интерфейс, кооперация, прецедент, компонент и др.); выражаются с помощью существительных;
- поведенческая сущность (взаимодействия, деятельность, автоматы); выражаются с помощью глаголов;
- группирующая сущность (пакет, при помощи которого модули объединяются в единое целое);
- аннотационная сущность (примечание, внешне похожее на обыкновенный бумажный стикер); используется для удобства.

На практике применяются сразу все типы сущностей. Использование первых двух типов является обязательным и неизбежным одновременно, поскольку они используются для создания статических и динамических моделей. Иногда, если речь идет о разработке небольшого программного решения, можно избежать использования группирующих и аннотационных сущностей. Однако они просты и удобны в использовании и существенно облегчают работу.

*Отношения* в UML (табл. 3.1) используются для того, чтобы показать взаимодействие двух и более сущностей. Для успешного моделирования при помощи UML обязательно нужно понимать отличия между разными типами отношений, поскольку практически все они активно применяются на практике.

Таблица 3.1

#### Типы отношений в UML

Тип	Синтаксис	Описание
Зависимость		Исходный элемент зависит от целевого
Ассоциация		Описывает набор связей между объектами
Агрегация		Целевой элемент является частью исходного
Композиция		Строгая агрегация
Включение		Исходный элемент содержит целевой элемент

Тип	Синтаксис	Описание
Обобщение		Исходный элемент является специализацией целевого
Реализация		Исходный элемент гарантированно выполняет контракт целевого элемента

Именно благодаря правильному указанию типа связи удается понять, какую роль сущность играет в UML-модели, поэтому ни одним типом отношений не следует пренебрегать.

И, наконец, в качестве третьего типа строительных блоков фигурируют **диаграммы**. В качестве диаграммы в UML рассматривается представление модели, а не сама модель. Различие между диаграммой и моделью имеет принципиальное значение. Говоря метафорически, диаграмму можно трактовать как картину, написанную с модели. Таким образом, удаление сущности или отношения из диаграммы не приводит к их автоматическому удалению из модели. Для этого требуется совершить отдельное действие.

Всего существует более десятка диаграмм UML, основные из которых будут подробно рассмотрены в отдельном параграфе.

**Общие механизмы UML.** Общие механизмы UML последовательно применяются по всему языку моделирования. Всего выделяют четыре общих механизма:

- 1) спецификации (описание «заднего плана» модели);
- 2) дополнения (возможности дополненного описания любого символа UML);
- 3) принятые деления (описывают конкретные способы представления объектов реального мира в модели);
- 4) механизмы расширения (применяются в случае, когда базовые возможности UML не удовлетворяют выдвигаемым требованиям).

*Спецификации* — это текстовое описание модели, которое отражает ее суть. Любая модель может быть представлена как графически, так и в текстовом виде — в виде спецификаций. Спецификации также используются для удобства моделирования, поскольку ряд элементов модели может присутствовать в спецификации и отсутствовать на диаграмме (рис. 3.3).

UML не сводится к диаграммам. Напротив, диаграммы служат для визуального отражения модели ИС. Соответственно, одних только графических диаграмм недостаточно для описания модели. Любая диаграмма должна содержать спецификации, благодаря которым она и оказывается вплетенной в общую модель. Только диаграмма, обладающая исчерпывающей спецификацией, может рассматриваться как по-настоящему полезная.

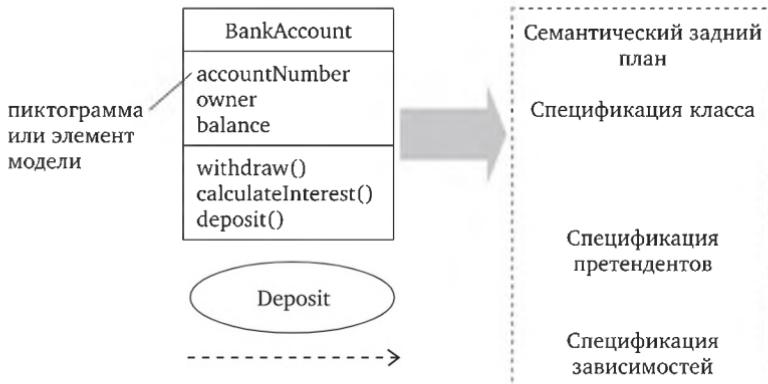


Рис. 3.3. Спецификации в UML

Дополнения используются в UML для того, чтобы представить какой-либо элемент диаграмм с необходимой полнотой. Визуализация как таковая не всегда достаточна, и в этом случае востребованными оказываются дополнения (рис. 3.4).

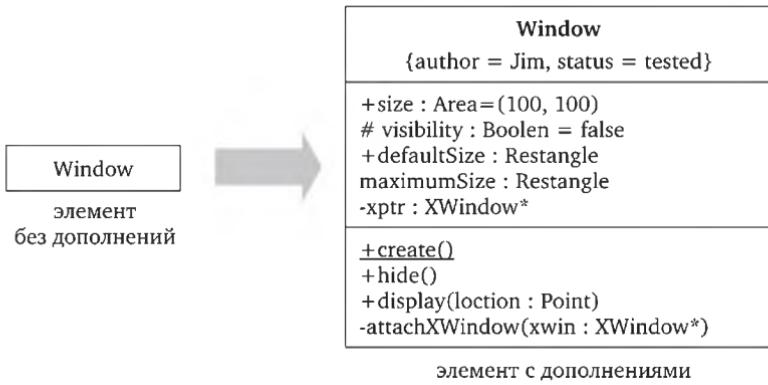


Рис. 3.4. Дополнения в UML

Дополнения нужны, чтобы отразить на диаграмме важные характеристики модели, которые не представляется возможным показать при помощи визуальных методов (например, из-за перегруженности диаграммы). Как правило, дополнения изображаются в форме прямоугольника, который содержит информацию об элементе модели.

Другой важный общий механизм UML — это принятые деления. В первую очередь следует рассмотреть деление «Классификатор — Экземпляр». В качестве классификатора понимается абстрактное

понятие (например, сотрудник предприятия). Соответственно, экземпляром является конкретный объект (например, «экземпляр» маркетолог Иванов Иванович).

Второе принятное деление — это «Интерфейс — Реализация». Это деление позволяет отделить, что выполняется, от того, как это выполняется. Разберем на примере самолета: приборная панель и штурвал будут рассматриваться в качестве интерфейса, тогда как устройство самолета будет реализацией. Как правило, конкретные виды реализаций обеспечивают существование того или иного интерфейса, но возможна и обратная ситуация.

И, наконец, следует сказать несколько слов о механизмах расширения. Выделяют три основных механизма:

1) ограничения (расширяют семантику элемента, позволяя тем самым добавлять новые правила);

2) стереотипы (возможность определять новые элементы модели на основании уже существующих);

3) помеченные значения (возможность добавлять новую специальную информацию в спецификации элемента).

**UML и архитектура ИС.** UML рассматривает архитектуру ИС в качестве организационной структуры системы. Архитектура — это взаимосвязанные части ИС, которые взаимодействуют друг с другом, а также механизмы и управляющие принципы, за счет которых обеспечивается функционирование ИС. Иное определение архитектуры ИС звучит как «высокоуровневое представление системы и ее окружения».

Для описания стратегических аспектов архитектуры информационной системы в UML чаще всего используется модель «4 + 1 представлений» архитектуры. Модель включает в себя логическое представление, процессное представление, представление реализации, представление развертывания и представление прецедентов. Модель является фундаментом для проектирования ИС при помощи UML, поэтому ее необходимо отдельно рассмотреть в следующем разделе.

### 3.3.7. «4 + 1 представления» архитектуры ИС

Объектно-ориентированный анализ и проектирование системы сосредоточиваются на создании архитектуры ИС. Правильно смоделированная архитектура ИС (или отдельно взятого приложения) является важным условием получения конечного продукта, который удовлетворял бы требованиям заинтересованных сторон.

В случае использования унифицированного языка визуального моделирования чаще всего применяется архитектура «4 + 1 представлений». Каждое представление, рассматриваемое в данной модели, описывает какой-либо аспект информационной системы, значимый для конечных пользователей, разработчиков, менеджеров проекта, специалистов по эксплуатации и т. п.

В случае применения объектно-ориентированного анализа и проектирования архитектура информационной системы наглядно демонстрирует, каким образом ИС разделена на компоненты и как осуществляется взаимодействие компонентов. Таким образом, архитектура ИС базируется на функциональных требованиях и создает ограничения для последующих проектных решений, поскольку они будут вписаны в общий архитектурный каркас.

Описание модели «4 + 1 представлений». Образно говоря, модель «4 + 1 представлений» архитектуры — это выжимка наиболее важных компонентов на высоком уровне абстракции (рис. 3.5). Компоненты разделены по четырем представлениям: логическому, процессному, реализации и развертывания. Пятое, стоящее отдельно, представление — это представление прецедентов.



Рис. 3.5. Модель «4 + 1» представлений архитектуры ИС

**Представление прецедентов** (Scenarios) описывает поведение системы в терминах прецедентов с точки зрения внешних (относительно ИС) акторов. Представление прецедентов отражает функциональные требования, которым должна удовлетворять ИС.

**Логическое представление** (Logical) описывает словарь предметной области. Для этого представление оперирует классами, подсистемами и интерфейсами ИС.

*Процессное представление* (Process) описывает, как процессы и потоки взаимодействуют во время работы ИС. Процессное представление отражает различные нефункциональные требования (например, параллелизм и т. п.).

*Представление реализации* (Development) охватывает систему на уровне артефактов (компонентов, файлов, модулей и т. п.), которые используются для сборки, выпуска и конфигурации программной части ИС.

*Представление развертывания* (Physical) отражает связь компонентов ПО с аппаратными средствами.

Модель «4 + 1 представлений» архитектуры никогда не создается за один подход. Она формируется итеративно в течение всего цикла создания информационной системы.

Следует отметить, что не все представления обязательно должны присутствовать в модели. В случае простых информационных систем допустимо отказаться от отдельных представлений. Кроме того, разработчики могут добавлять собственные представления, если это необходимо. Так, можно добавить представление защиты информации, представление данных, представление интерфейсов и т. п.

**UML и модель «4 + 1 представлений» архитектуры.** Диаграммы UML позволяют графически детализировать все представления данной модели. При этом следует понимать, что каждое из представлений архитектуры имеет свой синтаксис, а UML используется для детализации на более низком уровне.

Таблица 3.2

**Соответствие UML-диаграмм и представлений архитектуры модели «4 + 1»**

Представления	Диаграммы UML
Представление прецедентов	Диаграмма прецедентов
Логическое представление	Диаграмма классов Диаграмма состояний
Процессное представление	Диаграмма последовательности Диаграмма кооперации Диаграмма деятельности Диаграмма классов (применительно к процессам)
Представление реализации	Диаграмма компонентов
Представление развертывания	Диаграмма развертывания

Все перечисленные в табл. 3.2 диаграммы и порядок их создания будут рассмотрены далее в подпараграфе «Диаграммы в UML».

**Выбор подхода для визуального моделирования.** В некоторых современных средствах моделирования на UML работа начинается с выбора подхода, в рамках которого будет проходить дальнейшее моделирование. Выбор подхода определяет набор высокогенерализованных моделей, которые в дальнейшем будут детализироваться при помощи диаграмм UML более низкого уровня. Чаще всего визуальное моделирование основывается на следующих подходах:

- подход на основе «4 + 1» представлений архитектуры;
- подход Rational;
- подход на основе пяти моделей (модель прецедентов, модель анализа, модель разработки, модель реализации, модель развертывания);
- подход без заранее заданных высокогенерализованных моделей (применяется для небольших проектов).

Далее будет рассмотрен подход, основанный на «4 + 1» представлениях архитектуры ИС. Данный подход нередко рассматривается в качестве базового, поскольку он лежит в основе UML.

Для каждой модели используются свои элементы нотации, которые совпадают с элементами нотаций других диаграмм UML.

### 3.3.8. Диаграммы в UML

В нотации UML 1.0 используется девять диаграмм, которые чаще всего применяются в визуальном моделировании независимо от версии языка (в последующих версиях набор диаграмм был расширен). К каноническим типам диаграмм относятся:

- диаграмма прецедентов (*Use-Case Diagram*);
- диаграмма классов (*Class Diagram*);
- диаграмма состояний (*Statechart Diagram*);
- диаграмма деятельности (*Activity Diagram*);
- диаграмма последовательности (*Sequence Diagram*);
- диаграмма кооперации (*Collaboration Diagram*);
- диаграмма компонентов (*Component Diagram*);
- диаграмма развертывания (*Deployment Diagram*);
- диаграмма объектов (*Object Diagram*).

Часть диаграмм используется для моделирования структуры, другая часть — для моделирования поведения (рис. 3.6). При этом порядок создания диаграмм считается условным, хотя различные методологии рекомендуют использовать их в определенной последовательности или хотя бы фокусировать внимание на разработке определенных диаграмм в разных фазах ЖЦИС.

Назначение и состав всех представленных на рисунке диаграмм, как и порядок их создания, будут подробно рассмотрены далее.

В UML 2.0 число диаграмм увеличилось до 13. Некоторые диаграммы были переименованы. Так, диаграмма кооперации стала называться диаграммой коммуникации (*Communication Diagram*),

а диаграмма состояний стала диаграммой автоматов (*State Machine Diagram*).



Рис. 3.6. Типы основных диаграмм UML 1.0

Необходимо обратить внимание на следующие диаграммы:

- *диаграмма составной структуры* (Composite Structure Diagram), при помощи которой демонстрируется внутренняя структура классов и, по возможности, взаимодействие элементов внутренней структуры класса;
- *диаграмма пакетов* (Package Diagram) иллюстрирует пакеты и отношения между ними;
- *диаграмма обзора взаимодействия* (Interaction Overview Diagram) основана на совместном применении диаграммы последовательности и диаграммы кооперации;
- *диаграмма синхронизации* (Timing Diagram) является альтернативным представлением диаграммы последовательности, главной особенностью которой является ярко выраженная шкала времени.

В тексте учебника эти диаграммы не будут рассматриваться, хотя в редких случаях их создание может оказаться по-настоящему не-

обходимым условием успешного моделирования информационной системы.

*Синтаксис диаграмм.* Синтаксис диаграмм в UML достаточно прост. Любая диаграмма UML состоит из трех частей: рамки, заголовка и содержимого (рис. 3.7). Рамка служит для общего ограничения диаграммы. Область содержимого служит для размещения сущностей и отношений между ними. Заголовок изображается в виде неправильного пятиугольника и, как правило, содержит тип, имя и параметры диаграммы, хотя это и не является обязательным требованием.

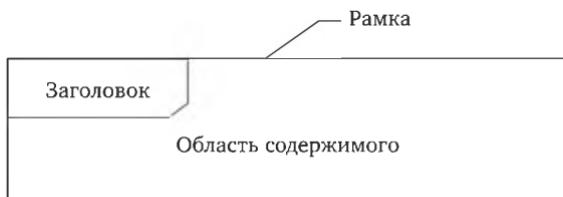


Рис. 3.7. Синтаксис диаграмм в UML

*Тип* диаграммы всегда соответствует типам, перечисленным на предыдущем рис. 3.6. Допустимым является сокращение используемого типа, однако официальные документы по UML не содержат общепринятого списка сокращений, в связи с чем разработчики имеют возможность выбрать вариант для сокращения самостоятельно. Тем не менее, обычно нет необходимости специально указывать тип диаграммы, поскольку все они легко различаются по внешним признакам.

*Имя* диаграммы в идеале должно отражать суть конкретной диаграммы. К примеру, если диаграмма иллюстрирует процесс регистрации нового клиента на сайте интернет-магазина, то ее можно назвать «*Customer\_Registration*». Разумеется, формулирование четких имен не является обязательным требованием со стороны языка моделирования, однако это серьезно упрощает работу, особенно когда речь идет о масштабных информационных системах.

И, наконец, параметры диаграммы содержат описание информации, которая нужна представленным на диаграмме элементам.

*Порядок создания* диаграмм. Единого и общеобязательного порядка построения UML-диаграмм не существует. Выбор той или иной последовательности действий чаще всего определяется обстоятельствами, связанными с созданием ИС. Тем не менее, достаточно часто используется следующий порядок создания диаграмм:

- 1) диаграмма прецедентов;
- 2) диаграмма классов;
- 3) диаграмма деятельности;

- 4) диаграмма кооперации;
- 5) диаграмма последовательности;
- 6) диаграмма состояний;
- 7) диаграмма компонентов;
- 8) диаграмма развертывания.

Допускается параллельное моделирование различных диаграмм. Готовые диаграммы нередко уточняются множество раз во время создания ИС.

**Диаграмма прецедентов.** Диаграмма прецедентов (она же диаграмма вариантов использования, англ. Use-Case Diagram) — первая диаграмма, которая моделируется средствами языка UML. Диаграмма прецедентов рассматривает корпоративные бизнес-процессы верхнего уровня с внешней точки зрения и позволяет понять, как выглядит деятельность компании «со стороны».

Диаграмма прецедентов достаточно проста. Все прецеденты в подобных диаграммах должны относиться к тому или иному действующему лицу (актору), так как определяют для них варианты/сценарии поведения.

*Представление прецедентов.* Несмотря на ее кажущуюся простоту, ни в коем случае нельзя с небрежностью относиться к построению диаграммы прецедентов. Диаграмма прецедентов формируется при детализации модели представления прецедентов. Диаграмма прецедентов включает все прецеденты, соответствующие функциональным требованиям к ИС, а не только архитектурно значимые.

*Рекомендации по разработке диаграммы прецедентов.* Существует ряд правил, способствующих созданию качественной диаграммы прецедентов.

Отсутствует потребность в доскональной детализации прецедентов. Диаграмма прецедентов показывает крупные функциональные блоки, а не описывает подробно всю функциональность системы.

Действия, которые совершаются внутри прецедента, должны составлять неделимую цепочку. Эта цепочка будет детализироваться в последующих диаграммах.

Прецедент не описывает, как именно будет выполняться что-то. Прецедент описывает, ЧТО ИМЕННО будет выполняться.

Элементы диаграммы нужно по возможности располагать без пересечений, а их расположение должно делать интуитивно понятным взаимодействие функциональных блоков.

Диаграмма прецедентов не должна учитывать конкретные варианты реализации ИС, связанные с программной или аппаратной стороной вопроса.

**Диаграмма классов.** Вслед за этим наступает время создания диаграммы классов. Диаграмма классов используется для описания структуры классов, атрибутов, методов и зависимостей между ними. При этом следует помнить, что класс в UML — это шаблон, по кото-

рому создается множество объектов, а не набор уже существующих объектов. Иными словами, класс первичен в том смысле, что объекты создаются на его основе, а не класс формируется на основе существующих объектов.

Как и в случае с любым объектно-ориентированным подходом, классы являются основными строительными блоками модели. Именно за счет диаграммы классов происходит формирование предметной области модели. Диаграмма классов представляет собой набор статических элементов модели.

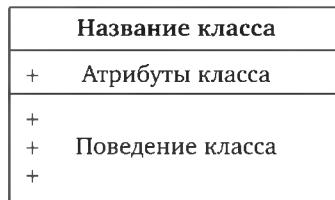


Рис. 3.8. Изображение класса в UML

Из рис. 3.8 видно, что класс изображается в UML в виде прямоугольника, состоящего из трех разделов. Верхний раздел содержит название класса, средний — атрибуты класса, а нижний — варианты поведения класса.

Классы могут быть отображены по-разному (рис. 3.9): необходимо отображать блок «Поведение класса», если таковой отсутствует. Существуют классы без атрибутов. Для упрощения в ряде случаев допускается отображать только название класса.

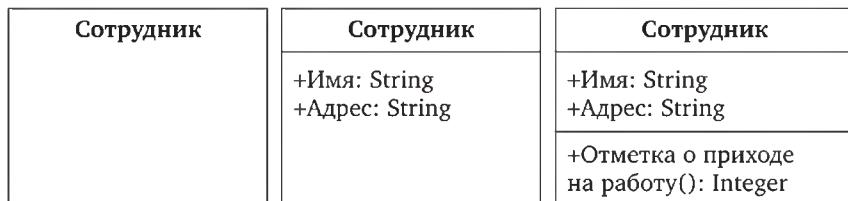


Рис. 3.9. Варианты отображения классов в диаграмме классов

Важно помнить, что класс может содержать и атрибуты, и поведение, даже если они не визуализируются в каком-либо конкретном случае.

*Атрибуты и варианты поведения.* Атрибуты используются, чтобы описать свойства для каждого объекта из данного класса. В каком-то смысле, атрибут — это именованное свойство класса, которое задает множество допустимых значений этого свойства. Формат описания атрибута таков:

[Visibility] [/]Name [:Type] [Multiplicity] [=DefaultValue],

где *Visibility* — видимость класса, которая отражает доступность атрибута одного класса для другого класса; *Name* — имя атрибута; *Type* — тип атрибута (например, String, Integer и т. п.); *Multiplicity* — множественность; *DefaultName* — текущее значение атрибута в конкретный момент времени.

В свою очередь, поведение класса — это услуга, которую класс может выполнить. Формат описания атрибутов выглядит следующим образом:

[*Visibility*] *MethodName* ([*ArgList*]) [:*ReturnType*],

где *MethodName* — имя поведения класса; *ReturnType* — тип возвращаемой переменной.

UML использует четыре варианта видимости для атрибутов и поведения классов (рис. 3.10):

- 1) + — открытый (*Public*), атрибут «виден» любому другому классу;
- 2) # — защищенный (*Protected*), атрибут «виден» только потомкам класса;
- 3) — закрытый (*Private*), атрибут «не виден» ни одному внешнему классу;
- 4) ~ — пакетный (*Package*), атрибут «виден» любому классу из данного пакета.

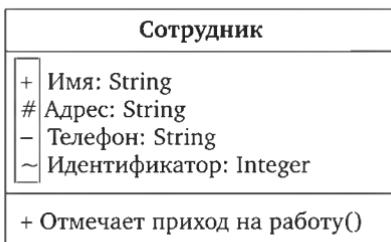


Рис. 3.10. Пример обозначения видимости на диаграмме классов

Данные варианты видимости используются и в других диаграммах UML. Их применение не ограничивается классами. Видимыми или невидимыми могут быть и другие элементы диаграмм.

Теперь, рассмотрев атрибуты и варианты поведения классов, можно перейти к типам связей в диаграмме классов.

Типы отношений элементов диаграммы классов приведены в табл. 3.3.

*Ассоциация* — один из самых часто применяемых типов отношений. Следует понимать, что разница между атрибутом и ассоциацией достаточно условна. С одной стороны, атрибут может рассматриваться как вырожденная ассоциация, на одном из концов которой

находится не выраженный в модели класс. В ряде случаев однополярные ассоциации могут быть безболезненно заменены соответствующими атрибутами.

Таблица 3.3

**Типы отношений в диаграмме класса**

Название	Обозначение	Описание
Ассоциация (Association)	_____	Объекты одного класса связаны с объектами другого класса
Агрегация (Aggregation)	_____ ◇	Отношение «часть — целое»
Композиция (Composition)	_____ ◆	Сложный вариант агрегации: если класс-контейнер уничтожен, то все более низкие классы тоже уничтожаются
Наследование (Generalization)	_____ ➤	Один класс (надтип) является формой обобщения другого класса (подтип)
Реализация (Realization)	----->	Частный случай наследования, при котором наследуется только поведение класса
Зависимость (Dependency)	----->>	Изменение спецификации одного класса ведет к изменению спецификации другого класса (односторонний характер)

С другой стороны, двусторонние направленные ассоциации очень сложно заменить атрибутами так, чтобы не потерять наглядность созданной диаграммы.



Рис. 3.11. Пример отображения множественности связей на диаграмме классов

И, наконец, следует рассмотреть кардинальность (множественность связи). Кардинальность любого типа отношений отражается в форме  $n..m$ , где  $n$  и  $m$  — целые числа, значение которых больше или равно нулю (рис. 3.11).

**Диаграмма деятельности.** Диаграммы деятельности (*Activity Diagram*) создаются для детализации прецедентов, выделенных в диаграмме прецедентов. Диаграммы деятельности служат для иллюстрации последовательности действий, которая выполняется для

реализации каждого отдельно взятого прецедента. Виды деятельности и отдельные действия соединяются между собой потоками, движение которых происходит от выходов одного узла к входам другого.

Диаграммы деятельности применяются и для моделирования деятельности предприятия (вариант описания бизнес-процессов), и для описания вычислительной работы информационной системы.

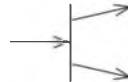
Основные фигуры диаграммы деятельности таковы:

- *действие (Action)*, которое означает какой-либо единичный вид деятельности и иллюстрируется при помощи прямоугольника со скругленными углами;
- *начало (Initial)*, обозначающее начальный момент деятельности. Изображается в виде черного круга с заливкой;
- *конец (Final)*, обозначающий завершение деятельности. Изображается в виде черного круга с обводкой;
- *поток (Control Flow)*, показывающий последовательность перехода. Изображается в виде односторонней линии.

В диаграммах деятельности используются четыре логических оператора, приведенные в табл. 3.4.

Таблица 3.4

**Логические операторы в диаграмме деятельности**

Название	Обозначение
Ветвление (Fork)	
Соединение (Join)	
Слияние (Merge)	
Принятие решение (Decision)	

Изучив приведенные выше элементы, а также логические операторы диаграммы, можно заметить общее сходство с логикой построения бизнес-процессов в структурном подходе. Действительно, по своей семантике они достаточно близки.

**Диаграммы взаимодействия.** Диаграммы взаимодействия тесно связаны также с диаграммой классов. Все отношения, которые отражены на диаграммах взаимодействия, являются формой ассоциации между классами.

Следует принимать во внимание, что, хотя диаграммы взаимодействия не являются прямым отражением диаграмм деятельности, они не должны противоречить друг другу.

Всего существует четыре типа диаграммы взаимодействия, но чаще всего применяются только два из них: *диаграмма последовательности* и *диаграмма кооперации*.

**Диаграмма последовательности.** Диаграмма последовательности акцентирует внимание на порядке передачи информации во времени. Диаграмма показывает сообщения в том же порядке, в каком они будут передаваться в ИС.

Сообщения в диаграмме последовательностей обозначаются линиями. Всего бывает три типа сообщений.

1. **Асинхронное.** В этом случае передатчик не будет ждать реакции от получателя и продолжит свою деятельность сразу после передачи сообщения.

2. **Синхронное.** Передатчик посыпает сообщение получателю и ждет ответ.

3. **Возврат,** означающий, что получатель вернул значение или управление передатчику.

Диаграмма последовательности содержит два типа объектов:

1) **линия жизни (Lifeline).** Вертикальная линия, которая отображает существование объекта в течение какого-то временного периода;

2) **фокус управления (Focus of Control).** Период времени, в течение которого активный объект выполняет действия.

**Диаграмма кооперации (Collaboration Diagram)** является другой формой иллюстрации взаимодействия. Фокус внимания смещается не на время и не на порядок передачи сообщений, а на сами отношения между объектами. Последовательность сообщений на диаграмме кооперации задается порядковыми номерами.

**Диаграмма состояний.** Диаграммы состояний формируются на основе диаграммы классов. Диаграмма состояний описывает логику перехода конечного автомата из одного состояния в другое под воздействием каких-либо событий (воздействий, сигналов и т. п.). Состояние объекта — это момент, в который объект удовлетворяет какому-то условию, выполняет определенное действие или ожидает какого-то события.

В качестве конечного автомата рассматривается последовательность состояний, через которые проходит объект в ответ на различные события, а также ответные действия на эти события.

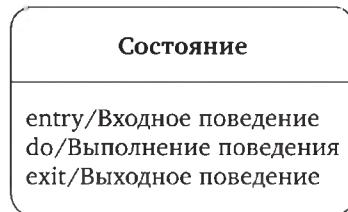
Особенность диаграммы состояний заключается в том, что каждая диаграмма описывает состояния только одного экземпляра класса. При этом экземпляр класса должен быть «реактивным», т. е. реагировать на внешние события.

*Простое состояние* — это основа диаграммы состояний (рис. 3.12). Состояние считается простым, если оно не имеет внутренних регионов и подсостояний. Простое состояние описывается при помощи трех внутренних секций:

1) *entry* — это метка входного поведения; она показывает, какое поведение выполняется при входе в данное состояние и не зависит от перехода, в результате которого произошел вход в состояние;

2) *exit* — это метка выходного поведения; она показывает, какое поведение выполняется при выходе из данного состояния и не зависит от перехода;

3) *do* — это метка, которая описывает поведение, выполняющееся вплоть до выхода из состояния.



*Рис. 3.12. Простое состояние (Simple State)*

Далее требуется рассмотреть *композитные состояния* (Composite State). Композитные состояния включают в себя *вложенные состояния* или *регионы*. Регионы — это элементы модели, которые содержат состояния и переходы, а также являются частью композитного состояния конечного автомата.

**Диаграмма компонентов.** Построение диаграммы компонентов — следующий этап объектно-ориентированного анализа и проектирования. Для построения диаграммы компонентов необходимо иметь построенные диаграммы прецедентов, диаграммы классов и диаграммы последовательности (и (или) диаграммы кооперации).

Диаграмма компонентов (*Component Diagram*) показывает структурные компоненты ИС и связи между ними. В качестве компонентов рассматриваются только информационные объекты: файлы, модули, библиотеки, пакеты и т. п.

Компонент — это физически существующая часть системы, благодаря которой обеспечивается реализация классов и отношений, а также достигается функциональность моделируемой ИС.

Для компонентов UML определяет следующие стереотипы:

- *file* (файл) — самая распространенная разновидность компонента, принимающая вид какого-либо файла;

- *executable* (исполнимый) — разновидность файла, являющегося исполнимым; может исполняться на какой-либо компьютерной платформе;

- *document* (документ) — разновидность файла в формате документа, который не является исполнимым и не содержит исходный код программы;
- *library* (библиотека) — разновидность файла в формате библиотеки (динамической или статической);
- *source* (источник) — разновидность файла, который содержит в себе исходный код программы;
- *table* (таблица) — разновидность компонента в формате таблицы БД.

Компонент изображается в виде большого прямоугольника, слева на котором расположены два маленьких вытянутых вдоль прямоугольника. Графическое изображение компонента происходит от изображения модуля системы.

Интерфейс — следующий элемент диаграммы компонентов. Интерфейс отображается в виде окружности (необязательно замкнутой), которая связана с компонентом определенным отношением. Рекомендуется начинать название интерфейса с буквы 'I' (например, IDialog). Интерфейс может либо реализовываться, либо использоваться компонентом.

Соответственно, диаграмма компонентов содержит следующие типы отношений:

- отношение зависимости;
- отношение реализации.

Наименование интерфейса напрямую зависит от его отношения к компоненту. Если компонент X реализует интерфейс Y, то интерфейс Y называется экспортруемым (поддерживаемым), поскольку данный интерфейс будет предоставляться другим компонентам в виде сервиса. Если компонент X использует интерфейс Y, реализуемый другими компонентами, то интерфейс Y называется импортируемым. Импортируемые интерфейсы отражаются при помощи отношения зависимости, а экспортруемые — при помощи отношения реализации (рис. 3.13).

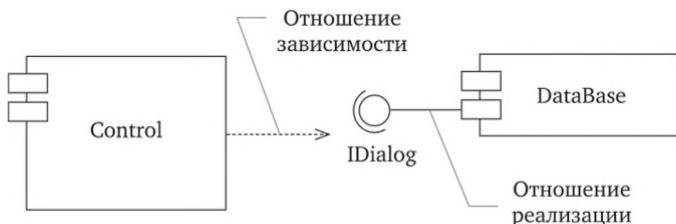


Рис. 3.13. Отношение зависимости и отношение реализации на диаграмме компонентов

Если компонент использует какие-либо классы, то на диаграмме изображается отношение зависимости между компонентом и клас-

сами. Рис. 3.14 показывает, что компонент использует классы, которые реализуются в другом компоненте.

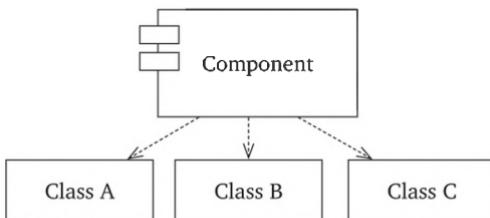


Рис. 3.14. Отношение зависимости между компонентом и классами

Диаграммы UML не позволяют устанавливать отношение реализации между компонентом и классами, поэтому графическое изображение такой ситуации выглядит иначе. Рис. 3.15 означает, что компонент реализует классы.

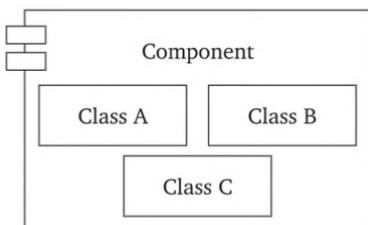


Рис. 3.15. Реализация классов компонентом

**Диаграмма развертывания.** После диаграммы компонентов разработчики переходят к созданию диаграммы развертывания (*Deployment Diagram*). С помощью диаграммы развертывания моделируются узлы (аппаратные средства) и артефакты (программные средства), которые будут развернуты на них. Следует понимать, что диаграмма развертывания не является подробным описанием аппаратной части ИС. Данный тип диаграмм предназначен для моделирования оборудования, связанного с ИС, на среднем и (или) высоком уровне абстракции.

Узел (*Node*) — первый из двух основных элементов диаграммы развертывания. Узел является аппаратным элементом ИС и представляет собой либо техническое устройство, либо вычислительный ресурс. Узлом может быть автоматизированное рабочее место, сеть, процессор, видеокамера наблюдения, сканер штрих-кодов и т. п.

Узел обозначается при помощи трехмерного куба (рис. 3.16, 3.17). В диаграмме компонентов можно отразить узлы как на уровне типа (например, ПК), так и на уровне экземпляра (ПК с инвентарным номером 154424).

В первом случае на узле указывается имя типа узла. Во втором — сначала указывается имя экземпляра, затем после двоеточия указывается тип узла, и все название подчеркивается. На узлах можно также размещать дополнительную информацию в формате {Текст примечания}. Примечания размещаются под названием внутри поверхности куба.

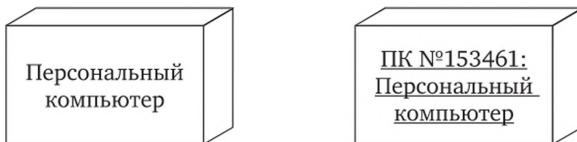


Рис. 3.16. Узлы на диаграмме развертывания

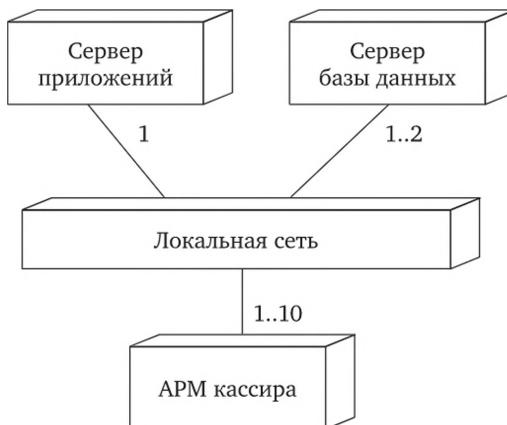


Рис. 3.17. Пример узлов на диаграмме развертывания

Стоит отметить, что диаграмма развертывания и диаграмма компонентов часто используются совместно. К примеру, нередко отражается отношение между узлами и компонентами, которые на нем развернуты (рис. 3.18).

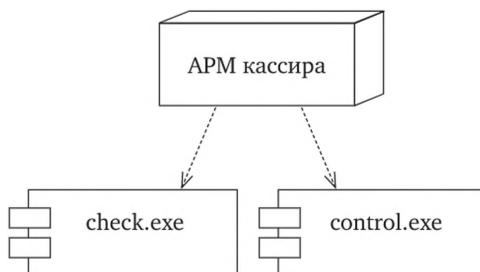


Рис. 3.18. Узлы и компоненты на диаграмме развертывания

Второй элемент диаграммы развертывания — это артефакт (*Artifact*). Артефакты в диаграмме развертывания близки по смыслу к компонентам. Однако в ряде случаев для большей наглядности требуется показать их именно в виде компонентов.

## 3.4. Разработка

Ко времени старта реализации системы, как уже было сказано, должна быть выбрана и подготовлена среда разработки, а также выбрана методология, в соответствии с которой будет осуществляться управление разработкой. От нее зависит очень многое: принцип взаимодействия команды разработки и внедрения (между собой и с основными стейкхолдерами проекта), следование одной из моделей жизненного цикла ИС (каскадной, спиральной и пр.), длительность самого процесса разработки и тестирования, а также прочие важные аспекты процесса реализации системы. Подробнее различные методологии как управления проектами, так и непосредственно разработки, будут рассмотрены в следующей теме.

### 3.4.1. Закупка ПО

Каждая организация выбирает собственные, оптимальные именно с учетом ее специфики, критерии выбора между рядом платформ и решений. Наиболее распространенными критериями являются:

- функциональность системы;
- надежность и стабильность работы решения (устойчивость к различным режимам работы и степени активности пользователей);
- наличие и качество встроенных средств администрирования и управления;
- стоимость (внедрения, лицензий, поддержки);
- удобство организации услуг по поддержке и сопровождению системы;
- наличие отраслевого решения;
- наличие успешного опыта реализации проектов на базе данной платформы (надежность системы);
- опыт и компетенции организаций-подрядчиков;
- учет корпоративной специфики и соответствие корпоративным стандартам.

Список критериев очень индивидуален. Он зависит от специфики проекта и, как правило, формируется из разработанных на этапе проектирования требований к системе.

Следующим шагом будет организация непосредственно процедуры закупки (например, путем открытого запроса предложений), выбор поставщика и заключение договора на поставку ПО.

Типовая структура подобного договора приведена ниже.

1. **Предмет договора.** Например, поставка, установка ПО, монтаж оборудования, обучение персонала заказчика.
2. **Стоимость договора и порядок оплаты.** Сумма оплаты, сроки и размер платежей, форма оплаты (наличный/безналичный платеж).
3. **Права и обязанности сторон.** Порядок поставки, установки и ввода в эксплуатацию ПО, предоставления информации со стороны поставщика. Порядок назначения ответственных лиц, предоставления информации со стороны покупателя.
4. **Ответственность сторон.** Условия выплаты неустойки / расторжения контракта в случае невыполнения или ненадлежащего выполнения обязательств (в том числе несвоевременной поставки или оплаты). Порядок разрешения споров, в том числе в случае действия обстоятельств непреодолимой силы.
5. **Гарантийное сопровождение и авторский надзор.**
6. **Условия конфиденциальности.**

### 3.4.2. Настройка конфигураций

К моменту инсталляции системы на стенде предприятия заказчиком должна быть обеспечена устойчивость работы ЛВС стенда, проведена закупка программно-технического обеспечения в виде основного внедряемого программного продукта (самой системы) и ключей электронной защиты, а также завершены все работы предшествующих фаз жизненного цикла. Программно-техническое обеспечение для установки и тестирования системы, в том числе клиентские лицензии, предоставляет заказчик (из числа закупленных ранее либо через осуществление их закупки на этом этапе).

В процессе инсталляции модули системы устанавливаются на стендовый сервер с проведением основных необходимых настроек под подлежащие автоматизации бизнес-процессы предприятия заказчика. В зависимости от типа и особенностей системы в ней могут быть реализованы разные механизмы дополнения и изменения объектов в системе.

**Создание кода программы.** Единого языка программирования, использующегося для всех приложений, просто не бывает. Сейчас для многих стандартных пользовательских приложений используется комбинация SQL и C# для серверной программной части и HTML/CSS/JavaScript для создания интерфейса пользователей.

#### Пример

Современные ERP-системы наиболее часто используют собственные языки программирования, в связи с чем поддерживающие их специалисты должны обладать знаниями не только по работе с формами и пользовательским интерфейсом, но и в первую очередь по структуре системы

и быть способными безопасно внести необходимые изменения в код. Вот некоторые сочетания ERP-систем и используемых в этих системах внутренних языков программирования:

- SAP — ABAP;
  - Oracle — PL/SQL, Java;
  - Microsoft Dynamics AX — среда разработки MorphX с языком программирования X+;
  - Microsoft Dynamics NAV — графическая среда разработки C/SIDE с языком программирования C/AL.
- 

В современных коммерческих продуктах практически всегда присутствуют средства автоматизированной настройки системы («конфигуратор», «конструктор»). В некоторых случаях код программы может являться самодостаточным способом настройки системы. Разница между ними состоит в возможностях внесения изменений (бизнес-логика и правила работы приложения, как правило, требуют большей и более сложной работы, нежели просто добавление нового поля). Для более простых функций обычно используются стандартные конфигураторы системы, не требующие знания языка программирования.

Таким образом, может быть выстроена следующая иерархия возрастания сложности и функциональных возможностей внесения изменений: *Интерфейс пользователя* → *Конфигуратор* → *Код программы*.

Разумеется, крайне важно, чтобы конфигурация была тщательным образом документирована, особенно в части «надстроек» над стандартными функциональными возможностями, описываемыми в том числе в поставляемой вендором документации. Доработку кода необходимо проводить так, чтобы исключить наличие так называемого закрытого кода, недоступного для изменения. В противном случае, если в ходе разработки была создана определенная схема расчета или логика процесса, при необходимости внесения изменений ИТ-специалистам придется с нуля создавать требуемый код/функциональность. Именно поэтому специалисты должны быть обучены не только технике конфигурирования продукта, но и особенностям конкретной конфигурации.

**Настройка параметров системы в режиме конфигуратора.** Примером системы, работающей в подобном режиме конструктора, является Salesforce.com. Эта система позволяет разработчику определять множество параметров, таких как: формат данных создаваемых полей, их взаимосвязи, формат отображения, источники данных (ручной ввод, вычисление по формуле или получение информации из других источников).

В том или ином виде конфигураторы присутствуют практически во всех системах, но объем их возможностей (и требования к ква-

лификации работающих с ними специалистов) значительно различаются.

**Настройка полей и создание библиотек системы.** Наиболее простой из типов конфигурирования. По сути, таким образом предоставляется возможность расширить информацию об объектах системы и расширить возможности их взаимодействия. Дополнительные прописываемые поля, типы данных и прочие элементы позволяют кастомизировать каждый объект, а также описать возможные для него действия.

### Пример

Возможность дополнить карточку сотрудника в системе набором новых полей с выбором типа каждого поля и установкой для них ограничений. Так, если в стандартной конфигурации существуют только три поля: Идентификатор сотрудника, Фамилия и Имя, то можно добавить любые дополнительные поля, например, Дата рождения, Должность, Дата принятия в штат. Для системы в дальнейшем не будет различий, были ли созданы поля автоматически или вручную, и они все в полной мере могут быть использованы в функциях поиска, фильтрации, создания отчетов.

**Настройка логики бизнес-процессов.** Вторым видом наиболее часто используемых объектов конфигурации являются сами процессы. Например, в системе могут прописываться пути движения документов, согласно которым после утверждения документа первым согласующим он автоматически направляется второму согласующему с отправкой уведомления о статусе «владельцу» документа в системе.

### Пример

Приведем пример настройки логики бизнес-процесса выплаты бонусов сотрудникам филиала компании на основе результатов продаж. Бонусы планируются к выплате на периодической основе.

На примере конфигурации в системе SAP в таком случае объектами настройки могут быть:

- создание вида начислений к счету;
- определение схемы перерасчета, схемы учета результатов для заказа;
- определение вида заказа для отражения бонуса;
- ведение профиля расчета, присвоение профиля расчета виду заказа.

**Интерфейс пользователя, расположение элементов на вкладках.** Конфигураторы большинства систем позволяют изменять расположение объектов, скрывать некоторые из них, создавать персонализированные фильтры и настройки для групп пользователей. Данный шаг подробнее рассматривается ниже.

По результатам конфигурирования системы подрядчик готовит описание оптимальных настроек сервера БД и рекомендации по развертыванию рабочих мест, которое и проводится на следующем шаге. Это также позволяет выявить и устранить ошибки и неточности в программном продукте, затем подготовив описание *тестового сценария* (контрольного примера действий пользователя) для проверки корректности и актуальности осуществленных настроек.

После инсталляции и настройки модулей системы справочники и данные должны быть развернуты, формы ввода данных адаптированы, подготовлен акт сдачи-приемки работ.

### 3.4.3. Создание ролей пользователей

Роли пользователей (и определяемые ими права доступа) являются именно тем инструментом, который определяет:

- какой именно функциональностью системы будет пользоваться тот или иной сотрудник (группа сотрудников);
- какие данные будут ему доступны;
- какие он будет иметь права на чтение, редактирование информации, ее экспорт (!) и удаление?

Отдельно отметим экспорт информации. Эта функциональность, предоставляемая системой, тесно взаимосвязана с информационной безопасностью. Поэтому возможность выгрузки конфиденциальных данных должна быть предоставлена только ограниченному кругу лиц.

С технической точки зрения подобное разграничение доступа позволяет одному пользователю иметь несколько ролей по отношению к разным объектам системы.

Роли пользователей зависят от профиля компании и специфики системы, однако в целом они различаются по правам доступа к каждой из категорий данных. В списках доступа чаще всего задаются следующие возможности (по возрастанию объема прав).

- **R (read, чтение).** Возможность видеть элементы страницы и просматривать прикрепленные файлы.
- **W (write, запись).** Возможность изменять и редактировать значения полей страницы, добавлять файлы (возможность чтения, разумеется, сохраняется).
- **D (delete, удаление).** Возможность удалять страницу вместе с прикрепленными файлами (доступ на чтение/запись при этом сохраняется).
- **A (admin, администрирование).** Возможность добавлять/удалять дополнительные поля, менять формат представления, предоставлять и отзывать доступ к элементам (при сохранении возможности чтения, записи и удаления).

Получается логическая связка следующего вида: Учетная запись пользователя — Роль пользователя — Права доступа (комбинация прав и элементов, к которым осуществляется доступ).

### Пример

Как показано на рис. 3.9, несколько сотрудников (например, работающих на равных позициях одного подразделения) могут обладать одной ролью в системе (например, аккаунт-менеджера). В таком случае все права доступа будут прописываться не для каждого отдельного пользователя, а для ролей (что гораздо эффективнее с точки зрения трудозатрат, в особенности при внесении изменений).

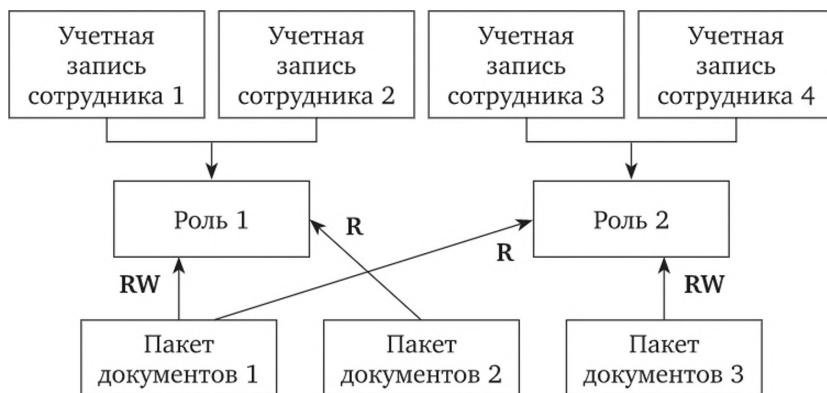


Рис. 3.9. Учетная запись сотрудника, роль и пакеты документов

Далее на рисунке показаны пакеты документов, к которым разрешен доступ, однако тот же самый принцип действует и для полей (групп полей) и страниц в системе. Так, в системе CRM аккаунт-менеджеру будет доступна возможность просмотра, редактирования и удаления мобильных телефонов его клиентов, но может быть отказано в доступе на внесение изменений в финансовые детали осуществления платежей. И напротив, сотрудники бухгалтерии с ролью «специалист по расчетам» смогут редактировать любые финансовые данные, но не будут видеть мобильные телефоны и историю сообщений руководства компании-клиента, которые доступны аккаунт-менеджеру.

#### 3.4.4. Миграция данных

Данные на протяжении своего жизненного цикла претерпевают огромное число действий над ними: создание, обработка/редактирование, резервирование, копирование, перемещение в новую виртуальную среду, новую систему. Миграция данных является очень важной составляющей процесса внедрения системы — по сути, на этом этапе создается механизм, позволяющий автоматически переносить основные справочники и другие данные в модули системы.

Важнейшим шагом в первую очередь является непосредственно перенос (первичная загрузка) и настройка **основных справочников**. По сути, это все те данные, которые являются едиными для различных модулей и систем предприятия и используются ими совместно и одновременно: справочники поставщиков, клиентов, складов, материалов, продукции (номенклатуры) и др. Именно благодаря наличию подобных основных данных не происходит дублирования записей, а в при внедрении систем не приходится заниматься ручным поиском, сбором и вводом этой информации. В целях тестирования основные справочники новой системы заполняются тестовыми данными, достаточными для тестирования модуля и принципов поступления и передачи из него информации, и затем переносятся в систему. Если по результатам выверки тестовых данных не наблюдается ошибок, то можно говорить об успешной миграции справочников и на первый план выходит уже другой вопрос, а именно: поддержание всех этих данных в актуальном состоянии.

Другой категорией данных для переноса является информация из функционирующих на предприятии программных решений и *legacy*-систем, на смену которым внедряется новый программный продукт. При корректной подготовке на стадии проектирования передача данных в новую систему организовывается автоматизированно, через специальные коннекторы и шины данных. В противном случае миграция данных становится «ночным кошмаром», когда они должны быть в ручном режиме выгружены из существующих систем, приведены к единому формату и после определения соответствия полей и типов данных загружены в новую систему. Но ведь данные в существующих системах могут обновляться ежесекундно. Насколько допустим этот простой на время миграции и как избежать всех проблем, ассоциированных с распределенным онлайн-вводом, не говоря уже о проблеме ресурсоемкости и неимоверной трудоемкости данного процесса, связанного с огромным риском потери информации?

В таких случаях миграция организуется в онлайн-режиме. Такой способ предполагает перенос данных при непрерывной работе всех серверов и приложений. Это особенно важно для критических для бизнеса приложений с высокими требованиями к доступности: Oracle, Exchange, биллинговых систем, систем электронной коммерции. Новые данные внедряются в уже существующий поток информации, объединяются с имеющимися в системе данными на логическом уровне через отдельный адрес диска, который после успешной миграции удаляется или архивируется.

К наиболее распространенным рискам миграции данных и причинам их возникновения можно отнести:

- **недостаточные компетенции/опыт** команды проекта в миграции данных (в том числе в части формирования документации об уже осуществленных проектах);

- **отсутствие единого ответственного за данные**, что может привести к невозможности определить наиболее корректные и актуальные данные;
- **низкое качество переносимых данных** (дублирование, некорректные форматы данных);
- **нестабильность целевой системы для переноса данных** — если разработка еще не завершена и миграция данных выполнена преждевременно, могут возникнуть ошибки или потеря информации;
- **несвоевременность организации синхронизации данных** — актуализация данных должна производиться в новой системе сразу после миграции;
- **низкая степень документированности процесса миграции** — данный риск относится не только к невозможности использования полученного опыта для выполнения дальнейших проектов, но и в большей степени связан с невозможностью в случае необходимости продемонстрировать аудиторам или компетентным контролирующими органам отчеты по датам и масштабам перемещения данных. А значит, потенциально может возникнуть огромное количество проблем, связанных с доказательством достоверности предоставленной отчетности. Особенно это актуально, если данные для отчетности выгружались в период параллельной работы старой и новой систем, при наличии различий в объемах данных, хранившихся в них в конкретный момент.

Таким образом, миграция данных — длительная, кропотливая, требующая большого внимания работа, связанная с высоким риском потери данных. В девяти из десяти случаев основной сложностью, с которой сталкиваются команды проекта при миграции, является дублирование данных. Ниже перечислены основные причины появления подобных «данных-двойников»:

- различия в написании объектов;

#### Пример

- Организационно-правовые формы и формы собственности («АО «Вымпел», ЗАО «Вымпел», ВЫМПЕЛ»). Это приводит к путанице: не ясно, идет речь о разных предприятиях или об одной и той же компании?
- Использование латиницы и сокращений: Ivanov Ivan, Иванов И.

- несовпадение информации из-за устаревания данных;

#### Пример

В одной системе в качестве адреса ООО «Вымпел» указывается г. Воронеж, а в другой — г. Москва. Разница может быть обусловлена переездом компаний либо же разной трактовкой понятия «адрес» сотрудниками, вводившими информацию в систему: в первом случае — юридический адрес компании, во втором — фактический.

- несовпадение форматов данных.

### Пример

- Неиспользование выпадающих списков для полей с ограниченным числом значений: г. Балашиха в одной системе и «Московская область» в другой.
- Использование текстовых полей для написания цифр (например, в случае миграции Excel-таблиц): 84951234567 добав 1111.
- Стиль написания цифр: + 7 (495) 123-45-67, 84951234567.

Во всех этих случаях необходима тщательная выверка всех данных, проводящаяся частично в автоматизированном режиме (например, для замены «+ 7» в телефонных номерах на «8», удаления лишних пробелов и других символов), частично в ручном режиме пользователями.

### 3.4.5. Разработка сценария тестирования

Сценарий тестирования включает описание начальных условий тестирования, входных данных, действий пользователя и ожидаемого результата. При помощи сценариев тестирования проводится как тестирование работоспособности функций ИС/ПО, так и тестирование работы при различной нагрузке. Иными словами, сценарий тестирования может объединять в себе элементы функционального и нефункционального тестирований.

Как правило, обычно создается не один сценарий тестирования, а набор таких сценариев. После этого они группируются по какому-то признаку, будь то последовательность реализации тестовых сценариев, применение в различных видах тестирования и т. п.

В идеальном случае результаты реализации сценариев тестирования должны полностью совпадать с ожидаемыми результатами. В противном случае все расхождения должны быть объяснимыми. В ряде случаев может потребоваться устранение расхождений.

Среди других характеристик сценария тестирования:

- проверка наиболее «слабых» мест системы для повышения вероятности обнаружения ошибок;
- выполнение необходимого минимума действий для проверки (неизбыточность);
- явное выявление ошибок и недоработок;
- понятность и логичность описания примера для пользователей.

Сценарии тестирования также могут относиться к конкретной проверяемой области: конфигурации системы, удобству использования, безопасности, взаимодействию или интеграции компонентов. Однако может создаваться и общий пример, с единым описанием и дальнейшей детализацией, в зависимости от специфики проверяемой области и задач тестирования.

Таким образом, ко времени старта тестовой эксплуатации должны быть подготовлены следующие элементы модели тестирования:

- **набор исходных данных**, условий тестирования, планируемых результатов;
- **методика испытаний**: технический документ, описывающий настройку системы для проведения теста и принцип оценки итоговых результатов;
- **сценарии тестирования**;
- **тестовый скрипт** (опционально: в случае использования автоматизированных инструментов тестирования).

Формальным основанием для старта тестовой эксплуатации могут служить:

- протокол совещания группы внедрения о передаче модуля системы в тестовую эксплуатацию;
- регламент реализации сценариев тестирования;
- перечень замечаний ключевых пользователей, собранных при предварительной реализации сценариев тестирования, с описанием решений по их устранению, отраженный в протоколе совещания группы внедрения.

#### 3.4.6. Тестовая эксплуатация

Между первоначальной настройкой системы на стенде по сформированным и утвержденным требованиям и ее полномасштабным вводом в эксплуатацию лежит один из самых сложных с точки зрения согласования интересов всех сторон этап тестовой эксплуатации. Как правило, прием системы в эксплуатацию и обучение пользователей проводятся исключительно после подтверждения согласия заказчика с «итоговой версией». Разумеется, для проведения опытной эксплуатации составляется отдельный регламент (методика). Все замечания, получаемые в ходе работы с системой выбранной группы ключевых пользователей, заносятся в отдельный журнал запросов на изменение с расстановкой приоритетов исполнения.

Основными блоками/классами тестов являются компонентное, модульное (*unit testing*), интеграционное и системное тестирование.

1. **Компонентное/модульное тестирование** (*component/unit testing*). Его основная цель состоит в идентификации ошибок реализации модулей в ходе проверки работоспособности (например, при различных поступающих на вход данных). Проверяются отдельные объекты, функции, классы, модули.

2. **Интеграционное тестирование.** Данный тип тестирования проводится для всей системы в целом. Его основная задача — проверка корректности передачи информации и взаимодействия между всеми модулями (компонентами) системы и слоями архитектуры.

ры (в частности, прикладным ПО, инфраструктурными сервисами, ОС). Особенno данный тип тестирования актуален для клиент-серверных и распределенных систем.

3. **Системное тестирование** (в том числе функциональное тестирование). Определение степени соответствия созданной ИС исходным требованиям (функциональным и нефункциональным). Проводится проверка корректности настройки внешних интерфейсов системы. Оцениваются различные характеристики надежности, производительности, отказоустойчивости системы, в дальнейшем также проверяемые в ходе нагрузочного тестирования.

В качестве дополнительных проверок могут создаваться тестовые окружения для ИС при эмуляции работы отдельных компонентов систем, внешнего окружения и действий пользователей. В частности, эмуляция позволяет смоделировать работу еще не созданных компонентов системы (модуля, сервиса) для более раннего проведения тестирования.

Тестирование в зависимости от специфики системы и предметной области может проводиться в ручном, автоматизированном или полуавтоматизированном режиме.

1. **В ручном режиме** специалисты по тестированию самостоятельно проходят весь сценарий тестирования. Этот способ применяется для крайне динамичной разработки в условиях постоянных изменений. По результатам составляется протокол тестирования, в который заносятся все замечания, комментарии и предложения.

2. **Автоматизированное тестирование** значительно упрощает и ускоряет процесс тестирования, предоставляет возможность провести большее число проверок, в том числе до и после внесения доработок в тестовую среду.

3. **Полуавтоматизированное тестирование** выполняется по заданному сценарию под постоянным контролем тестировщика. Таким образом, оно сочетает преимущества автоматизированного тестирования с возможностью в случае необходимости изменить сценарий его выполнения и организовать дополнительные проверки.

Тестирование как наиболее важный элемент обеспечения и контроля качества может и должно быть организовано по специальной методике в соответствии со стандартами. Среди наиболее часто применявшихся стандартов:

- ГОСТ Р ИСО/МЭК 12119—2000. Пакеты программ. Требования к качеству и тестирование;
- ГОСТ Р ИСО/МЭК 12207—2010. Процессы жизненного цикла программных средств;
- ГОСТ Р ИСО/МЭК 14764—2002. Сопровождение программных средств;
- ГОСТ Р ИСО/МЭК 15288—2005. Системная инженерия. Процессы жизненного цикла систем;

- ГОСТ Р ИСО/МЭК 15504—2009. Оценка процессов;
- ГОСТ 28195—89. Оценка качества программных средств. Общие положения;
- ГОСТ Р ИСО/МЭК 9126—93. Оценка программной продукции. Характеристики качества и руководства по их применению;
- **ГОСТ серии 19.** Единая система программной документации (ЕСПД);
- **ГОСТ серии 34.** Разработка автоматизированной системы управления;
- **IEEE 829.** Стандарт документирования тестирования программного обеспечения и систем.

Отдельно можно отметить унифицированный процесс RUP, который по своей сути исходит из идеи постоянного регулярного тестирования на каждой итерации.

Срок тестовой эксплуатации, как правило, не превышает одного месяца и не меняется при отсутствии или несвоевременном представлении замечаний. Среди выполняемых для организации тестовой эксплуатации мероприятий (все они проводятся по каждому функциональному модулю в отдельности):

- подготовка перечня замечаний ключевых пользователей, собранных в процессе окончательного тестирования контрольного примера (по каждому модулю системы), с описанием решений по их устранению, отраженного в Протоколе совещания Группы внедрения;
- предварительное тестирование командой внедрения контрольного примера (по отдельным модулям!) с занесением замечаний в журнал тестирования и последующим их устранением (в случае необходимости с дополнительными настройками и доработками);
- финальное тестирование с привлечением ключевых пользователей (здесь помимо функциональности крайне важно также проверить степень удобства и интуитивности интерфейса системы для пользователей);
- разработка общего руководства по работе с системой и инструкций пользователей по модулям системы (включая назначение и условия работы с модулем, описание операций по подготовке и самой работе);
- обучение пользователей базовым принципам работы с системой на контрольном примере (в отдельности для каждого модуля системы). В процессе обучения пользователь знакомится с интерфейсом системы, получает информацию по основным принципам работы и основным возможностям системы. В случае наличия разработанных методических руководств пользователю дополнительно предоставляется необходимая документация.

### 3.4.7. Доработка по результатам тестирования

Основная цель данного этапа — устранить замечания, возникшие в результате тестирования на контрольном примере, и при необходимости осуществить дополнительные настройки и доработки модулей системы. Как уже было сказано, на данном этапе важно разделять требования, критичные для успешной работы с системой, и дополнительные (*nice-to-have*) пожелания пользователей-экспертов, которые можно будет включить в следующие итерации работы и версии системы.

Что касается порядка организации данных типов проверки, то для первых сборок проводится поверхностное *smoke-тестирование*, направленное на критическую функциональность. Этот короткий цикл тестов должен подтвердить, что после сборки кода (нового или исправленного) приложение будет успешно выполнять свои основные функции. Код проверяется на предмет наличия быстронаходимых критических дефектов, которые необходимо срочно исправить, отправив на доработку. В случае их отсутствия и успешного прохождения тестирования приложение передается для проведения полного цикла тестирования.

Так как при *smoke-тестировании* в максимально короткие сроки покрывается максимально широкий объем функциональных возможностей, то существует и другой вид тестов, дополняющих первый (функции проверяются скорее «вглубь», с максимально тщательным тестированием каждого аспекта). Это обусловлено тем, что при проведении любого рода изменений необходимо обеспечивать их корректность и отсутствие негативного влияния на работоспособность и эффективность приложения.

С этой целью проводится *регрессионное тестирование* (*regression testing*), один из наиболее известных видов которого получил название *sanity-тестирования*. Это тестирование определенных участков кода, проводимое по результатам внесенных изменений. Его основная задача — удостовериться в том, что внесенные изменения не затронули работоспособность системы и не послужили причиной новых ошибок. К примеру, проверяется, действительно ли исправленные ранее ошибки исправлены, не приводят ли сделанные изменения кнейтрализации исправлений старых ошибок (регистрация багов, англ. *bug regression*) и есть ли новые ошибки.

В результате должны быть:

- проведены тестирование, выявление и устранение замечаний по контрольным примерам для каждого модуля системы (замечания обработаны, и либо сделаны соответствующие изменения в модуле, либо они отклонены с указанием причин);
- проведены дополнительные настройки и доработки модуля системы (в срок до четырех недель с момента передачи модуля си-

стемы в тестовую эксплуатацию и получения перечня окончательных доработок и настроек модуля системы);

- составлен протокол выполненных работ по устранению документированных замечаний ключевых пользователей заказчика по результатам тестирования контрольного примера по каждому модулю системы;
- ключевые пользователи обучены базовым основам работы с модулем системы на данных контрольного примера и готовы к его практической эксплуатации на своих рабочих местах, а также развертыванию на основном рабочем сервере и проведению опытно-промышленной эксплуатации (ОПЭ);
- сформирован протокол выполненных работ по обучению ключевых пользователей заказчика базовым основам работы с модулем системы на данных контрольного примера;
- подготовлен перечень окончательных доработок и настроек каждого модуля системы.

### **3.4.8. Прием результатов тестирования**

Результаты тестовых испытаний заносятся в единый отчет для рассмотрения заказчиком. При этом организуется приемочное тестирование (*acceptance testing*), представляющее собой комплексную проверку компонентов системы на предмет их работы с плановыми параметрами производительности и выполнением всех функциональных требований, в том числе на различных конфигурациях.

Прием результатов испытаний проводится при выполнении следующих условий:

- специалисты группы внедрения со стороны заказчика владеют функциональностью модуля системы и готовы к его развертыванию на рабочем сервере и опытно-промышленной эксплуатации;
- все замечания, возникшие в ходе первоначального тестирования контрольных примеров, задокументированы и устраниены;
- ключевые пользователи готовы к работе с модулем на своих рабочих местах;
- все замечания ключевых пользователей, собранные во время окончательного тестирования контрольного примера, обработаны, и либо сделаны соответствующие изменения в модуле, либо замечания отклонены с указанием причин отклонения;
- заказчиком утверждены итоговые документы.

## **3.5. Развёртывание и внедрение**

Этап развертывания системы является одним из наиболее важных с точки зрения распределения ответственности за работы между заказчиком и подрядчиком. Исполнитель выполняет основную

задачу — инсталляцию модуля на рабочий сервер с перенесением на него итоговой конфигурации с тестового сервера. Однако при первом последовательном обучении и вовлечении специалистов заказчика в процесс создания и внедрения системы остальные операции могут быть осуществлены ими:

- предшествующее работам обеспечение охвата всех автоматизируемых рабочих мест пользователей функционирующей ЛВС;
- настройка рабочего сервера;
- определение перечня и количества рабочих мест, которые необходимо развернуть для ОПЭ;
- развертывание рабочих мест пользователей с определением прав доступа;
- подготовка наиболее актуальной информации по каждому модулю для ввода в систему (включая подготовку, дополнение и выверку справочников, которые могли потерять актуальность за время между тестовой эксплуатацией и развертыванием системы);
- организация поддержания данных системы в актуальном состоянии (в том числе при интеграции с другими системами).

Целью данного этапа является подготовка модуля системы к опытно-промышленной эксплуатации — подготовка конечных пользователей, ввод начальных данных, подготовка приказов по предприятию заказчика о передаче модуля системы в опытно-промышленную эксплуатацию.

На этом этапе нужно:

- обеспечить безусловное выполнение условий готовности модулей системы к сдаче в опытно-промышленную эксплуатацию (закрепленных в отдельном документе);
  - ввести и выверить начальные данные (например, начальные остатки) по каждому модулю для ввода информации в систему;
  - отработать на рабочих местах пользователей модуля системы практические действия пользователей в параллельном режиме с имеющимися приложениями;
  - разработать, согласовать с подрядчиком и утвердить регламент взаимодействия подразделений заказчика, участвующих в опытно-промышленной эксплуатации модуля системы;
  - осуществить интеграцию модуля с другими модулями или внешними системами, внедренными ранее;
  - подготовить и издать приказ по предприятию заказчика о передаче модуля системы в опытно-промышленную эксплуатацию, который должен содержать:
    - наименование модуля системы, проходящего опытно-промышленную эксплуатацию;
    - наименование компании-исполнителя;
    - сроки проведения опытно-промышленной эксплуатации;

— список должностных лиц со стороны заказчика и исполнителя, ответственных за проведение опытно-промышленной эксплуатации модуля;

— перечень подразделений предприятия заказчика, участвующих в проведении опытно-промышленной эксплуатации;

— порядок и сроки перевода персонала заказчика на работу в условиях функционирования модуля системы.

К моменту ОПЭ уже должны быть проведены следующие работы:

- модули системы успешно перенесены и функционируют на рабочем сервере и автоматизированных рабочих местах пользователей (с разделением прав доступа);

- подготовлены, дополнены и введены недостающие справочники по каждому модулю системы, проведена выверка введенных в систему недостающих справочников;

- заказчиком утверждены итоговые документы.

### 3.5.1. Закупка и настройка требуемой ИТ-инфраструктуры

Важно учитывать, что для всех предприятий чрезвычайно высока ценность хранимой и обрабатываемой информации, а также стоимость времени простоя бизнеса. Поэтому к техническому обеспечению ИТ-систем компании должны предъявляться жесткие требования:

- по производительности — обеспечение приемлемого времени реакции с точки зрения пользователя, использующего данную систему;

- надежности и доступности — пользователи должны продолжать работу при выходе из строя единичных устройств, система не должна простоять и не должно быть потерь данных;

- катастрофоустойчивости — простой системы должен быть минимальным, а все данные должны быть сохранены; допускается некоторое ухудшение производительности и увеличение времени реакции системы;

- безопасности — система работает с данными, составляющими коммерческую тайну, и должна обеспечивать должный уровень защиты, вплоть до аттестации и сертификации решений и оборудования;

- масштабируемости — система должна иметь возможность приспосабливаться к увеличивающейся нагрузке со сравнительно малыми архитектурными изменениями или совсем без них.

Все эти требования к характеристикам решения в процессе подготовки ИТ-инфраструктуры детализируются на гораздо более подробном уровне. Ниже мы рассмотрим пример описания конкретных действий по обеспечению масштабируемости решения.

## Пример

---

Основной объем обработки проводится при помощи рабочих процессов сервера приложений (диалог, фон, обновление, очередь или подкачка данных). Эти процессы запускаются на инсталляциях сервера приложений (первичная и дополнительная инсталляции дополнительного сервера приложений, за исключением процессов постановки в очередь, которые запускаются на инсталляции центральных сервисов). Вся обработка, связанная с базой данных, контролируется инсталляцией базы данных. Это значит, что критически важно обеспечить высокую масштабируемость инсталляций сервера приложений и базы данных. Масштабируемость этих инсталляций обеспечивается следующим образом.

- Сервер приложений масштабируется по горизонтали путем добавления дополнительных серверных хост-узлов, на которых выполняются дополнительные инсталляции приложений. Отправка запросов на инсталляции сервера приложений выполняется при помощи сервиса отправки сообщений, работающего на инсталляции центральных сервисов.
  - Инсталляция базы данных масштабируется по вертикали путем адаптации необходимых аппаратных ресурсов (увеличения мощности процессора, расширение памяти и т. д.).
- 

К ИТ-инфраструктуре в таком случае можно отнести следующие компоненты.

Вычислительная инфраструктура:

- базовые инфраструктурные сервисы. Описание и процесс организации таких инфраструктурных сервисов, как IP-телефония, электронная почта, резервное копирование, антивирусная защита и антиспам, служба каталогов AD, служба удаленных рабочих столов, сетевая печать и пр. Данные приложения автоматизируют вспомогательные задачи конечных пользователей, поэтому к этому классу в первую очередь относятся приложения, связанные с пользовательскими коммуникациями;
- серверное и пользовательское оборудование, системы хранения данных (СХД). Определение перечня пользовательского и серверного оборудования и систем хранения данных;
- серверные площадки и центры обработки данных (ЦОД). Планирование типовых центров обработки данных и серверных комнат, вплоть до составления некой усредненной модели серверной комнаты/ЦОД;
- системное ПО. К этой категории относятся описания операционных систем и программных платформ как для серверов, так и для оборудования, доступного конечному пользователю. Помимо этого, в данном разделе рассматриваются службы виртуализации.

Сетевая инфраструктура:

- инфраструктура локальных вычислительных сетей (ЛВС). Данный компонент должен представлять собой типовое решение по организации внутренней локальной вычислительной сети. Имен-

но этот элемент является ключевым для обеспечения связи между оборудованием в офисах, филиалах, ЦОДах и других типах помещений. Важно, что в случае производственной или любой другой компании со значительной долей технологических систем проводится физическое и логическое разделение локальной сети на два сегмента, каждый из которых эксплуатирует только собственное сетевое оборудование. Таким образом, в корпоративной сети работают все бизнес-пользователи, а в технологической располагаются основные сервисы (оборудование, пользователи), поддерживающие технологическую составляющую и процессы компаний;

- корпоративные сети передачи данных (КСПД);
- телефония, ВКС, подключение к сети Интернет.

Инженерная инфраструктура: устройства бесперебойного питания, электропитания, охлаждения, кабельная инфраструктура.

Можно упомянуть, что среди современных тенденций построения ИТ-инфраструктуры — использование нескольких уровней абстракции для вычислительных комплексов, сетей передачи данных с целью повышения гибкости и обеспечения регулирования потребления ресурсов (что снижает в конечном итоге суммарную стоимость оборудования).

Действия на этапе построения необходимой для системы ИТ-инфраструктуры складываются из следующего.

1. Закупка оборудования — данный шаг обычно значительно увеличивает время подготовки технической платформы, так как закупка может быть связана с проведением конкурсов, заключением договоров, ожиданием поставки оборудования. Все эти шаги могут приводить к потерям времени, от нескольких недель до нескольких месяцев.

2. Подготовка инфраструктуры — расчет возможности установки оборудования в серверное помещение, выделение и конфигурирование сетевых портов, планирование мощностей компонентов инженерной инфраструктуры серверных и пр.

3. Аппаратное конфигурирование — получение оборудования, его конфигурирование и установка (в том числе тестовый сервер, рабочие станции, вспомогательное оборудование).

4. Установка системного программного обеспечения — система управления базами данных, операционная система, генераторы отчетов (либо установка базового программного обеспечения, реализующего слой виртуализации).

5. Конфигурирование пользовательских рабочих мест — рекомендуется определить несколько стандартных конфигураций пользовательских рабочих мест, которые позволяли бы обеспечивать приемлемое качество работы пользователей. Это дает возможность не только упростить поддержку клиентских рабочих мест, но и, например, настроить систему мониторинга для проверок ключевых

параметров работы приложений на всех конфигурациях пользовательских рабочих мест, используемых в компании.

В результате команда внедрения со стороны подрядчика получает развернутую функционирующую стендовую локальную вычислительную сеть, удовлетворяющую требованиям ТЗ, и программно-аппаратную платформу, уже готовую к последующей инсталляции системы.

### **3.5.2. Ввод начальных остатков**

Данный этап, актуальный для систем финансового управления, является логическим продолжением проведенной ранее миграции данных (со вводом справочников и первичной информации). Формирование начальных остатков проводится уже после настройки параметров приложения. На их основе составляется начальный баланс, с помощью которого проверяется, насколько корректно были введены данные. В зависимости от системы при вводе и выверке данных может проводиться либо редактирование сальдо счета (субсчета), либо формирование фиктивных проводок. Делаются записи по дополнительным регистрам. При этом проводится тщательный контроль реакции программы на все операции, проверки сбалансированности сальдо, проверки каждого счета (в том числе аналитических).

### **3.5.3. Обучение пользователей**

Для минимизации сопротивления, которое может встречаться со стороны специалистов организации при внедрении «очередной» системы, обучение проводится в три основных этапа.

1. Выделение и обучение ключевых пользователей.
2. Обучение рядовых пользователей.
3. Повышение квалификации или переобучение.

Рассмотрим эти этапы подробнее. В первую очередь необходимо объяснение принципов работы уже настроенной системы перед первым тестированием с ключевыми пользователями.

Следующая веха, инсталляция на стендовый сервер, важна как промежуточный этап, на котором снова будут привлекаться ключевые пользователи, только уже для описания работы и интерфейсов в гораздо более детализированном виде, для идентификации и устранения всех остающихся критических замечаний перед ОПЭ. Разумеется, к этому моменту все ИТ-специалисты, которые будут заниматься поддержкой системы, уже должны в деталях знать внутреннюю архитектуру системы и быть способными проводить ее доработку (или формировать требования на доработку) в случае необходимости.

И наконец, наиболее масштабное обучение организуется уже для всех остальных сотрудников, которые будут работать с системой.

В данном случае система представляется не с точки зрения внутренней архитектуры, а с точки зрения пользователя, т. е. описываются принципы ввода, обработки и получения данных.

Для организации процесса обучения необходимо выполнить следующие шаги.

**Шаг 1. Выделение и обучение ключевых пользователей.** В первую очередь выделяются ключевые пользователи — один-два специалиста от каждого структурного подразделения (деление организации может выполняться не с организационной, а с логической точки зрения).

Важно, что в зависимости от масштаба и специфики системы эти специалисты могут работать не только с пользовательскими аспектами системы, но и самостоятельно разворачивать функциональные компоненты и модули, конфигурировать их по требованиям пользователей. Именно поэтому их обучение, аттестацию и вовлечение в работу крайне важно проводить гораздо раньше ОПЭ, ведь основное число сотрудников увидят систему и ознакомятся с принципами работы в ней лишь во время опытно-промышленной и промышленной эксплуатации.

Проведение учебных курсов по основам работы с системой для сформированной группы ключевых пользователей (по функциональным направлениям) организуется специалистами исполнителя (включая аттестацию). В качестве формы организации обучения может быть выбрана очная (преподавателем от команды исполнителя в офисе компании заказчика или подрядчика) либо дистанционная (когда личное присутствие преподавателя нецелесообразно или невозможно). Все популярнее становится проведение вебинаров и телеконференций, позволяющих экономить временные и денежные ресурсы.

Их поддержка будет крайне важна на следующем этапе при проведении тренингов для остальных пользователей: помимо предоставления комментариев по пользованию системой и о необходимости доработок, они могут помочь остальным пользователям разобраться с работой в системе и не обращаться каждый раз к команде сопровождения со стороны подрядчика.

**Шаг 2. Обучение рядовых пользователей.** К этому моменту необходимо, чтобы достаточное количество специалистов группы внедрения от заказчика было способно самостоятельно разворачивать функциональные компоненты системы или модуля на рабочих местах пользователей, проводить обучение конечных пользователей, оказывать консультации по стандартной конфигурации системы, а также сопровождать ее в оперативном режиме.

Принципы проведения тренингов могут значительно различаться в зависимости от масштабов компании, проекта/системы и пожеланий заказчика, однако на практике они чаще всего проводятся

несколько раз, не более чем для 10—15 человек единовременно. При этом группы обычно формируются под каждую отдельную роль пользователя. Основные темы включают общий обзор задач системы, рассмотрение ее интерфейса, основных возможностей, затем специфической для конкретной роли пользователя функциональности и секции «вопрос — ответ». Иногда после этого проводится настройка необходимых параметров (например, интеграции с учетными записями Outlook, настройка синхронизации) уже на местах пользователей, однако необходимость этого зависит от конкретной системы.

**Шаг 3. Повышение квалификации и переобучение.** Следует учитывать также необходимость последующего периодического обучения с целью повышения квалификации сотрудников.

Условно можно выделить три основные категории такого обучения.

1. Плановое обучение после изменений в системе (при добавлении дополнительных возможностей, внедрении нового модуля и т. п.).

2. Плановое повышение квалификации: для совершенствования знаний работников, уже обладающих определенными профессиональными навыками работы с системой.

3. Обучение по запросу — относится к сфере поддержки пользователей на этапе эксплуатации системы и проводится в случае, если пользователь в своей работе столкнулся с конкретными проблемами или нуждается в проведении методического занятия.

Отметим, что для работы с некоторыми классами систем (как правило, если их деятельность связана с управлением объектами, представляющими в случае выхода из-под контроля опасность для жизни и здоровья людей) просто проведения обучения недостаточно. В таких случаях организуются дополнительные тестирования знаний, успешное прохождение которых дает допуск к пользованию системой при предоставлении персонального сертификата. Примером могут являться системы, автоматизирующие управление безопасностью железнодорожного движения и авиаперевозок, запуск космических объектов, химическое производство и подобные виды деятельности.

И разумеется, при планировании обучения следует учитывать такие аспекты, как опыт сотрудников, предполагаемые и видимые знания, а также навыки руководства и управления, организационные способности.

#### **3.5.4. Развёртывание системы на рабочих местах**

Следующим шагом организуется развертывание модуля для дальнейшей его опытно-промышленной эксплуатации. Успех данного этапа целиком зависит от проведенной ранее работы. Так, если уже

был реализован пилотный проект в другом подразделении, филиале или в рамках другой функциональной области, этот опыт поможет компании воспользоваться рекомендациями по развертыванию системы и выполнить оптимальные настройки. Кроме того, инсталляция проводится еще до тестирования, а значит, результаты используются исключительно в целях совершенствования системы в так называемой песочнице (*sandbox*).

---

**«Песочница» (*sandbox*)** — закрытое для доступа извне виртуальное пространство, в котором можно работать с ПО без риска изменения системных файлов, в связи с чем они используются для запуска кода, еще не прошедшего тестирование.

---

Настройки «песочницы» и ее элементы дублируют основные элементы и возможности реальной системы для первоначальной отладки и проверки кода в виртуальной среде перед его переносом в продукт. Это помогает избежать значительного числа рисков и сбоя настроек сложной конфигурации системы.

Набор ресурсов системы, выделяемый для «песочницы», жестко ограничен в части доступа к объему памяти, к сети, обмену информацией с основной системой. Это обусловлено в первую очередь нестабильностью работы тестируемого кода и значительной нагрузкой на систему. По этой причине «песочницы» часто используют исключительно для проверки отдельных критических частей кода.

Сделанные в «песочнице» настройки используются при полноценном развертывании модуля системы после тестирования перед ОПЭ. Модули системы устанавливаются на стендовый сервер с проведением основных необходимых настроек под подлежащие автоматизации бизнес-процессы предприятия заказчика. По результатам этих действий подрядчик готовит описание оптимальных настроек сервера БД и рекомендации по развертыванию рабочих мест, которое и проводится следующим шагом.

Перед развертыванием системы необходимо формирование командой внедрения следующей информации:

- выделение основных блоков функциональности, их запусков, взаимозависимостей;
- выделение основных блоков работ, требуемых для запуска функциональности системы, и их взаимозависимости;
- расстановка контрольных точек для верхнеуровневого мониторинга процесса;
- информация по исполнителям работ (зонам ответственности).

При переходе к стадии эксплуатации должны быть осуществлены следующие действия:

- проверка необходимых предпосылок для перехода в производственную систему;

- описание мероприятий по переходу в продуктивную систему;
- описание критериев, на основании которых будет приниматься решение о начале работы пользователей;
- описание порядка действий в случае возникновения ошибок, сбоев или нарушения работы системы.

При развертывании системы на рабочих местах выполняется следующее:

- установка и подключение технических средств;
- установка и конфигурация ПО;
- развертывание баз данных, служебных программ;
- доработка интерфейса и процессов, настройка профилей пользователей, настройка отчетов;
- установка прав доступа;
- пуско-наладочные работы и окончательная отладка конфигурации.

Потенциальные сложности могут возникнуть также, если несколько смежных подпроектов (например, в части разных модулей системы) реализуются несколькими подрядчиками. Тогда следует уделять пристальное внимание управлению интеграцией всех проектов на протяжении всего времени их реализации и при подготовке к ОПЭ — в особенности.

### 3.5.5. Основные виды тестирования

Прежде чем приступить к промышленной эксплуатации системы, важно проверить ее работоспособность во всех предусмотренных спецификацией режимах, а также в экстремальных условиях. Если планируется одновременный распределенный ввод данных сотней пользователей, необходимо, как минимум, проверить правильность обработки системой одновременной активности именно не менее сотни пользователей. Подобная идея лежит в основе нагрузочного тестирования.

Нагрузочное тестирование необходимо для предсказания поведения системы в реальных и экстремальных условиях, выявления ошибок, отслеживания производительности и доступности при изменении различных параметров работы с системой. Среди других его задач:

- оценка производительности и работоспособности приложения на этапах:
  - разработки,
  - опытно-промышленной эксплуатации,
  - сопровождения и эксплуатации (выпуск новых релизов, патчей);
- оптимизация приложений;
- подбор оптимальной программно-аппаратной платформы и конфигурации сервера.

Перечислим еще несколько ключевых категорий тестирования.

Тестирование производительности (*load testing*) — моделирование ожидаемой интенсивности использования путем распределенной работы большого числа пользователей с различными модулями системы.

Среди возможных целей этого типа тестирования:

- определение времени выполнения операций с заданной интенсивностью (на разных нагрузках);
- определение максимально достижимой производительности системы.

Стрессовое тестирование (*stress testing*) — определение стабильности системы при интенсивности работы, превышающей плановые или стандартные значения. Стрессовое тестирование по своей сути проверяет, возвращается ли (и насколько быстро) система после запредельной нагрузки к нормальному режиму работы, тестируется способность системы к регенерации.

Стрессовое тестирование особенно важно для компаний, в которых стоимость отказа системы в экстремальных условиях может быть очень велика, а стандартного тестирования разработчиками недостаточно для эмуляции тех условий, в которых произойдет отказ системы.

Среди возможных целей этого типа тестирования:

- оценка динамики производительности при нестандартных ситуациях: аварийном изменении серверной конфигурации либо резком повышении числа пользователей и выполняемых одновременно операций;
- определение условий и скорости возвращения системы к нормальному режиму работы после окончания стрессового тестирования.

Тестирование надежности (*reliability testing*) — определение длительности бесперебойной и безошибочной работы системы. Цели этого типа тестирования следующие:

- оценка стабильности системы при многочасовом тестировании со стандартным средним уровнем нагрузки для определения:
  - утечек памяти,
  - некорректных конфигурационных настроек,
  - случаев перезагрузки сервера и других требующих устранения проблем.

Конфигурационное тестирование (*configuration testing*) — оценка степени влияния на производительность изменений в конфигурации и различной балансировки нагрузок. Конфигурационное тестирование относится как к серверному уровню (совместимость с окружением), так и к клиентскому уровню (например, кросс-платформенное или кросс-браузерное тестирование). Цели этого типа тестирования:

- оценка обработки ошибок и исключительных ситуаций, а также «узких мест» отдельных модулей и компонентов системы при не-пропорциональных нагрузках;
- определение оптимальной конфигурации оборудования, которая бы обеспечивала требуемые характеристики производительности и отклика системы;
- проверка определенных компонентов системы на предмет совместимости с указанным в спецификации оборудованием, операционными системами и программными продуктами внешних поставщиков.

Последовательности действий для проведения данного вида тестирования такова.

1. Создается матрица покрытия с описанием всех возможных конфигураций системы.

2. Выполняется приоритезация конфигураций.

3. И наконец, в ходе тестирования в соответствии с имеющимися приоритетами организуется проверка всех основных конфигураций.

Объемное тестирование (*volume testing*) — тестирование программного обеспечения системы на предмет стабильности обработки определенного объема данных. Цели этого типа тестирования:

- измерение динамики выполнения операций при увеличении объемов данных в базе данных, постепенном росте числа запросов пользователей;
- определение максимального числа пользователей, которые могут работать с приложением без снижения производительности.

Отсутствие или ошибка проведения подобных видов тестирования значительно увеличивают риски и могут даже приводить к ситуациям, когда после внедрения срок эксплуатации систем не превышает даже года в связи с невозможностью конфигурации системы справляться с необходимыми задачами и нагрузками. Именно для нивелирования подобных рисков необходимо своевременно организовывать тестирование еще на этапе внедрения (хотя его проведение возможно (и часто проводится) и для уже эксплуатируемых систем).

### 3.5.6. Опытно-промышленная эксплуатация

Опытно-промышленная эксплуатация представляет собой тестирование в полной функциональности и полной нагрузке для определенного количества пользователей. Ее основной целью является апробирование работы пользователей в системе в реальных производственных условиях. Это означает, что если по спецификациям системы предполагается, что она будет обрабатывать 500 000 записей в день, необходимо проверить корректность обработки именно этого количества записей. При этом важные задачи:

- тестирование модуля системы в условиях, максимально приближенных к реальным условиям промышленной эксплуатации (в том числе, при необходимости, в интеграции с другими модулями или внешними системами) — нагружочное тестирование, рассмотренное ранее;
  - проведение множественных расчетов по фактической производственной информации с применением соответствующих программно-аппаратных средств, предусмотренных техническим проектом (проектным решением);
  - достижение наиболее полного охвата бизнес-процессов, автоматизируемых подразделений предприятия.

При условии соблюдения ранее определенных требований к предыдущим этапам и их успешном завершении в первую очередь необходимо провести:

- ввод наиболее актуальной информации по каждому модулю ввода в систему (подготовка которой проведена на предыдущем этапе);
- отработку действий пользователя в параллельном режиме с уже имеющимися приложениями (это проводится непосредственно на рабочих местах пользователей и включает дополнительную, но необходимую нагрузку для пользователей, которые в короткий период времени должны адаптироваться к работе в другой системе (а также помочь адаптировать ее для оптимальной и эффективной работы));
- интеграцию внедряемого модуля с уже существующими модулями и (или) внешними системами, внедренными ранее;
- разработку и согласование регламента взаимодействия внутренних подразделений предприятия-заказчика в рамках ОПЭ и дальнейшей промышленной эксплуатации (помимо схемы и порядка передачи функций с обязательным определением зон ответственности за актуальность и корректность данных).

### Пример

Несмотря на то что официально «владельцем» системы будет являться одно подразделение (или даже одно ответственное лицо), ввод и редактирование информации будут осуществляться распределенно и часто одновременно различными категориями пользователей. Соответственно, существует практика определения зон ответственности за различными подразделениями, которые вне зависимости от источника ввода будут обладать исключительными правами на удаление и изменение информации. Этот механизм может быть организован многими способами, самым «простым» из которых является четкое определение алгоритма взаимодействия подразделений в рамках работы с системой (сотрудники будут сами знать, какую информацию в какой момент они будут вводить и обрабатывать). Во многом подобный механизм исходит из принципов «презумпции невиновности», предполагая что каждый специалист достаточно

компетентен и отвечает за каждое совершенное им действие (как в целом в работе, так и в системе).

Другим, менее распространенным, способом управления взаимодействием подразделений (и главное — контроля данных) является распределение прав доступа таким образом, что у наиболее важной информации будет только один источник ввода/проверки. И хотя другие пользователи могут изменять поле (например, вносить данные об инвестициях и договорах), в системе эта информация будет «висеть» в режиме ожидания до момента ее одобрения/авторизации действия ответственным за поле пользователем. Однако, несмотря на значительную гарантию достоверности данных, подобная схема крайне неэффективна с точки зрения использования ресурсов, особенно в условиях большого числа и масштаба операций.

---

Основная часть (как по объему, так и по трудозатратам) проводимых в ОПЭ мероприятий предусматривает сопровождение заказчиком системы на рабочих местах пользователей с формированием перечня замечаний ключевых пользователей (при их «переводе» из бизнес-терминологии в более техническую область) и устранением этих замечаний.

### Пример

---

Замечания пользователей:

В список партнеров нельзя добавить университеты (нет пункта в списке).

Требование к реализации подрядчиком:

Создать в типе организаций «Прочие» наследующий его свойства подтип «Университет» с дополнительными полями «Ректор», «Факультет», «Кафедра». Настроить общедоступный отчет по всем университетам.

---

Устранением замечаний и внесением изменений в систему на этом этапе (длящемся, как и в случае тестовой эксплуатации, не менее одного месяца) занимается команда внедрения со стороны исполнителя, которая помимо простого соответствия системы всем требованиям и бизнес-процессам должна удостовериться в устойчивости работы системы при нагрузке, максимально приближенной к реальной промышленной нагрузке этапа эксплуатации (с исключением риска отказа системы в момент ее пика). Чтобы перейти к полноценной промышленной эксплуатации, система также должна безошибочно функционировать в параллельном режиме и при интеграции с внешними приложениями в течение всего срока ОПЭ. При наличии аналогичных *legacy*-приложений об успешном внедрении можно говорить, если результаты обработки данных в них и в новой системе совпадают и (или) различия объяснимы и некритичны.

Для окончания ОПЭ необходимо подтверждение того, что модуль системы функционирует в параллельном режиме с существую-

щими внешними приложениями (при их наличии) в течение срока опытно-промышленной эксплуатации. Вторым условием является подтверждение, что любые отклонения (в случае их обнаружения) признаны некритичными и объяснимыми.

### **3.5.7. Приемо-сдаточные испытания и интеграционное тестирование**

Именно передача системы из ОПЭ в промышленную эксплуатацию (в частности, путем проведения приемо-сдаточных испытаний) является целью комплексного проекта по автоматизации деятельности компании. Приемо-сдаточные испытания такого типа называют также интеграционным тестированием. Такое тестирование обеспечивает комплексную проверку реализации проектных решений системы.

Цели интеграционного тестирования:

- проверка корректности функционирования бизнес-процессов и функций;
- уточнение настроек бизнес-процессов в системе;
- уточнение и доработка настроек пользовательского интерфейса;
- проверка функций начальной загрузки в систему основных и переменных данных;
- проверка полноты переносов настроек посредством транспортной системы;
- проверка работоспособности интерфейсов с внешними системами;
- проверка корректности проектной и эксплуатационной документации;
- определение правильности функционирования системы на реальном объеме данных в реальном времени.

Тестирование проводится на продуктивном наборе данных для проверки работоспособности всей системы и правильности настройки интерфейсов взаимодействия с внешними системами.

Как уже было сказано, именно этот этап является переходным между этапом ОПЭ и реальной эксплуатацией. С организационной точки зрения перед началом промышленной эксплуатации необходимо удостовериться в наличии следующего набора документов:

- рабочий журнал опытно-промышленной эксплуатации с перечнем замечаний ключевых пользователей и результатами их устранения (заполняемый командой внедрения и согласовываемый затем с представителями заказчика);
- протокол об окончании ОПЭ и готовности к промышленной эксплуатации совместно с Актом приемки-передачи работ;
- приказ по предприятию заказчика о приеме системы в промышленную эксплуатацию (с составом функций, описанием про-

граммно-аппаратной платформы, списком ответственных за работу модуля и списком подразделений-пользователей, а также с порядком и сроками перехода персонала на работу в системе, включая аспект разработки и принятия новых форм документов в случае необходимости).

После этого использование системы уже не имеет признаков проектной деятельности и ограничений по срокам, содержанию или стоимости. Однако такие последующие этапы ЖЦИС, как сопровождение эксплуатации, модернизация и утилизация системы, будут по своей сути являться проектами и могут также выполняться не только за счет внутренних ресурсов компании, но и (что случается гораздо чаще) с привлечением внешних подрядчиков.

### 3.6. Эксплуатация

Этап эксплуатации системы является наиболее ожидаемым для бизнес-заказчиков, поскольку только в этот момент они начинают получать возврат инвестиций и видеть реальный результат внедрения. Однако это и самый опасный этап, так как полученный результат может (и скорее всего, будет) не соответствовать новым ожиданиям и представлениям как о функциональности, так даже и о внешнем виде и интерфейсе системы. Технические специалисты и подрядчик фокусируются на решении задач бесперебойной работы, обслуживания баз данных, а в случае необходимости — на донастройке системы (что означает одновременное наступление стадии модернизации, продолжающейся параллельно с эксплуатацией). Но бизнес-заказчик и спонсор системы, как и все участники процесса сбора требований, часто имеют собственное представление о целевом состоянии системы. Именно поэтому критически важно с самого начала вовлекать в проект специалистов, переводящих пожелания бизнеса в задачи для разработчиков (данная роль в английской терминологии получила название *IT/Business Relationship Manager*).

Стадия сопровождения, приводящая к изменениям системы в процессе эксплуатации (по окончании приемки работ), охватывает несколько аспектов:

- устранение замечаний, не приводящих к изменению ТЗ;
- обновления (по сути — новые версии системы), выпускаемые при накоплении критического объема доработок;
- увеличение производительности системы.

Важно отметить, что на сегодняшний день основной задачей компаний-подрядчика является не просто настройка системы, но передача накопленных и базирующихся на опыте множества проектов знаний и методик работы с внедренным программным решением. По окончании проекта специалисты компании-заказчика должны

быть способны самостоятельно осуществлять полноценную поддержку и развитие системы. В первую очередь это относится к развитию уже созданных модулей, совершенствованию функциональности, реализации новых задач для новых категорий пользователей, формированию аналитических отчетов и панелей мониторинга. Разумеется, часто заказчик принимает решение о продолжении сотрудничества с подрядчиком, внедрявшим систему, в части ее сопровождения. Это относится как к базовым функциям (построению дополнительных типов отчетов или изменению параметров настроек), так и к технологической и методической поддержке.

### 3.6.1. Сопровождение эксплуатации

**Авторский надзор.** В условиях промышленной эксплуатации в течение определенного договором времени после сдачи модуля системы со стороны исполнителя-подрядчика проводится наблюдение за его функционированием и, по мере необходимости, оказание помощи специалистам заказчика по устранению замечаний. В свою очередь, уже обученные ответственные за систему сотрудники заказчика в тесном взаимодействии с ключевыми пользователями формируют перечень замечаний, осуществляют технологические обновления модуля (системы) и ведут документирование эталонных версий модулей. При этом предприятие-подрядчик включается в работу только при необходимости значительных доработок либо если таковое определено заключенным контрактом на поддержку. Это объясняется тем, что ключевая функциональность, необходимая для успешной и бесперебойной работы системы, и программно-аппаратная база уже были сформированы на предыдущих этапах и специалисты заказчика уже в состоянии осуществлять стандартную техническую поддержку на своем предприятии самостоятельно.

Именно в это время устраняются основные возникающие замечания, ошибки и неточности первоначальных расчетов нагрузки на систему. Как правило, длительность этапа авторского надзора составляет не менее года, и, в отличие от фактора масштаба и сложности проекта, отсутствие или несвоевременное представление замечаний не влияют на продолжительность этапа авторского надзора за промышленной эксплуатацией модуля.

Для этой и последующих стадий сопровождения могут применяться специальные **метрики оценки работ** со стороны предприятия-подрядчика, отвечающего за сопровождение. Четыре основные метрики (актуальные для всего жизненного цикла) были выделены Институтом программной инженерии университета Карнеги-Меллон (SEI CMU): *размер, усилия, расписание и качество*.

Однако они могут дополняться и другими параметрами:

- **анализируемость:** оценка не предусмотренных изначально усилий или ресурсов, необходимых для диагностики недостатков

или причин сбоев, а также для идентификации тех фрагментов программной системы, которые должны быть модифицированы;

- *изменяемость*: оценка усилий, необходимых для проведения заданных модификаций;
- *стабильность*: оценка случаев непредусмотренного поведения системы, включая ситуации, обнаруженные в процессе тестирования;
- *тестируемость*: оценка усилий персонала сопровождения и пользователей по тестированию модифицированного программного обеспечения.

**Техническая поддержка.** Техническая поддержка системы начинается после приема в промышленную эксплуатацию и может продолжаться до снятия с эксплуатации и утилизации системы. Как правило, варианты осуществляющей поддержки варьируются по процентам от стоимости приобретенного ПО в зависимости от набора оказываемых услуг, а объем и периодичность выполнения работ на данном этапе определяются отдельным договором.

В качестве поддержки могут предоставляться следующие услуги в различных комбинациях в зависимости от прописанных в договоре условий:

- консультирование по «горячей линии» (телефон, *e-mail*, *skype*) по определенному в договоре графику;
- консультирование по вопросам программно-аппаратной платформы системы;
- создание новых учетных записей пользователей системы;
- настройка форм отчетов и панелей мониторинга для определенных групп пользователей заказчика;
- диагностика и устранение неисправностей (с учетом гарантии на сами программные решения);
- уведомление о выпуске обновлений и их загрузка через Интернет;
- замена ключей электронной защиты, переустановка ПО в случае необходимости (например, при замене компьютерной базы заказчика);
- миграция данных и настройка интеграции с новыми системами предприятия-заказчика;
- помошь в формулировании требований к новым программным решениям предприятия заказчика в части интеграции с внедренной системой либо в постановке и формулировании задач (на уровне ТЗ) по включению новых бизнес-процессов в систему.

**Постгарантийное сопровождение.** Постгарантийное сопровождение предполагает заказываемые у производителя работы, не предусмотренные изначальным контрактом на автоматизацию процессов или системную интеграцию (рис. 3.20). Сопровождение программного обеспечения определяется стандартом IEEE Standard

for Software Maintenance (IEEE 1219) как «модификация программного продукта после передачи в эксплуатацию для устранения сбоев, улучшения показателей производительности и (или) других характеристик (атрибутов) продукта, или адаптации продукта для использования в модифицированном окружении». Таким образом, функционирование программного продукта поддерживается на протяжении всего периода его эксплуатации.



Рис. 3.20. Работы процесса сопровождения по стандарту IEEE 1219

В рамках заключаемого на этом этапе договора исполнитель оказывает помощь по устранению замечаний и выделяет в случае необходимости специалистов для выполнения доработок (например, в части новой функциональности или отчетов). Однако если в течение предыдущих этапов ЖЦИС исполнителем проводились мероприятия по обучению заказчика и вовлечению его в проект, на фазе постгарантийного сопровождения специалисты заказчика будут способны:

- сопровождать модуль системы в ходе промышленной эксплуатации на участках;
- формировать перечень замечаний ключевых пользователей;
- выполнять работы по устранению возникших замечаний (оказывать помощь по устранению замечаний);
- осуществлять своевременные обновления модуля системы в компании исполнителя (прежде всего — технологические);
- хранить и вести эталонные версии модуля системы и программной документации (постгарантийное сопровождение старых версий может продолжаться и после выпуска новых версий программных продуктов, однако чаще поддерживаются исключительно новые версии).

Со стороны компании-исполнителя в данном случае осуществляются постановка задачи, анализ проблем в предметной области, планирование и реализация необходимых доработок и прочие активности.

В договорах на постгарантийное сопровождение, как правило, прописываются условия, касающиеся:

- возможности приобретения дополнительных клиентских лицензий;
- возможности поддержки устаревших версий программных продуктов, на которых основана система;
- оплаты командировочных расходов компании-подрядчика при необходимости выезда в филиалы других регионов и государств;
- скидок и особых условий при продлении контракта.

При рассмотрении работ по сопровождению системы будем использовать руководство SWEBOK в области знаний «Поддержка ПО» (*Software maintenance*) как содержащее достаточно полный набор материалов на эту тему.

SWEBOK приводит ряд процессов (работ, практик), которые являются уникальными для деятельности по сопровождению.

- **Передача** (*Transition*). Внутренние коммуникации разработчиков и группы поддержки для грамотной координации процесса передачи программного решения на сопровождение.
- **Принятие/отклонение запросов на модификацию** (*Modification Request Acceptance/Rejection*). Рассмотрение таких характеристик, как: объем и (или) сложность требуемых изменений, необходимые для их реализации активности. Решения по принятию или отклонению запросов на модификацию могут также основываться на анализе приоритетности, оценке обоснованности, отсутствии ресурсов (в том числе отсутствии возможности привлечения разработчиков к решению задач по модификации при реальном наличии такой потребности), внесении в план к реализации следующих релизов.

• **Средства извещения персонала сопровождения и отслеживания статуса запросов на модификацию и отчетов об ошибках** (*Modification Request and Problem Report Help Desk*). Поддержка конечных пользователей, в том числе анализ приоритетности и стоимости модификаций, связанных с поступившим запросом или сообщенной проблемой.

• **Анализ влияния** (*Impact Analysis*). Анализ возможных последствий изменений, вносимых в существующую систему, при идентификации всех связанных с ней систем и программных продуктов, на которых эти изменения могут потенциально отразиться.

• **Поддержка программного обеспечения** (*Software Support*). Консультирование пользователей, проводимое по их информационным запросам (*request for information*), например, в отношении биз-

нес-правил, проверки содержания данных и получаемых сообщений о проблемах (ошибках, сбоях, непредусмотренном поведении, не-понимании принципов функционирования системы).

• **Контракты и обязательства.** Закрепленные в договорной форме обязательства по определенным параметрам оказания услуг сопровождения (в том числе классическое соглашение об уровне предоставляемого сервиса, SLA).

**Обновление и релизы.** Неотъемлемой частью сопровождения является выпуск обновлений и релизов программ/конфигураций. В них, в частности, могут быть реализованы новые правила обмена данными, настроены новые формы регламентированной отчетности, усовершенствованы интерфейсы, адаптированы функциональные возможности и произведены другие доработки.

Подобного вида изменения вносятся в ПО либо подрядчиком в рамках постгарантийного сопровождения при накоплении критического числа замечаний, либо вендором программного продукта в рамках планового выпуска его очередной версии. Это может проводиться в рамках подготовки нового технического дистрибутива, в рамках выпуска локализованной под конкретную страну или регион конфигурации, в рамках отраслевого решения, в рамках обновления определенного модуля и пр.

Если речь идет о *выпуске версий при накоплении критического числа замечаний*, то сопровождение подобного рода, как правило, складывается из следующих этапов.

1. Формирование заявок для специалистов, работающих в рамках сопровождения.

2. Планирование работ (выявление проблем и требований, объема и содержания задач).

3. Оказание услуг (согласование/уточнение требований, реализация работ, консультации, обучение, доработки).

4. Сдача-приемка работ.

Говоря о плановых обновлениях версий программного обеспечения вендорами, приведем пример компании 1С.

### Пример

В 2012 г. 1С выпустила новую конфигурацию «Бухгалтерия предприятия» (редакция 3.0), и в пресс-релизе были отмечены следующие внесенные изменения:

1) повышение удобства работы. Новые пиктограммы в интерфейсе пользователя, возможность настройки панелей действий и навигации. Появление внутренних ссылок для каждого из объектов информационной базы;

2) развитие функциональности учета заработной платы. Начисление заработной платы, НДФЛ и страховых взносов стало проводиться одним документом автоматически при начислении заработной платы. Разделы

«Зарплата» и «Кадры» объединены в один раздел, за счет чего все кадровые документы стали доступны на карточке сотрудника;

3) развитие прав доступа. Добавлена функция доступа к информационной базе в режиме просмотра без прав на внесение изменений;

4) работа через Интернет по модели SaaS. Для работы начиная с версии 3.0 более не требуется установка платформы и информационных баз на компьютере пользователя. Вместо этого доступен запуск программы через веб-браузер с сайта компании-поставщика «облачного» сервиса;

5) повышение удобства работы при выполнении длительных операций. Возможность выполнения операций в фоновом режиме;

6) новые средства защиты персональных данных. Изменения внесены в связи с требованиями Федерального закона.

Таким образом, важность удаления должного внимания выпускаемым обновлениям несомненна, и активности по их реализации также несут в себе все признаки проектной деятельности.

---

**Увеличение производительности системы.** Во время проектирования и внедрения системы для уточнения ее возможностей определяется, как быстро приложение должно работать, какой объем памяти и какую загрузку процессора использовать, какие другие характеристики системы можно считать приемлемыми.

В течение срока эксплуатации и сопровождения системы ее окружение меняется, и прежние параметры производительности становятся недостаточными для успешной работы. В частности, это может относиться к таким аспектам, как:

- *время отклика* — время, необходимое серверу для выдачи ответа на произведенный запрос;
- *пропускная способность* — число запросов, которые могут быть обработаны за единицу времени; часто при измерении данного параметра определяется число запросов / логических транзакций в секунду;
- *использование ресурсов* — потребление приложением серверных/сетевых ресурсов (ЦП, память, дисковое пространство);
- *рабочая загрузка* — общее количество пользователей (либо только активных пользователей), объем данных и транзакций.

Соответственно, рано или поздно владельцы системы сталкиваются с необходимостью совершенствования этих характеристик. Иногда подобный подход является проактивным, когда принимаемые меры направлены на оптимизацию, снижение операционных расходов. Однако чаще эти действия являются уже реактивными и проводятся, когда стоимость владения системой и затраты на нее становятся неприемлемыми и поднимается вопрос о целесообразности ее эксплуатации.

Проводимое в рамках сопровождения системное улучшение параметров получило название *инженерии производительности (performance engineering)*. Изначально данная область разрабатывалась

в рамках направления системотехники и включала в себя единый набор ролей, знаний, практик, инструментов и результатов каждого этапа ЖЦ системы. Основная цель состояла в том, чтобы гарантировать соответствие создаваемого программно-аппаратного решения нефункциональным требованиям к его производительности.

В настоящее время область применения инженерии (и реинжиниринга) производительности в значительной части сфокусирована на проектах сопровождения и модернизации системы, когда требования уже проверены временем в ходе эксплуатации и могут быть определены более конкретно.

Среди задач инженерии производительности:

- повышение окупаемости бизнеса с помощью обеспечения своевременной обработки необходимых объемов транзакций;
- своевременное выявление потенциальных проблем масштабирования и производительности и их устранение;
- снижение стоимости поддержки ПО и внеплановых затрат через своевременное выявление проблем производительности;
- предотвращение задержек, вызванных проблемами с производительностью, при развертывании систем.

При планировании производительности необходимо понимать, какие ресурсы системы доступны в текущий момент. Для этого оцениваются:

- *сетевое обеспечение* (в том числе пропускная способность);
- *аппаратное обеспечение* (характеристики серверов, рабочих станций);
- *зависимости ресурсов* (число доступных соединений баз данных, веб-сервисов);
- *совместно используемые ресурсы* (в том числе принципы распределения ресурсов между разными элементами системы);
- *проектные ресурсы* (финансовые и человеческие ресурсы) для поддержки системы и осуществления проекта по повышению ее производительности.

В процессе могут быть задействованы администратор системы, системный архитектор, разработчики, тестировщики и другие члены проектной команды. Именно они производят моделирование/прототипирование желаемого результата, проектирование новых параметров системы, доработку программного кода, мониторинг, создание надежных тестов производительности, различного рода тестирование и настройку системы до необходимых значений.

### Пример

Вполне конкретное требование к определению планового значения отклика системы на запрос может выглядеть следующим образом:

«Для сценария использования А отклик системы на корректный запрос пользователя не должен превышать:

- 5 секунд для нагрузки в 250 активно использующих систему пользователей (200 онлайн-пользователей в 95% случаев);
  - 10 секунд для пиковой нагрузки в 500 активных пользователей (400 онлайн-пользователей в 90% случаев)».
- 

В общем случае для формирования подобных требований определяется типичный бизнес-день, разбитый на часы, для формирования представления о том, какова динамика нагрузки на систему в течение дня. В результате можно получить конкретные предложения по улучшению (например, «распараллелить работу модуля, запустив ее на четырех процессорах вместо одного») и конкретные результаты (например, «повысить таким образом эффективность загрузки ресурсов модуля, получив прирост производительности в 5 раз»).

### 3.6.2. Модернизация

**Стратегии управления legacy-системами.** В течение периода эксплуатации системы происходят неизбежные изменения как вне организации (в том числе изменение рынка, контрагентов), так и внутри организации (к примеру, смещение бизнес-приоритетов в другую индустрию, на другой рынок и пр.). Растет объем обрабатываемой информации, объем файлов, снижается пропускная способность, оборудование выходит из строя и появляется новое — все эти факторы неизбежно влияют на актуальность самой системы и решаемых ей задач. В то время когда системы уже не способны с должной степенью эффективности решать задачи бизнеса, говорят об «унаследованных» системах.

---

**Унаследованные системы** (англ. *legacy systems*) — системы, более не соответствующие текущим потребностям бизнеса, но по-прежнему эксплуатирующиеся компаниями. Как правило, в их основе лежат устаревшие технологии и платформы, и не представляется возможным далее совершенствовать и видоизменять их для соответствия требованиям безопасности, новому аппаратному обеспечению, обновленным операционным системам.

---

Чаще всего модификация подобных приложений требует значительных затрат времени и финансовых средств, в то время как их замена может требовать даже реинжиниринга бизнес-процессов организации. В таком случае требуется доработка системы, для чего необходимо заново пройти все фазы жизненного цикла — от сбора и формирования требований до сопровождения и эксплуатации — и, соответственно, снова определять сроки, ресурсы и содержание. Результат подобной доработки будет уникальным, и потому можно говорить об отдельном проекте по модернизации. Этим проектом

может заниматься как команда внедрения, так и абсолютно другие компании-подрядчики (особенно если речь идет о модернизации системы, созданной несколько лет назад, ведь за этот срок организация-подрядчик может уже прекратить свое существование).

Модернизация системы подобного рода позволяет продлить срок ее эксплуатации, снизить совокупную стоимость владения (TCO), расширить функциональные возможности, усовершенствовать программно-аппаратную платформу минимальными доступными средствами без остановки процесса эксплуатации системы. В ходе модернизации устраняются многие проблемы, возникавшие при эксплуатации. Однако необходимость реализации определенных доработок, число которых уже достигло критического значения, является лишь одной из возможных причин старта модернизации системы.

Другим важным фактором (триггером) является аудит информационных систем, проводимый организацией самостоятельно или с привлечением внешних экспертов и консультантов. Результаты аудита могут сочетаться с реинжинирингом бизнес-процессов компании, разработкой новой целевой прикладной архитектуры или же обновлением ИТ-стратегии в целом. Принятию решения о модернизации в обязательном порядке должно предшествовать определение сроков, стоимости и состава работ по доработке, так как возможна ситуация, когда списание системы и внедрение новой в конечном итоге окажется лучшим вариантом, чем изменение существующей. Ведь в целом задачи модификации, будучи на первый взгляд простыми (хотя и трудоемкими), представляются компаниям, как правило, несвоевременными либо неперспективными. Поколения сотрудников (как заказчика, так и компаний-подрядчика) меняются, работавшие при внедрении специалисты уходят, а в случае некорректного документирования системы при ее создании результаты попыток совершенствования ПО могут быть достаточно плачевными.

Таким образом, среди внешних и внутренних факторов, которые могут служить своеобразными индикаторами необходимости модификации систем, можно перечислить следующие (на примере экономической ИС):

- изменения в системе документооборота, формате и структуре формируемых документов;
- изменения в перечне задач, принадлежащих к основной функциональности системы, или в методах их решения;
- мнения пользователей о работе с системой, качестве обработки данных и результирующей информации;
- информация специализированного ПО (например, в составе СУБД и ОС) по статистике работы системы, ее быстродействию и отказоустойчивости;
- характеристики организации и внешней среды (выпуск новых изделий, появление новых технологических ограничений);

- изменение законодательства и требований регуляторов (например, в части стандартов учета);
- уход из компании и недостаток на рынке специалистов со знаниями, необходимыми для поддержки конкретной *legacy*-системы;
- слишком высокая стоимость поддержки системы или совокупная стоимость владения;
- отсутствие технических возможностей (высокая сложность) интеграции с новыми, внедряемыми на предприятии системами;
- рыночная конкуренция и бизнес-приоритеты (например, интернет-магазин, который фокусируется на удовлетворении потребностей пользователей, может принять решение о модернизации пользовательского интерфейса и системы принятия и обработки заказов клиентов);
- осуществляемые процессы слияний и поглощений M&A (так, в случае объединения информационных систем с другой компанией требуется полный пересмотр существующего ландшафта систем и частичная или полная модернизация).

Сами действия по модификации и совершенствованию системы могут представлять собой: трансформацию исходного кода вплоть до смены среды или языка программирования, совершенствование архитектуры приложения и принципов взаимодействия его модулей на основе анализа качества кода. Чаще всего выделяются следующие варианты проведения модернизации.

**Миграция (*migration*).** Переход на более новую версию платформы, ОС, СУБД или языка программирования. Обеспечивается достаточно эффективный с точки зрения соотношения стоимости и качества способ трансформации устаревающих систем.

### Пример

В процессе миграции могут изменяться:

- платформа: миграция с IBM Mainframe COBOL/CICS/DB2/VSAM на Intel-платформу Windows Micro Focus COBOL (с БД Oracle);
- язык программирования: конвертация кода на C#;
- пользовательский интерфейс: переход от desktop-версии к веб-интерфейсу;
- аппаратное обеспечение: переход с DEC VAX-обеспечения на Intel-серверы и рабочие станции (что, в свою очередь, требует перевода ПО с VAX OpenVMS на MS Windows);
- миграция баз данных: переход с IDMS на Oracle.

**Реинжиниринг (*re-engineering*).** Чаще всего применяется при адаптации сервисно-ориентированной архитектуры SOA, при построении прежних приложений на основе новых технологий и платформ, с прежней или же расширенной функциональностью.

**Рефакторинг кода.** Преобразования/реорганизация кода. Рефакторинг может быть обоснован необходимостью реализации но-

вой функции, которая не соответствует текущему архитектурному решению. Однако в особенности подобные активности необходимы, когда логика программы слишком сложна для понимания и требует структуризации и совершенствования в целом.

### Пример

---

Дублирование кода, слишком длинный список параметров метода или код самого метода, «мертвый код», лишние переменные, несгруппированные данные и пр.

Для устранения таких недостатков кода используются различные методы, например, инкапсуляция полей, делегирование, замена операторов полиморфизмом и пр.

---

Следует отметить, что несмотря на то, что рефакторинг относится к одному из вариантов модернизации, он должен проводиться на протяжении всего цикла разработки (к примеру, именно такая логика лежит в основе концепции экстремального программирования и ряда других методологий).

**Смена хостинга (*re-hosting*).** Чаще всего проводится в качестве промежуточного шага при предстоящей замене аппаратного обеспечения (например, на UNIX/Wintel-платформах).

**Внедрение «коробочного» решения (*package implementation*).** Замена устаревшего ПО целиком или частично на целые семейства решений (SAP, Oracle Apps).

**Виртуализация как стратегия модернизации решений.** Одним из возможных решений при модернизации ИС является виртуализация.

---

**Виртуализация** — предоставление вычислительных ресурсов, абстрагированное от аппаратной реализации и изолирующее вычислительные процессы конкретных физических ресурсов. Таким образом, при виртуализации создаются различные уровни абстракции.

---

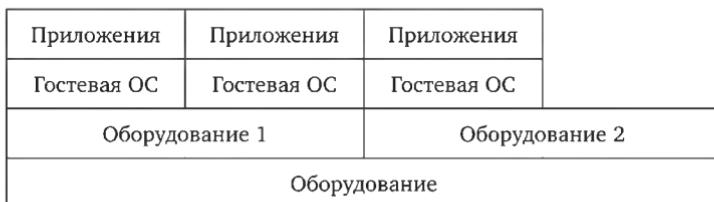
Сама концепция виртуализации появилась еще в 1970-х гг., когда IBM виртуализировала интерфейсы своего оборудования. VMS запускается непосредственно на основном оборудовании, позволяющем создавать множество виртуальных машин (VM), каждая из которых может обладать своей собственной операционной системой. Другим вариантом виртуализации в то время была симуляция процессора (выполнение псевдокода на виртуальной машине вместо реального оборудования).

В настоящее время виртуализация не теряет своей актуальности и становится все более и более популярным решением в силу следующих причин:

- происходит оптимизация расходов на ИТ;

- увеличивается степень контроля над ИТ-инфраструктурой;
- сокращаются плановые и внеплановые простой.

Среди вычислительных ресурсов, для которых доступна виртуализация, операционные системы и сети передачи данных, программно-аппаратные платформы, серверы/системы хранения данных. Один из наиболее сложных видов виртуализации обеспечивается эмуляцией аппаратных средств (когда виртуальные машины аппаратных средств создаются на хост-системе, чтобы эмулировать интересующее оборудование) (рис. 3.21).



*Рис. 3.21. Эмуляция аппаратных средств*

**Виртуализация приложений** обеспечивает изоляцию и безопасность ресурсов, которые предоставляются приложениям контейнерами. Каждый контейнер обслуживает содержащиеся в нем приложения при определенном уровне рабочей нагрузки. В некотором роде каждый контейнер имеет сходство с физическим сервером, однако виртуализация происходит на уровне приложений, а не на аппаратном.

При этом создается возможность выполнять приложение в виртуальном пространстве, которым можно управлять как отдельной единицей. Приложения могут выполняться в потоковом режиме на сервере приложений или на клиентском устройстве. Поэтому виртуализация отлично подходит для компаний, обладающих широким спектром приложений с относительно небольшим потреблением ресурсов центра данных, но используемых большим количеством сотрудников.

**Виртуализация серверов** маскирует от рядовых пользователей детали использования ресурсов (количество и основные данные серверов, процессоров, операционных систем), с которыми ведется работа. Она призвана обеспечить централизованное управление всеми серверными мощностями с максимальной гибкостью и высокой степенью масштабирования, а также быстрое развертывание виртуальных машин из готовых шаблонов, оперативное восстановление работоспособности серверов, мониторинг текущей загрузки физических и виртуальных серверов.

Одновременно с этим снижается зависимость эффективности сервиса от старения оборудования, так как появляется возможность

осуществлять модернизацию практически без прерывания сервиса, что критично для крупных территориально распределенных компаний.

Крайне ярким примером виртуализации серверов является услуга *VPS-хостинга* (*виртуальный выделенный сервер — virtual private server, VPS*). В подобных случаях в сети могут создаваться несколько независимых друг от друга виртуальных серверов с собственным набором служб и уникальными характеристиками, которые должны существовать как независимые узлы сети.

*Виртуализация систем хранения данных* с точки зрения пользователя переносит данные с разрозненных сетевых устройств хранения на единое, управляемое централизованно.

Основной задачей в данном случае является повышение эффективности и надежности, гибкости хранения данных и их мобильности. Дополнительное отделение систем хранения от серверного оборудования уровнем виртуализации позволяет прозрачно перемещать данные между системами хранения и упростить процессы обслуживания и поддержки.

*Виртуализация представлений* — предоставление вычислительных ресурсов терминальным сервером и выполнение всех операций клиентских приложений на нем. Несколько компьютеров могут быть представлены как один отдельный компьютер (*серверный кластер/ grid computing*). Пользователь при этом лишь видит собственное представление, что значительно снижает сложность администрирования, так как к подобной терминальной сессии всегда можно подключиться, избегая установки дополнительных программных средств. К тому же, несмотря на высокую стоимость подобных мощных серверов, итоговая стоимость проекта может оказаться более низкой, нежели при установке многочисленных локальных рабочих станций.

**Особенности проектов по модернизации.** Большая часть задач по модернизации решения может проводиться параллельно с эксплуатацией (не дожидаясь ее прекращения), однако это требует повышенного контроля всех параметров и данных системы, вероятность ошибок в которых резко повышается. С другой стороны, успешное применение подобного способа позволяет избежать необходимости миграции данных после модернизации и поиска «промежуточного» решения на время модификации ИС, заменяющего по функциональности и удобству основную систему.

Перед началом модернизации следует рассмотреть ряд аспектов.

**Целесообразность модернизации.** Иногда лучшим выбором может быть продолжение использования системы с фокусом на снижении затрат на поддержку. Однако любая *legacy*-система со временем будет требовать все больших вложений за счет необходимости ее сопровождения сотрудниками с необходимым набором компетенций и опытом работы с подобными системами.

**Аутсорсинг систем и процессов.** Все большее число компаний выбирает вариант рехостинга *legacy*-систем (особенно в условиях наличия многочисленных вариантов планов аутсорсинга, предлагаемых поставщиками). При этом модификации самой системы можно избежать.

**Целесообразность перехода на другую платформу.** Для компаний, активно участвующих в процессах слияний и поглощений (M&A), эффективной альтернативой модернизации может стать консолидация и перевод *legacy*-систем на единую корпоративную платформу (в идеале — с хорошей масштабируемостью). Подобные действия могут значительно повысить затраты в краткосрочном периоде, но долгосрочные выгоды оказываются очень существенными.

**Минимизация числа изменений.** В некоторых случаях предприятия могут вносить изменения во внутренний дизайн приложения, сохранив его функциональные возможности. Так, рефакторинг кода программного решения иногда позволяет модифицировать приложение гораздо более эффективно, нежели масштабная модернизация.

**Минимальная кастомизация.** Многие вендоры положительно относятся к проектам по кастомизации/локализации их программных решений, когда это означает более высокую выручку и выход на новые рынки. Так, в большинстве случаев компании приобретают бизнес-платформы и адаптируют исходный код под собственную специфику (либо выпускают патч для более старой версии, например, с учетом изменений налогового законодательства).

Однако в тот самый момент, когда предприятие примет решение о выпуске новых релизов, ему придется столкнуться с множеством кастомизированных доработок и отклонений от исходной версии, что увеличит затраты труда и времени на модификацию.

**Длительные проекты внедрения значительно повышают риски.** В современных условиях, когда жизненный цикл программных решений постоянно сокращается, а новые технологии и стандарты появляются с видной регулярностью, система, разработанная в сложном длительном проекте внедрения с многочисленными итерациями тестирования и доработок, может оказаться неактуальной еще до окончания проекта.

**Интеграция систем на основе SOA не является панацеей от всех проблем.** Все чаще вендоры рекомендуют настройку SOA-интерфейсов *legacy*-систем, когда «монолитные» приложения разбиваются на определенные компоненты, которые затем по отдельности реализуются по SOA-принципам. АдAPTERЫ инкапсулируют различные технологии для реализации интерфейса между приложениями, а для подключения используется единая сервисная шина.

Однако следует отметить, что данный подход никоим образом не снижает сложность приложения и затраты на его сопровождение,

а лишь обеспечивает его более эффективное взаимодействие с окружающей средой.

**Возможности обновлений.** Необходимо взвешенно принимать решение об обновлении ПО: обновлять его при выпуске каждой новой версии, либо же с определенными промежутками, только после выпуска самых значительных обновлений. Компании, заключающие контракт на сопровождение, чаще всего ограничиваются обновлением приложений до новой версии в случае необходимости. Однако возможна ситуация, когда приобретенное приложение больше не выпускается и не поддерживается и организациям приходится сталкиваться с масштабной заменой программного решения.

**Планирование архитектуры приложений.** Замена системы может решить только небольшую часть проблем, важно в целом эффективно планировать ландшафт приложений, формировать принципы принятия решения о внедрении, модернизации и утилизации систем, согласовывать действия со стратегическими целями ИТ и предприятия в целом.

### 3.7. Утилизация

Не существует вечных систем, а значит, необходимо предусмотреть условия, при которых понадобится отказаться от системы и вывести ее из эксплуатации. Эта деятельность, в свою очередь, также является проектом, ключевая цель которого — снятие системы (или ее отдельных модулей) с эксплуатации, что чаще всего проводится уже после внедрения новой системы. Рассмотрим классификацию необходимых в рамках данного проекта действий. Обратите внимание, что деятельность по выбору альтернативной ИС и принятие решения по замене не входят в эту фазу. Фаза утилизации предусматривает только практические действия по выводу ИС из эксплуатации, так как при формировании ТСО затраты на анализ альтернатив и выбор новой системы будут отнесены на новую систему. Допустим, на предприятии принято решение о внедрении системы CRM и выводе из эксплуатации «самописной» программы, которая поддерживала учет выставленных коммерческих предложений заказчикам. Функциональность новой системы CRM будет шире. Конечно, бухгалтерия отнесет затраты по обследованию, постановке задачи, процессу выбора ПО, доработке и внедрению на новый программный продукт.

#### Технические аспекты

**Аппаратное обеспечение.** Существующую платформу (компьютерное и серверное оборудование, сетевые и телекоммуникационные устройства, параметры сети и безопасности, например, выделенные IP-адреса и настройки сетевого экрана) можно использовать

для развертывания других систем (например, изменив некоторые параметры или конфигурацию), продать, расторгнуть контракт лизинга.

**Программное обеспечение.** Основные программное решение, компиляторы и среды разработки, коннекторы, внутренние библиотеки и прочие компоненты ПО, относящиеся к системе, можно сохранить в качестве резервной копии.

**Данные.** Для данных (как необходимых для работы с системой, так и созданных в процессе ее эксплуатации) необходимо создать резервные копии как в «родном» формате системы, так и в читаемых на других платформах форматах (pdf, xml, ...).

**Документация.** Бизнес- и техническая документация системы (в виде детального описания алгоритмов и бизнес-правил, заложенных в систему) также должны быть сохранены в резервных копиях внутреннего формата системы и общедоступных (pdf, docx, ...).

**Замещающие системы.** К моменту вывода системы из эксплуатации должны быть предусмотрены все средства плавного перехода к эксплуатации другого решения (включая аспекты управления изменениями).

### Пример

При смене системы «Учет кадров компании “Парус”», которая ранее эксплуатировалась в компании, на систему 1С решение о снятии с эксплуатации первой системы (utiлизации этой подсистемы) принимается только после полной миграции данных, одновременного сопровождения двух систем и полномасштабного ввода соответствующего модуля 1С в эксплуатацию.

**Зависимые/интегрированные системы.** Этот пункт предполагает два основных аспекта. Первый — вопрос прекращения передачи и получения данных в интегрированные системы и из них, а также необходимость предусмотреть другие источники данных для них. Особенно это касается внешних приложений и интерфейсов работы с системами контрагентов. Второй — вопрос определения стратегии действий для вспомогательных систем (БД, средства отчетности, приложения автоматизации бизнес-процессов и пр.), необходимость в которых со снятием системы с эксплуатации может исчезнуть или потерять актуальность.

### Организационные аспекты

**Сотрудники ИТ-департамента компании.** К моменту прекращения необходимости поддержки эксплуатации и модернизации системы ИТ-специалистами необходимо предусмотреть их занятость в других проектах с учетом оптимального применения приобретенных ими за время проекта знаний и умений

*Подрядчики и специалисты, нанятые по контракту.* Внешние специалисты, на постоянной или временной основе занятые в проекте внедрения и поддержки системы, могут остаться в компании (продлив контракт или перейдя в штат) или покинуть ее по окончании контракта. Соответственно, в случае необходимости использовать их компетенции на других проектах необходимо своевременно предусмотреть им замену и организовать передачу критически важных знаний новым сотрудникам.

*Конечные пользователи.* Если на смену утилизируемому ПО внедряется альтернативное решение, пользователи перед этим в обязательном порядке должны пройти необходимое обучение.

### Коммерческие аспекты

*Контрактные вопросы.* В процессе эксплуатации системы могли заключаться договоры с подрядчиками по поддержке, сервисами веб-хостинга, дата-центрами и прочими организациями, а значит, каждый из контрагентов должен быть своевременно оповещен о планируемых изменениях и принято решение о продолжении или прекращении совместной работы.

### Юридические аспекты

*Лицензирование.* Закупленные лицензии на ПО могут быть деактивированы либо «заморожены» (в этом случае право на владение лицензиями сохраняется, но право пользования ими временно утрачивается до заключения дальнейших соглашений с поставщиком (например, при обмене лицензий на другую версию или другой программный продукт)).

*Отчетность.* Любые обязательства компаний по раскрытию информации, ранее осуществлявшиеся при помощи выводимой из эксплуатации системы, должны быть предусмотрены в заменяющей ИС (включая передачу исторических данных).

### Пример

В качестве еще одного напоминания, каких ситуаций следует избегать, приведем следующий пример.

В одной из организаций используемое решение (по учету транспорта) было разработано 5—10 лет назад, и главные разработчики давно перешли на работу в другую компанию. Тем временем, по мере роста масштабов бизнеса, понадобилось внедрение приложения с расширенной функциональностью, отвечающего новым потребностям бизнеса. Однако миграция данных оказалась невозможной в силу отсутствия открытой для пользователя функции экспорта и недоступности аккаунта администратора системы, как и документации, включая исходный код со спецификациями, единственными версии которых были только у разработчиков системы.

В качестве общих замечаний к жизненному циклу ИС можно добавить, что цена ошибки экспоненциально увеличивается на протяжении этого времени. Из этого следует, что в течение всего жизненного цикла необходимо применять специальные методы поиска ошибок, почти всегда предусматриваемые различными методологиями внедрения систем.

## **Контрольные вопросы и задания**

1. Перечислите виды действий, которые требуется совершить на подготовительном этапе.
2. Что содержат в себе такие документы, как отчет об экспресс-обследовании, технико-экономическое обоснование и оценка целесообразности проекта?
3. Какая деятельность происходит на стадии анализа и постановки задачи?
4. Что понимается под информационным обследованием предприятия?
5. При помощи каких нотаций и программных продуктов осуществляется моделирование бизнес-процессов?
6. На основании каких стандартов производится классификация требований к ИС?
7. Какие аспекты включает в себя фаза проектирования ИС?
8. Какие CASE-средства используются для проектирования?
9. Какие программные продукты используются для поддержки UML?
10. Какие представления архитектуры ИС существуют в модели «4+1»?
11. Какие диаграммы UML и с какими целями применяются?
12. В каком порядке вы бы стали создавать UML-диаграммы?
13. Что происходит на стадии разработки информационной системы?
14. Какие действия совершаются при настройке конфигурации, создании ролей пользователей, миграции данных и разработке контрольного примера?
15. Какие цели преследует проведение тестовой эксплуатации?
16. Как осуществляется развертывание и внедрение информационной системы?
17. Почему особую важность приобретает обучение пользователей?
18. Какие основные виды тестирований существуют?
19. Как производятся приемно-сдаточные испытания информационной системы?
20. В чем заключается важность фазы эксплуатации ИС?
21. Какие виды сопровождения эксплуатации существуют и чем они различаются между собой?
22. Зачем проводится модернизация информационной системы?
23. Какие существуют стратегии управления legacy-системами?
24. На чем основывается концепция виртуализации и как она применяется на фазе модернизации ИС?
25. Какие аспекты фазы утилизации вы отметите? Какими причинами вызвана потребность в данной фазе?

## **Рекомендуемая литература**

1. *Васильев, Р. Б. Стратегическое управление информационными системами / Р. Б. Васильев [и др.]. — Москва : Интуит, 2010.*
2. *Васильев, Р. Б. Управление развитием информационных систем : учебное пособие для вузов / Р. Б. Васильев, Г. Н. Калянов, Г. А. Левочкина ; под редакцией Г. Н. Калянова. — 2-е изд. — Москва : Горячая линия-Телеком, 2014.*
3. *Грекул, В. И. Проектирование информационных систем // Национальный открытый университет «ИНТУИТ», 2012. — URL: <http://publications.hse.ru/books/74833061> (дата обращения: 17.10.2020).*
4. *Зараменских, Е. П. Управление жизненным циклом информационных систем : монография / Е. П. Зараменских. — Новосибирск : ЦРНС, 2014. — С. 270.*
5. *Каменнова, М. С. Моделирование бизнеса. Методология ARIS / М. С. Каменнова [и др.]. — Москва : Весть-МетаТехнология, 2001. — С. 36—115.*
6. *Лодон, Дж. Управление информационными системами : перевод с английского / Дж. Лодон, К. Лодон. — 7-е изд. — Санкт-Петербург : Питер, 2005.*

# **Тема 4**

## **АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ**

---

Эта тема подробно рассматривает этап анализа и постановки задачи и его место в жизненном цикле информационной системы. Подробно рассматривается подход, получивший название структурного анализа. Тема включает описание основных нотаций, которые применяются в рамках структурного подхода. Отдельное внимание уделяется моделированию бизнес-процессов и соответствующим нотациям.

После изучения данной темы студент будет:

**знать**

- основы структурного анализа;
- основы функционального анализа и проектирования;
- методологию SADT;
- стандарт и графическую нотацию IDEF0;
- стандарт и графическую нотацию IDEF3;
- методологию и графическую нотацию DFD;
- стандарт и графическую нотацию IDEF1X;
- диаграммы «сущность-связь» (ERD) и варианты их графической нотации;
- место бизнес-процессов в анализе и проектировании ИС;
- основные нотации моделирования бизнес-процессов (BPMN, eEPC);
- основные программные продукты моделирования деятельности организации;

**уметь**

- производить функциональный анализ и проектирование;
- формировать информационную модель;
- описывать бизнес-процессы;

**владеть навыками**

- моделирования в графической нотации IDEF0, IDEF3, IDEF1X;
  - моделирования в графической нотации DFD;
  - моделирования диаграмм «сущность-связь» (ERD);
  - моделирования бизнес-процессов в нотации BPMN, eEPC;
  - использования основных программных продуктов моделирования.
- 

### **4.1. Основы структурного анализа**

Элементы современного структурного анализа начали использоваться уже при создании ранних программ для ЭВМ. Сложность

программ росла стремительно, и в языках программирования стали применяться дополнительные конструкции (функции, модули, процедуры), облегчающие разработку. Программа, исходно сводившаяся к простому перечислению исполняемых команд, по мере развития подобных конструкций переставала быть неотделимой от аппаратной части. Иными словами, программы стали структурными и иерархичными, поскольку в их основе лежала структурная модель.

---

**Структурная модель** — это способ описания объектов как сущностей и атрибутов, но без описания их свойств<sup>1</sup>.

---

*Принцип декомпозиции* — это основа структурного подхода, поскольку системная структура описывается в виде иерархии функций и передачи информации между функциональными элементами. Принцип декомпозиции означает, что ИС разбивается на подсистемы по признаку функциональности. Подсистемы делятся на функции, которые в дальнейшем делятся на задачи и т. п. Процесс декомпозиции повторяется до тех пор, пока не будут получены конкретные процедуры. Тем не менее ИС остается целостной, поскольку декомпозиция не делает подсистемы несвязанными.

Базовые принципы структурного подхода таковы:

- принцип «разделяй и властвуй» (сложные проблемы решаются как более простые за счет разбиения на удобные для понимания проблемы);
  - принцип иерархичности;
  - принцип абстрагирования (выделение основных аспектов системы, чтобы не отвлекаться на несущественные);
  - принцип формализации (применение строго формализованной методологии);
  - принцип непротиворечивости (все элементы ИС должны быть обоснованными и согласованными);
  - принцип структурирования данных (все данные должны иметь четкую иерархическую структуру).

Первые методологии, нацеленные на применение принципов структурного программирования, стали развиваться и активно применяться на практике уже в 60-х гг. XX в. При этом структурный подход не менее актуален и сейчас.

В наиболее общем виде структурный подход к анализу основан на представлении ИС в виде системы взаимосвязанных функций. За счет этого ход работ начинается с общего описания системы и продолжается детализацией ее отдельных функций, причем число

---

<sup>1</sup> Вичугова А. В. Методы и средства концептуального проектирования информационных систем: сравнительный анализ структурного и объектно-ориентированного подходов // Прикладная информатика. 2014. № 1.

уровней варьируется вплоть до достижения уровня конкретных процедур, которые невозможно или не требуется детализировать. Вместе с этим констатируется, что ИС должна быть представлена как единое целое, а ее компоненты должны оставаться взаимосвязанными.

Сегодня существует множество методологий, использующихся в структурном анализе. Одна из самых известных — методология SADT (*Structured Analysis and Design Technique*), которая возникла еще в 60-е гг. XX в. Методология позволила упорядочить процесс создания ИС, что стало своеобразным прорывом для отрасли. Уже в 1981 г. SADT применяли около 50 крупнейших компаний, а ее успешное применение было продемонстрировано более чем в 200 проектах.

SADT применяется и сегодня, причем для ее поддержки используются различные современные программные решения. Тем не менее гораздо чаще можно встретиться с методологией IDEF0, основанной на идеях и принципах SADT. Стандарт IDEF0 был разработан в 1981 г. департаментом Военно-воздушных сил США для обеспечения групповой работы над созданием различных моделей. Предполагалось, что стандарт позволит участвовать в работе всем разнопрофильным аналитикам и специалистам. Последняя редакция IDEF0 датируется 1993 г., а сама методология по-прежнему применяется для функционального моделирования.

Следующая важная методология структурного подхода носит название DFD (*Data Flow Diagram*). DFD позволяет описать внешние (относительно ИС) источники данных, адресатов, логические функции, хранилища данных. DFD применяется для функционального анализа наравне с IDEF0.

Для информационного моделирования в рамках структурного подхода зачастую применяется методология ERD (*Entity-Relationship Diagram*). Как правило, ER-модели применяются для высокоуровневого концептуального моделирования баз данных. Команда разработчиков формирует исходную ER-модель, которая затем трансформируется в конкретную схему БД. Методология была предложена в 1976 г. П. Ченом.

Функциональное и поведенческое моделирование очень часто производится при помощи EPC-диаграмм (*Event-driven Process Chain*). Диаграммы данного типа широко применяются в бизнесе для описания бизнес-процессов. Метод был разработан в начале 1990-х гг. А.-В. Шеером в рамках работы над созданием ARIS.

Не меньшее значение для функционального и поведенческого моделирования имеют BPMN-диаграммы. BPMN также нацелена на описание бизнес-процессов и широко применяется как в бизнесе, так и в сфере проектирования информационных систем.

Теперь, перечислив основные методологии и диаграммы структурного анализа, можно перейти к рассмотрению его общей логики.

ки. Общая логика структурного анализа подразумевает последовательное выполнение четырех шагов.

1. Разработка функциональной модели, которая определяет, анализирует и фиксирует требования к составу и структуре ИС. Функциональная модель включает определение, для чего разрабатывается ИС и какие функции она будет выполнять. Также функциональная модель включает промежуточные и итоговые результаты работы системы и исходную информацию.

2. Разработка информационной модели, основанной на информационных потоках, которые определяют состав и структуру данных, хранимых и используемых системой.

3. Разработка поведенческой модели, описывающей поведение системы, т. е. формирование процедур непосредственно для реализации сформированных ранее функций. Поведенческие модели включают алгоритмы обработки данных и поведения элементов системы.

4. Разработка компонентной модели. Заключительная стадия включает в себя распределение функций по подсистемам, техническое обеспечение и модель распределения по узлам системы.

Разумеется, модели могут создаваться параллельно или в любом другом порядке. Практика показывает, что разработка каждой последующей модели начинается еще до середины разработки текущей модели, а иногда и раньше.

## 4.2. Функциональный анализ и проектирование

*Функциональный анализ и проектирование* — первое, с чем сталкивается проектная команда на этапе анализа и постановки задачи. Очевидно, что перед началом разработки системы обе стороны — как заказчик, так и разработчик — должны иметь полное, подробное и, по возможности, исчерпывающее представление о функциональных возможностях разрабатываемой ИС. Не менее важно понимать, как будет устроено взаимодействие функциональных компонентов внутри информационной системы.

Потребность в функциональном анализе и проектировании возникает из-за наличия ярко выраженных специфических проблем, которые встречаются достаточно часто, едва ли не в каждом проекте. Наибольшее число проблем возникает, когда сторона заказчика представлена небольшим числом ИТ-специалистов или когда их подготовка не слишком хороша. В целом же можно выделить четыре главные проблемы.

1. Заказчик не представляет, что конкретно должна делать создаваемая система. Существует только очень общее представление о функциональности системы, однако детализация даже до уровня

задач затруднена. При этом нередко заказчик имеет слабое представление о том, что такое требования и каким образом их нужно сформулировать.

2. Специалисты, представляющие сторону заказчика, имеют разное представление о функциональности создаваемой ИС. Как правило, такая проблема возникает, когда группа представителей заказчика разнородна и в нее входят ИТ-специалисты, рядовые пользователи, топ-менеджеры, управленцы среднего звена и т. п. Естественно, что каждый из них видит функциональность ИС со своей точки зрения. Нередко требования представителей оказываются взаимоисключающими. А если имеется несколько объектов автоматизации, проблема приобретает еще более серьезный масштаб.

3. Зачастую сторона заказчика не имеет адекватного представления о возможностях современных ИС. Часто можно столкнуться с заблуждением, будто ИС необходима только для автоматизации наиболее простых (и даже примитивных) функций. Однако задачи ИС не сводятся к автоматизации рутинных видов деятельности. Поэтому для получения наиболее значимого результата требуется изменение бизнес-процессов, о чем будет рассказано ниже.

4. Четвертая проблема стала носить ярко выраженный характер, когда ИТ стали по-настоящему интеллектуальными. Сегодня информационные системы могут решать задачи, которые требуют выполнения интеллектуальных операций и видов деятельности. Тем не менее многие заказчики фактически отказываются поверить, что технология способна заменить человека в ряде областей.

Именно для решения данных проблем применяется функциональный анализ и проектирование. Как неоднократно показывала практика, построение функциональной модели позволяет решить эти и множество других проблем.

#### **4.2.1. Основы функционального анализа и проектирования**

Наиболее распространена форма функционального анализа, в которой изначально строится модель существующего в настоящий момент предприятия (модель «как есть», *As-Is*). Для построения модели «как есть» анализируются должностные инструкции, приказы, нормативные документы организации, отчеты, реальная практика выполнения рабочих операций и т. п.

В наиболее общем виде создание модели «как есть» позволяет ответить на вопрос: каким образом организация работает сегодня? Анализируя данную модель, команда разработчиков совместно с заказчиком пытаются выявить слабые места организации, которые мешают эффективной работе.

Именно на данном этапе функционального анализа и проектирования выявляются неэффективные места в деятельности организаций. Среди них фигурируют:

- виды деятельности, не приносящие дохода организации;
- дублирующиеся виды деятельности;
- неуправляемые работы;
- неэффективный документооборот и обмен информацией и т. п.

После выявления и формализации найденных недостатков разработчики совместно с представителями заказчика могут приступить к разработке модели «как должно быть» (To-Be). При помощи модели «как должно быть» формулируются и выявляются альтернативные варианты функционирования, которые могут помочь компании стать более эффективной.

Важно отметить, что модель To-Be не должна быть идеализированной. По ряду объективных причин деятельность компании не может быть полностью рациональной в научном понимании данного термина, поэтому в приоритете должно быть построение действительно жизнеспособной модели. Как правило, модель получается идеализированной в том случае, если ее формирование происходит на основе информации, полученной из официальных документов и (или) со слов руководителя. Реальная деятельность обычно существенно отличается от этого. Если не принимать этого во внимание, то результатом работы станет приукрашенная и искаженная модель, не имеющая никакого отношения к реальности.

Также стоит понимать, что результаты применения новой информационной системы во многом зависят от построенных моделей. Так, если остановиться на модели As-Is и начать ее автоматизацию, то заказчик практически не увидит пользы: работа компании не изменится, а автоматизация даже большого числа функций не приведет к серьезным изменениям. В ряде случаев организация может попросту понести дополнительные издержки, обусловленные затратами на разработку и закупку оборудования.

Попытка взять за основу идеализированную модель To-Be также не приведет к положительным переменам в деятельности компании-заказчика, хотя нередко кажется, что это хороший путь. На деле же организация может столкнуться с тем, что работа практически остановится: люди не смогут работать с ИС, которая не учитывает реальные условия их деятельности. Соответственно, оптимальный вариант — разрабатывать ИС на основе реалистичной модели To-Be.

#### **4.2.2. Методология SADT**

Под методологией SADT понимается набор методов, процедур и правил, позволяющих построить функциональную модель объекта в любой предметной области. Иными словами, при помощи SADT можно описать все действия объектов и связи между ними.

**История SADT.** SADT как методологию начал разрабатывать в 1960-е гг. в США Дуглас Росс. Появление и развитие SADT стало одним из первых следствий возникновения структурного програм-

мирования и, соответственно, структурного подхода. По мере того как бизнес и ИТ-специалисты сталкивались с потребностью в создании крупномасштабных систем, все более очевидным становился дефицит структуризации работ над каждым проектом. При помощи SADT процесс создания ИС был формализован за счет выделения следующих фаз:

- анализ;
- проектирование;
- реализация;
- тестирование;
- установка;
- эксплуатация.

Первый крупный проект, продемонстрировавшей эффективность структурного подхода, был реализован в 1973 г. компанией SofTech. Проект касался аэрокосмической отрасли, которая нуждалась в масштабных и сложных проектах, а потому успех SADT дал «зеленый свет» массовому применению методологии.

Компания SofTech и Дуглас Росс продолжили работу над SADT, и уже в 1974 г. методология была передана крупной европейской телефонной компании. А год спустя SADT приобрел вид конкретного продукта и вышел на рынок. В течение последующих пяти лет методология SADT была успешно применена для реализации более чем 200 проектов, над которыми работали порядка 2000 специалистов. Среди областей применения SADT фигурировали аэрокосмическая отрасль, телекоммуникации, управление и контроль, учет, обработка данных и т. п.

**Реализация SADT.** Перечисленные выше фазы на практике реализовались на основе четко прописанной формальной процедуры. Основой реализации были *диаграммы и титульные листы*.

Существенным шагом вперед для SADT стало появление мощных компьютеров, которые позволяли с адекватной скоростью создавать графические изображения. Методология SADT оказалась настолько эффективной, что BBC США приняли решение профинансировать разработку системы автоматизации SADT, которая стала первым автоматизированным средством, применяющимся в рамках структурного подхода. Система автоматизации получила название AUTOIDEFO.

SADT была реализована на мини- и микрокомпьютерах. Современные средства автоматизации SADT обладают полным покрытием данной методологии.

**SADT сегодня.** SADT как методология в чистом виде сегодня практически не применяется. Тем не менее широкое распространение сегодня получило семейство методологий IDEF, которые рассматриваются как самостоятельные начиная с 1981 г. Последняя редакция стандарта IDEF датируется 1993 г. Версия была выпуще-

на Национальным институтом по стандартам и технологиям США (NIST). Методологии IDEF используются преимущественно в качестве нотаций для моделирования без привязки к фазам, которые обозначены в методологии SADT.

#### 4.2.3. Семейство методологий IDEF

*IDEF (Icam (Integrated computer-aided manufacturing) DEFinition for functional modeling)* представляет собой семейство стандартов описания и отображения бизнес-процессов:

- IDEF0, отображающая процесс на уровне функций;
- IDEF1, фокусирующая на информационных потоках;
- IDEF1X для разработки реляционных баз данных;
- IDEF3, моделирующая технологические процессы как следующий уровень после IDEF0;
- прочие (реже применяющиеся методологии IDEF):
  - IDEF2 для динамического моделирования систем,
  - IDEF4 для построения объектно-ориентированных систем,
  - IDEF5 для онтологического исследования сложных систем,
  - IDEF6, акцентирующую внимание на процессе создания модели (обстоятельствах и причинах выбора того или иного метода моделирования),
  - IDEF7 для аудита информационных систем,
  - IDEF8 для разработки пользовательских интерфейсов,
  - IDEF9 для определения бизнес-ограничений при сценарном проектировании информационных систем,
  - IDEF10—IDEF14 — методы в области проектирования компьютерных сетей, архитектуры внедрения и прочих предметных областей, которые были определены как необходимые и востребованные, но разработка которых не была завершена.

Рассмотрим наиболее распространенные диаграммы IDEF0 и IDEF3 подробнее. Диаграммы IDEF1X будут рассмотрены в подл. 4.3.3.

IDEFO. IDEF0 представляет процесс в виде иерархической структуры функциональных блоков. Можно считать, что IDEF0 рассматривается в качестве главного стандарта, тогда как другие диаграммы данного семейства служат для его дополнения и детализации.

Стандарт IDEF0 включает четыре типа диаграмм.

1. Контекстные диаграммы, которые иллюстрируют назначение системы и ее связь с внешней средой.
2. Диаграммы декомпозиции, которые описывают фрагменты системы в виде подфункций.
3. Диаграммы дерева узлов, которые иллюстрируют иерархическую зависимость функций (но не связь между ними).
4. Диаграммы только для экспозиции, которые иллюстрируют фрагменты модели или альтернативные точки зрения.

**Графическая нотация IDEF0.** В виде прямоугольников представляются работы (функции), каждая из которых имеет фиксированную цель и приводит к какому-либо ожидаемому результату (рис. 4.1—4.2). Взаимодействие работ друг с другом или с внешним миром описывается при помощи стрелок пяти различных типов.



Рис. 4.1. Элементы графической нотации IDEF0

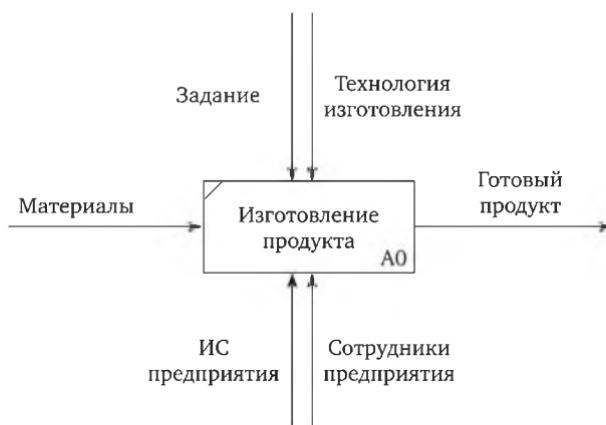


Рис. 4.2. Пример контекстной диаграммы IDEF0

1. **Вход (Input)** — материал или информация, используемые Функцией для получения Выхода. В общем виде Вход отвечает на вопрос: что будет обработано функцией? В качестве Входов могут фигурировать как материальные объекты (сырье, материалы, инструменты), так и нематериальные сущности (информация, запрос пользователя). Допустимо, если Функция не имеет входов.

2. **Управление (Control)** — управляющие и регулирующие данные, которыми руководствуется функция. Управление отвечает на вопрос: чем руководствуется функция? Нередко Управление рассматривается как ограничение, которое накладывается на Функцию. К Управлению относят нормативные акты, устные указания руководства, правила, процедуры и т. п.

3. **Выход (Output)** — результат выполнения Функции. Выход отвечает на вопрос: что является результатом выполнения Функции? В качестве выходов также могут фигурировать материальные и нематериальные объекты.

4. **Механизм (Mechanism)** — это ресурсы, которые непосредственно реализуют Функцию. Механизм отвечает на вопрос: кто и посредством чего реализует функцию? К Механизмам относится персонал, производственное оборудование, ПО и т. п.

Обратите внимание, что число стрелок, отраженных на диаграмме, может быть различным. На приведенном ниже примере отражены две стрелки Управления и две стрелки Механизма.

Диаграмма декомпозиции. Контекстная диаграмма в дальнейшем детализируется при помощи диаграмм декомпозиции (рис. 4.3). Основная функция, отраженная в контекстной диаграмме, делится на составные подфункции, причем разбиение продолжается до требуемого уровня детализации системы. Как правило, после создания каждого уровня декомпозиции проводится экспертиза, в ходе которой эксперты по предметной области оценивают, насколько реально существующие процессы соответствуют созданным диаграммам декомпозиции.

Диаграммы IDEF0 используются не только для иллюстрации процессов предприятия, но и в целом для описания схемы его функционирования. Они показывают, с какими ресурсами организация работает, от чего зависит и какой итоговый результат производит. Однако в некоторой степени содержание функциональных блоков IDEF0 остается «черным ящиком», в связи с чем проводится построение дополнительных схем процессов, уже в нотации IDEF3.

IDEF3. IDEF3 предназначена для документирования процессов системы, последовательности их операций, детализирующих функции, описанные в IDEF0. Данный подход более структурирован, чем IDEF0, и определяет схемы последовательности процессов, их ветвления и слияния, что также полезно в случае использования модели для дальнейшей работы уже с методами имитационного анализа (рис. 4.4), где перекрестки J1 и J4 — асинхронный «И», перекрестки J2 и J3 — асинхронный «ИЛИ», 1.2—1.6 — активности.

IDEF3 оперирует понятиями «единиц работы», которые объединяются по принципу временного предшествования либо использования результатов одной единицы работы в качестве входных потоков для другой.

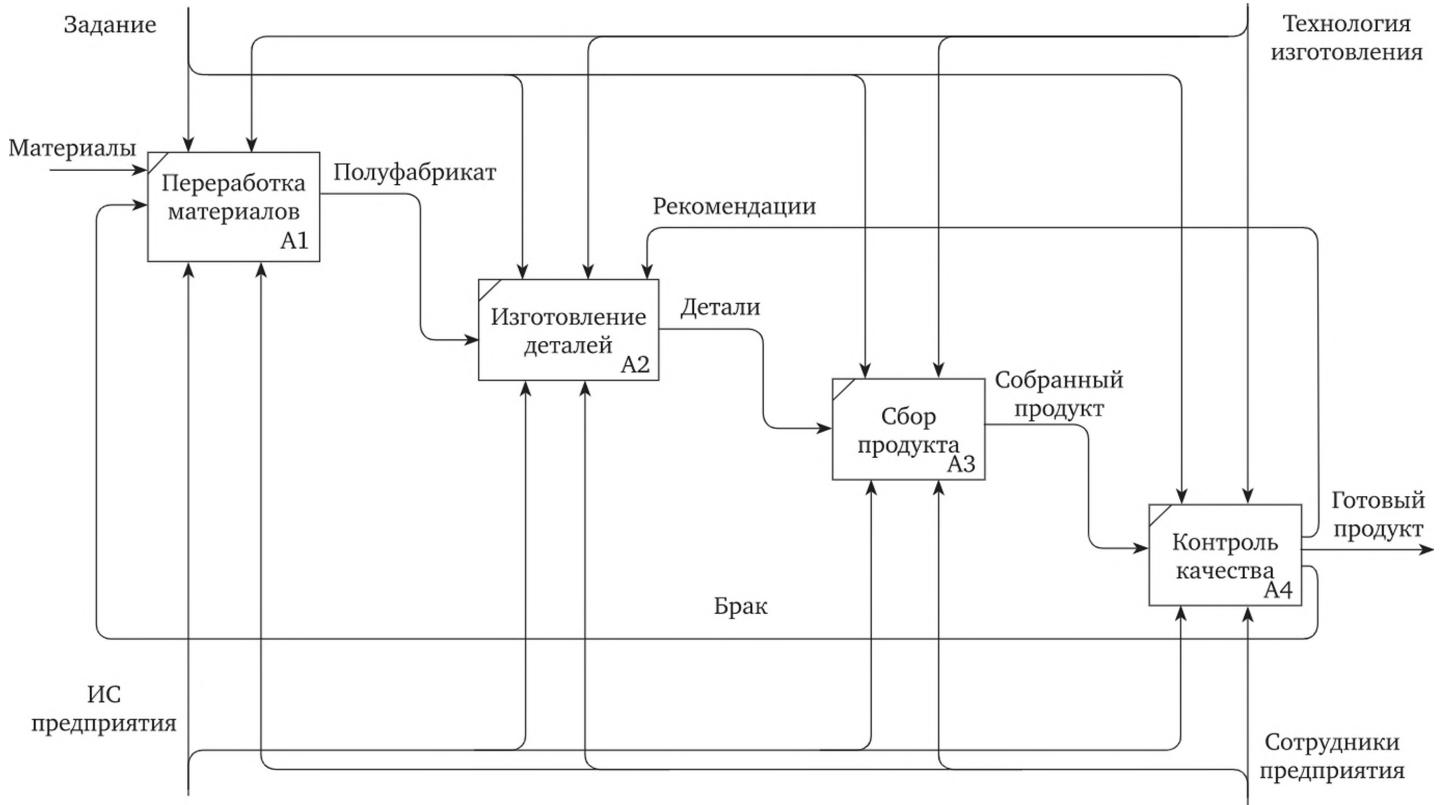


Рис. 4.3. Пример диаграммы декомпозиции IDEF0

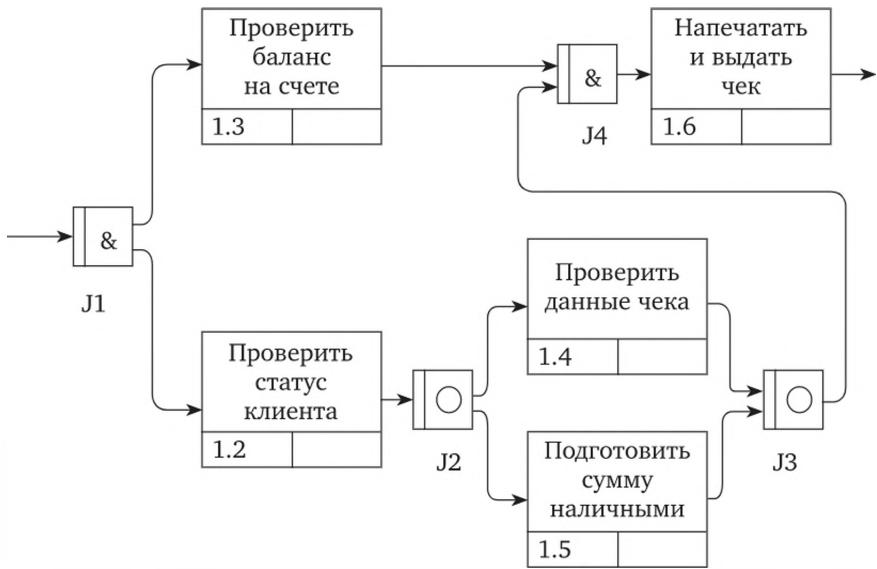


Рис. 4.4. Пример диаграммы IDEF3<sup>1</sup>

### Пример (функция «Анализ сведений о приемке товара»)

#### Текстовое описание

Старший кладовщик направляет упаковочный лист в отдел закупок. В отделе закупок анализируются сведения о приемке товара. Если фактическое количество прибывшего товара имеет отклонение в пределах нормы, производится расчет себестоимости товара. Составляется три вида документов: расчет себестоимости товара направляется на склад; расчет себестоимости с раскладкой себестоимости по поставщикам ТМЦ и услуг направляется в финансовый отдел; расчет себестоимости без раскладки по поставщикам направляется начальнику отдела продаж. Если есть значительные ошибки (нарушение сроков годности, недостача), то проводится анализ причин, выявляются виновные, предъявляются претензии.

Главная и наиболее очевидная задача диаграмм IDEF3 — представление причинно-следственных связей между ситуациями и событиями в понятной эксперту форме.

#### 4.2.4. Методология DFD

Диаграмма потоков данных (DFD) описывает весь процесс преобразования информации, начиная с ее ввода в систему и заканчивая предоставлением пользователю конечного результата. На верхнем уровне используются контекстные диаграммы, которые описывают

<sup>1</sup> Черемных С. В., Семенов И. О., Ручкин В. С. Моделирование и анализ систем. IDEF-технологии: практикум. URL: <http://vernников.ru/krisis/item/212-modelirovaniye-i-analiz-sistem-idef-tehnologii-praktikum.html> (дата обращения: 15.06.2020).

ключевые процессы и подсистемы ИС с внешними входами и выходами. Для их детализации используются диаграммы нижнего уровня, причем декомпозиция продолжается до уровня, дальнейшая детализация которого не имеет смысла.

Диаграмма потоков данных:

- показывает источники и получателей данных;
- идентифицирует процессы и данные, которые их связывают;
- идентифицирует хранилища данных.

Структура потоков данных и все определения компонентов хранятся в специальном глоссарии.

Диаграммы, выполненные в нотации DFD, сегодня чаще всего рассматриваются как альтернатива или дополнения к диаграммам IDEF0. Однако в случае небольших систем (а чаще — в случае небольшого ПО) диаграммы DFD могут использоваться самостоятельно.

**Элементы графической нотации DFD.** В современной практике используются два вида графической нотации DFD — нотация Йордана и нотация Гейна — Сарсона. Графические элементы приведены в табл. 4.1.

Таблица 4.1

#### Элементы графической нотации DFD

Название элемента	Нотация Йордана	Нотация Гейна — Сарсона
Поток данных	Имя →	Имя →
Процесс (система, подсистема)	Имя Номер	Номер Имя
Накопитель данных	Имя	Номер   Имя
Внешняя сущность	Имя	Имя

На практике чаще всего можно встретиться с диаграммами, выполненными в нотации Гейна — Сарсона.

**Описание элементов нотации DFD.** Приведенная выше таблица показывает, что в нотации DFD используются четыре типа элементов. Рассмотрим эти элементы подробнее.

1. **Поток данных** — это информация, которая передается от источника к приемнику посредством соединения. Поток данных в DFD — это аналог Входов и Выходов в IDEF0. На практике поток

данных — это информация, которая передается по кабелю между устройствами; письма, отправляемые по почте; информация, переносимая с помощью съемных носителей (карты памяти, CD-диски и т. п.).

**2. Процесс** — это преобразование входных потоков данных в выходные на основе заранее определенных алгоритмов (аналог Функции/Работы в IDEF0). Как правило, процессы обозначаются при помощи глаголов неопределенной формы (вычислить, проверить, рассчитать и т. п.). В ряде случаев процесс может рассматриваться с точки зрения подсистемы. В этом случае он будет обозначаться как, например, «Подсистема расчета стоимости» и т. п.

**3. Накопитель данных** — абстрактное устройство для хранения информации, причем информация может быть помещена в него или извлечена в любое время без указания конкретных способов. На практике накопитель данных всегда имеет физическую реализацию (оперативная память, жесткий диск, ящик в картотеке и т. п.).

**4. Внешняя сущность** — это материальный объект или физическое лицо, которое является приемником или источником информации (потребитель, клиент, сотрудник, руководитель, библиотека, склад и т. п.). Внешняя сущность — это аналог Управления или Механизма в IDEF0.

В нотации DFD каждый Процесс (и, соответственно, каждая Подсистема) должен иметь не менее одного входящего и не менее одного исходящего Потока данных (рис. 4.5). Сказанное верно и для объекта «Накопитель данных», однако в ряде случаев допустима ситуация, когда информация накапливается, но не используется. Такая ситуация встречается, если необходимо хранить бумажные версии договоров, лицензий и т. п.

Обратите внимание, что каждый Процесс, Накопитель данных и Внешняя сущность обозначены своим уникальным номером.

### 4.3. Формирование информационной модели

Функциональный анализ и проектирование, рассмотренные выше, являются основой создания информационной системы. Тем не менее применение функциональных методологий выпускает из поля зрения множество важных деталей. В наиболее общем виде можно констатировать, что функциональный анализ и проектирование позволяют ответить на вопросы: что должна делать система? и каким образом можно достичь желаемого результата?

Очевидно, что ответов на эти вопросы недостаточно для создания ИС. Помимо функциональной основы любая ИС работает с информацией. Функциональный анализ и проектирование позволяют понять, какие данные будут использоваться в системе, но подобное описание нуждается в дальнейшей детализации.

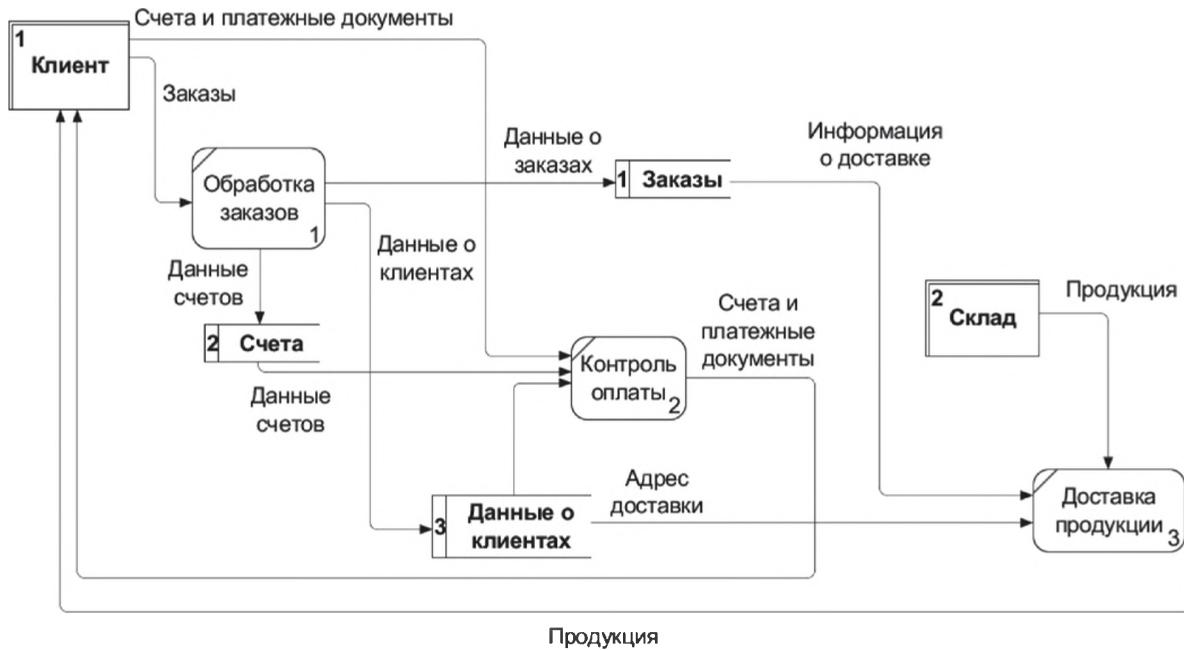


Рис. 4.5. Пример диаграммы DFD, выполненной в нотации Гейна — Сарсона (программное средство Ramus Educational)

С этой целью производится формирование информационной модели, способной ответить на вопрос: как данные организованы в проектируемой системе?

Классический подход к созданию информационной модели, применявшийся во многих проектах, насчитывает три этапа:

1) концептуальное проектирование, в рамках которого создается общая схема баз данных, включая основные сущности и связи между ними;

2) логическое проектирование подразумевает дальнейшее развитие концептуальной модели с той разницей, что на данном этапе в расчет принимается модель баз данных (иерархическая, сетевая, реляционная и т. п.);

3) физическое проектирование завершает цикл, добавляя к ранее созданным моделям целевую СУБД.

Совокупность концептуального и логического проектирования зачастую носит общее название **инфологического проектирования**.

#### 4.3.1. Методологии проектирования данных

Методологии данной группы применяются для быстрого и эффективного создания программно-информационного обеспечения информационной системы. *Проектирование ИС нередко основывается на данных*, поскольку данные — это самая стабильная составляющая компании, в отличие от ее структуры и функций, изменения в которых происходят, как правило, чаще. Подход, основанный на данных, оптимален, если создаваемая ИС нацелена на использование приложений и обеспечение их интеграции и согласованной работы.

Одной из наиболее распространенных в мире методологий является методология проектирования данных DATARUN (компания CSA, США). Методология DATARUN создавалась для проектирования и разработки ПО и ИС для переносимых распределенных ИС, работающих по архитектуре клиент-сервер. Методология использует современные инструменты и средства моделирования, быстрой разработки и прототипирования.

Основа DATARUN — это *системный подход*, рассматривающий деятельность организации через описание модели организации и модели ИС. Проектирование в данном подходе базируется на получении неких первичных данных из модели бизнес-процессов, а также на построении модели данных. Под первичными данными в DATARUN понимается стабильное подмножество данных, с которыми работает организация. К ним относятся продукты, услуги, транзакции, ресурсы, внешние и внутренние сущности (сотрудники, потребители, поставщики). Также к первичным относятся данные, которые получены в процессе принятия решений (графики, цены на продукты и пр.).

DATARUN использует принцип, согласно которому дальнейшая модель данных строится на основе собранных первичных данных и, соответственно, становится фундаментом для построения архитектуры ИС. Информационная система, тесно связанная с первичными данными, является достаточно стабильной, поскольку определена природой бизнеса, а не его текущей моделью.

Все модели, использующиеся методологией DATARUN, являются взаимосвязанными и разделенными на несколько уровней, каждый из которых определяет спецификации для разработчиков. При этом модель данных не является статичной: в ходе создания ИС она развивается от начальной версии до итоговой спецификации приложения, которое будет использоваться для генерации данных. А полная реляционная модель данных может быть разделена на подмодели, каждая из которых будет представлять отдельные элементы системы, распределенные по сети на основе архитектуры клиент-сервер и в соответствии с архитектурой самой ИС.

На основании вышеизложенного можно сформулировать две цели использования методологии DATARUN.

1. Выделить стабильную структуру, которая станет «каркасом» для создаваемой ИС. Модель данных является такой структурой.

2. Спроектировать информационную систему, которая была бы основана на модели данных.

В табл. 4.2 представлены шаги, выполнение которых необходимо в процессе проектирования информационной системы.

Таблица 4.2

#### Шаги в методологии DATARUN

Действие	Выход
Разработка модели организации	Модель организации
Определение генераторов первичных данных	Генераторы первичных данных
Определение первичных данных	Первичные данные
Конструирование модели первичных данных	Модель первичных данных
Определение расширенной реляционной схемы	Расширенная реляционная схема
Разработка архитектуры компьютерной ИС	Архитектура компьютерной ИС
Отображение на новую организационную модель	Новая модель организации

Методология DATARUN использует следующие модели:

- BPM (Business Process Model) — модель бизнес-процессов;
- PDS (Primary Data Structure) — структура первичных данных;

- **CDM** (Conceptual Data Model) — концептуальная модель данных;
- **SPM** (System Process Model) — модель процессов системы;
- **ISA** (Information System Architecture) — архитектура информационной системы;
- **ADM** (Application Data Model) — модель данных приложения;
- **IPM** (Interface Presentation Model) — модель представления интерфейса;
- **ISM** (Interface Specification Model) — модель спецификации интерфейса.

Вполне логично, что информационная система, которая создается, должна базироваться на основных организационных функциях. Поэтому при помощи BPM создается модель бизнес-процессов. Далее выявляются ключевые информационные объекты, которые документируются как структуры данных. Структуры данных связываются с потоками данных и хранилищами модели, причем для создания структур применяются все организационные документы, будь то инструкции для сотрудников или описание производственных процессов. Работа с их избыточностью или недостаточностью происходит позже, на этапе создания концептуальной модели данных.

После обследования работы организации аналитики приступают к обнаружению и документированию структуры первичных данных. Выделенные структуры переносятся в репозиторий модуля BPM с целью описания циркуляции данных и информации в компании. Таким образом, первичная модель бизнес-процессов оказывается связанной с потоками информации и информационными хранилищами.

После создания модели первичных данных разработчики переходят к созданию концептуальной модели данных (модель ER, модель «сущность-связь»). Концептуальная модель в данном случае отличается стандартизацией, нормализацией и удалением избыточности. В итоге получается модель, которая описывает информацию компании в понятной структурированной форме.

Проектирование архитектуры информационной системы происходит на базе концептуальной модели данных и модели бизнес-процессов. Для построения архитектуры ИС разработчики определяют приложения, которые будут входить в систему. Затем для каждого приложения определяются данные и функции. Архитектура ИС создается на основе нотации ISA. В результате создания данной модели получается рассмотреть все структурные компоненты системы и связи между ними.

Следующим шагом становится проектирование корпоративной БД, которая основывается на реляционной модели. Реляционная БД в дальнейшем детализируется, а с помощью модели системных процессов документируется поведение приложений. Модель системных

процессов создается параллельно с этим в модуле BPM. Она нужна для того, чтобы понять, каким образом реализуются бизнес-процессы. Такая модель создается персонально для каждого приложения.

Каждое приложение в методологии DATARUN состоит из объектов интерфейса. Каждый интерфейс работает с подмножеством баз данных. На данном этапе также уточняются правила обработки данных для каждого отдельно взятого интерфейса.

Описание внешнего вида интерфейса пользователя производится при помощи модели представления интерфейса, которая выглядит так же, как видит интерфейс конечный пользователь. Благодаря этому пользователь может наглядно увидеть прототип приложения еще до его полного создания.

Далее для каждого приложения создается детальная структура, представленная в виде схемы навигации, экранов, отчетов и процедур. Эта структура предполагает достаточно высокую детализацию, основанную на конкретных столбцах и таблицах БД, правилах обработки и экранных формах.

Следующим этапом становится фактическое создание системы, т. е. программирование приложений и их дальнейшая интеграция в информационную систему предприятия.

#### 4.3.2. Инфологическое и даталогическое проектирование

Во время работ, связанных с проектированием информационной системы, разработчики имеют дело с моделями трех разных уровней абстракции: *концептуальными, логическими и физическими*. Модель для каждого из этих уровней представляется в форме схемы, созданной с использованием заранее выбранной нотации<sup>1</sup>.

Наиболее низким уровнем абстракции характеризуется *физическая модель* реляционной базы данных. Такая модель изображается в виде взаимосвязанных таблиц и отвечает (обычно) требованиям третьей нормальной формы или форме Бойса — Кодда. Поскольку модель обладает низким уровнем абстракции, она применяется непосредственно перед созданием БД.

Физическая модель является продолжением и следствием существования более абстрактной модели. Если физическая модель связана с каким-то конкретным ПО, то модель более высокого уровня основана на данных. Модель более высокого уровня отражает данные в форме блоков, которые обозначают сущности и их атрибуты. Такая модель в зависимости от источников называется *логической, даталогической, концептуальной или инфологической*.

---

<sup>1</sup> Боковой Ю. В. Особенности методологии проектирования информационных систем для малого и среднего бизнеса // Прикладная информатика. 2006. № 5. URL: <http://cyberleninka.ru/article/n/osobennosti-metodologii-proektirovaniya-informatsionnyh-sistem-dlya-malogo-i-srednego-biznesa> (дата обращения: 07.10.2020).

Инфологический подход, который используется для создания такой модели, ставит целью рассмотреть смысл данных, а не конкретные способы их представления. Для создания инфологической модели нужно ответить на ряд вопросов.

1. О каких объектах, явлениях или событиях ИС будет сохранять, накапливать и обрабатывать информацию?

2. Какие характеристики объектов и какие взаимосвязи должна учитывать ИС?

3. Какие уточнения ИС должна содержать?

Говоря более строгим языком, создание инфологической модели позволяет определить предметную область системы, т. е. определить, с чем конкретно будет работать создаваемая ИС.

Иногда даталогический аспект рассматривается отдельно от инфологического. Как правило, такое разделение особенно важно в случае вопросов, связанных с представлением данных в самой ИС. Даталогическое проектирование основывается на представлении информации в ИС. При этом представление информации основывается на данных и, соответственно, зависит от возможностей ИС, ее средств, ресурсов и особенностей обработки. Поэтому ИС использует методы и модели, которые предоставляют и преобразуют имеющиеся данные.

Даталогическая модель ориентируется прежде всего на модель «сущность-связь». Задача ER-модели — отразить все имеющиеся сущности, их атрибуты и связи между сущностями, в результате чего происходит исчерпывающее описание предметной области. Для построение ER-модели требуется инструментарий, при помощи которого строятся особые диаграммы ERD<sup>1</sup>.

И, наконец, существует наиболее абстрактная **концептуальная модель**, которая применяется для того, чтобы получить необходимый и достаточный набор сущностей. За счет высокой абстракции построение такой модели — сложный и наиболее ответственный шаг. Формализовать создание концептуальных моделей полностью не представляется возможным из-за широкого разнообразия предметных областей. Тем не менее в данной области существуют определенные методологии и стандарты (к примеру, методология SADT и совокупность стандартов IDEF).

#### 4.3.3. Диаграммы IDEF1X

IDEF1X — метод разработки реляционных баз данных, основанный на условном синтаксисе. При помощи IDEF1X осуществляется концептуальное, логическое и физическое проектирование баз данных. При этом важно помнить, что использование IDEF1X для по-

<sup>1</sup> Боковой Ю. В. Особенности методологии проектирования информационных систем для малого и среднего бизнеса.

строения нереляционной базы данных является зачастую ошибочным решением. Реляционные базы данных требуют выделения ключевых атрибутов, тогда как объектно-ориентированные системы не нуждаются в этом. В результате модель, выполненная в IDEF1X и переданная для реализации методами объектно-ориентированного проектирования, окажется некорректной.

**Сущности в IDEF1X.** Сущностью в IDEF1X называют описание набора однотипных экземпляров, имеющих существенные отличия от других экземпляров хотя бы по некоторым признакам, причем каждый конкретный экземпляр может рассматриваться в качестве практической реализации сущности. К примеру, сущность «Сотрудник» в IDEF1X отражает всех сотрудников предприятия, тогда как Иванов Иван Иванович будет рассматриваться в качестве экземпляра данной сущности. При этом важно понимать, что нотация IDEF1X во многом нацелена не на создание абстрактных сущностей, а на группировку существующих экземпляров по сущностям.

Таким образом, выделение сущностей — это первый шаг. Если был произведен анализ посредством IDEF0, то в качестве сущностей будут рассматриваться Входы, Управление и Выходы. В случае с DFD в качестве прообразов сущностей для IDEF1X будут рассматриваться накопители данных, также потребуется анализ потоков данных.

Выделение сущностей всегда связано с целым набором проблем. Так, очень часто при выделении сущностей происходит путаница из-за использования аналогий. К примеру, вместо общей сущности «Сотрудник» могут быть созданы сущности «Руководитель», «Менеджер» и т. п., хотя такие характеристики было бы уместнее рассматривать в качестве атрибутов. Не меньшую проблему представляют и многочисленные синонимы («Разработка» и «Проект»). Кроме того, зачастую объект может рассматриваться и в качестве сущности, и в качестве атрибута.

Сущность в IDEF1X должна обладать набором свойств, среди которых:

- наличие уникального имени, к которому каждый раз применяется одна и та же интерпретация;
- наличие одного или нескольких атрибутов, которые принадлежат сущности или наследуются через связь;
- наличие атрибута, который однозначно идентифицирует каждый экземпляр сущности (первичный ключ);
- наличие связей с любым количеством других сущностей (или отсутствие таковых).

Выделяют два типа сущностей: **независимая** и **зависимая** (рис. 4.6). Независимая сущность не зависит от существования другой сущности, т. е. каждый экземпляр такой сущности может быть идентифицирован без связи с другой сущностью. Уникальность экземпляра независимой сущности определяется его собственными

атрибутами. Соответственно, идентификация экземпляров зависимой сущности зависит от других сущностей.



Рис. 4.6. Сущности в IDEF1X:  
а — независимая; б — зависимая

**Атрибуты в IDEF1X.** В IDEF1X атрибуты обычно указываются только для сущностей, поэтому наличие атрибута у связи может рассматриваться как сигнал для выделения ее в самостоятельную сущность. Выделяют следующие типы атрибутов:

- **простой** (один компонент с независимым существованием, пример — Имя);
- **составной** (несколько компонентов, пример — Ф. И. О.);
- **однозначный** (может содержать только одно значение для одного экземпляра, пример — Возраст);
- **многозначный** (для одного экземпляра может быть много значений, пример — Контактные номера для связи);
- **производный** (значение атрибута вычисляется на основании других атрибутов);
- **ключевой** (используется для уникальной идентификации экземпляра сущности);
- **неключевой**;
- **обязательные** (ввод обязательен при появлении нового экземпляра сущности).

**Связи между сущностями в IDEF1X.** Связи в IDEF1X описываются глаголами, которые могут иллюстрировать связь между двумя сущностями. При чтении связей на русском языке они могут выглядеть, к примеру, следующим образом: *Отдел <состоит из> нескольких Сотрудников; Сотрудник <пишет> разные Отчеты* и т. п.

Для описания каждой связи в IDEF1X (табл. 4.3) используется набор параметров, перечисленных ниже.

1. *Имя* в виде глагола, который определяет смысловую составляющую связи (состоит из, входит в, делает, ...).

2. *Кардинальность* (мощность): один к одному (1,1), один ко многим (1,N), многие ко многим (M,N). Кардинальность иллюстрирует, какое число экземпляров одной сущности определяется экземплярами другой. К примеру, если Отдел состоит минимум из пяти Сотрудников, то кардинальность будет выражаться в виде (1,5).

3. *Обязательность*: заполнение атрибутов внешнего ключа обязательно или нет.

4. *Степень участия* (количество сущностей, которые участвуют в связи). Как правило, в одной связи участвуют две сущности. Связи,

не являющиеся бинарными или унарными, приводят к бинарному виду.

Таблица 4.3

Типы связей в IDEF1X

Изображение	Тип и обязательность	Кардинальность справа
—	Обязательная, идентифицирующая	1
—●	Обязательная, идентифицирующая	0..∞
—● Z	Обязательная, идентифицирующая	0 или 1
—● P	Обязательная, идентифицирующая	1..∞
—● <число>	Обязательная, идентифицирующая	<число>
-----●	Обязательная, неидентифицирующая	0..∞
◇-----●	Необязательная, неидентифицирующая	0..∞

Можно заметить, что *идентифицирующая* связь отображается при помощи сплошной линии, а *неидентифицирующая* — при помощи пунктирной. Соответственно, для обозначения необязательности в начале линии добавляется ромбик.

На рис. 4.7 приведен пример отображения связи. Связь рассматривается в качестве необязательной, поскольку не все сотрудники обязательно должны входить в какой-либо отдел. Неидентифицирующей связи является потому, что для идентификации сотрудника используется его ФИО или номер служебного удостоверения, а не принадлежность к отделу. Стоит отметить, что неидентифицирующая связь является уникальной для IDEF1X и не применяется в IDEF1.

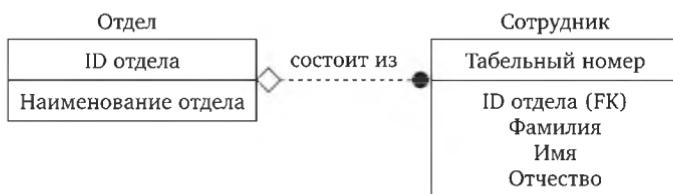


Рис. 4.7. Пример отображения связи в IDEF1X

**Суперклассы и подклассы в IDEF1X.** Суперклассы и подклассы применяются в IDEF1X, когда есть две и более сущностей, которые не сильно отличаются друг от друга по набору атрибутов. В таком случае в модель вводится иерархия наследования, оперирующая категориями суперклассов и подклассов.

**Суперкласс** — это сущность, которая включает в себя **подклассы**. К примеру, Сотрудники компании (суперкласс) может включать в себя два подкласса: штатные и внештатные сотрудники. При этом выделяют два типа иерархии: полная и неполная (рис. 4.8). В случае с полной иерархией мы утверждаем, что существует всего два типа сотрудников, обозначенных выше. Использование неполной иерархии означает, что могут существовать (или уже существуют) другие классы сотрудников, которые на данный момент еще не выявлены.



Рис. 4.8. Полная (а) и неполная (б) иерархии категорий

В случае наличия суперклассов и подклассов данные элементы обозначаются на диаграмме IDEF1X.

Отображение суперклассов и подклассов делает диаграмму не только более понятной, но еще и более удобной для последующей работы с ней при создании информационной системы (рис. 4.9).

**Пример диаграммы IDEF1X.** Приведенной выше информации достаточно для построения диаграмм IDEF1X. Рассмотрим пример такой диаграммы (рис. 4.10).

На рис. 4.10 показано, что название сущности указывается прописными буквами над ее графическим изображением<sup>1</sup>. Атрибуты сущности отображаются внутри графического символа сущности, причем ключевые атрибуты размещаются в верхней части (над чертой). Название связи указывается возле линии, которая используется для ее обозначения.

#### 4.3.4. Диаграммы «сущность-связь» (ERD)

Моделирование данных проводится для того, чтобы разработчики информационной системы получили в свое распоряжение модель (или модели), описывающую систему баз данных. Для моделирования данных чаще всего используется диаграмма «сущность-связь» (ERD), при помощи которой выделяются важные сущности, их свойства и отношения друг с другом.

<sup>1</sup> Сапожкова Т. Е. Интернет-курс по дисциплине «Проектирование информационных систем». URL: [http://e-biblio.ru/book/bib/01\\_informatika/proekt\\_info\\_system/sg.html](http://e-biblio.ru/book/bib/01_informatika/proekt_info_system/sg.html) (дата обращения: 28.10.2020).

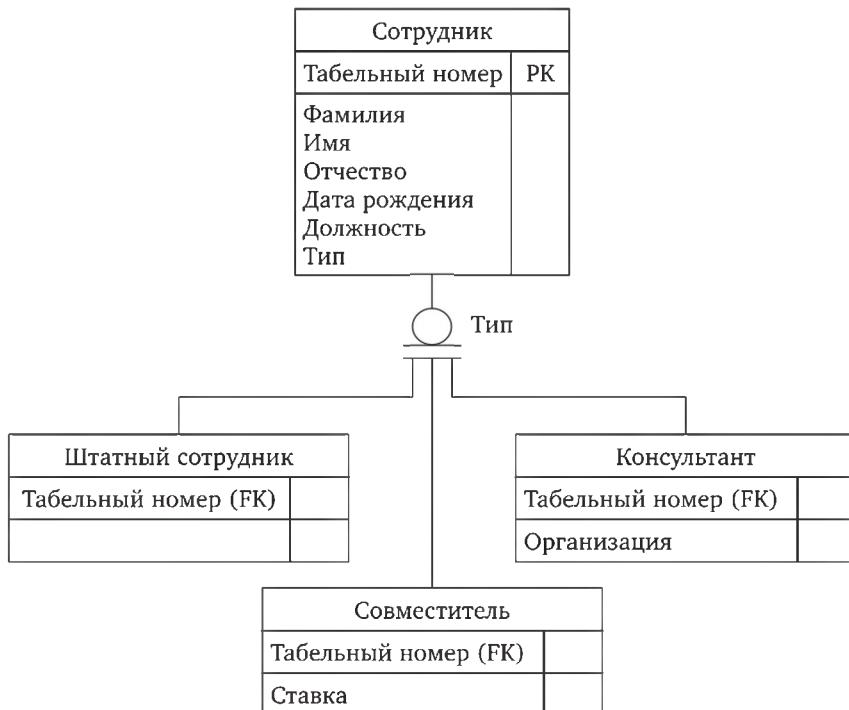


Рис. 4.9. Пример отображения полной категории в IDEF1X<sup>1</sup>



Рис. 4.10. Пример диаграммы IDEF1X

<sup>1</sup> Академия Microsoft на Intuit. Проектирование хранилищ данных для приложений систем деловой осведомленности: метод моделирования «сущность-связь» // INTUIT.RU: Национальный открытый университет. URL: <http://www.intuit.ru/studies/courses/599/455/lecture/10163?page=2> (дата обращения: 07.10.2020).

Впервые диаграммы «сущность-связь» были применены в 1976 г. В качестве основной работы принято рассматривать исследование Питера Чена «Модель сущность-связь — направление к унифицированному представлению данных» (The Entity Relationship Model — Toward a Unified View of Data). Именно после этой публикации нотация Erd приобрела всеобщую популярность и прочно вошла как в научное, так и в практическое применение. В дальнейшем получила она активное развитие в исследовательских работах Баркера.

С помощью Erd определяются важные для конкретного случая объекты или сущности, их отношения, свойства и атрибуты. Важно отметить, что диаграммы «сущность-связь», несмотря на кажущуюся универсальность, эффективнее всего применяются для создания реляционных баз данных. Если планируется использовать иную модель баз данных, то разумнее воспользоваться иным инструментом.

Диаграммы «сущность-связь» основаны на использовании четырех элементов, каждый из которых будет подробнее рассмотрен ниже.

1. Сущность (таблица), под которой понимается набор или класс однотипных физических или абстрактных экземпляров, которые важны для рассматриваемой предметной области. Чтобы понять, что конкретно является сущностью, достаточно ответить на вопрос: информация о чем должна храниться? Сущностями могут быть сотрудники, клиенты, оборудование, детали, запасы и т. п.

2. Экземпляр сущности (запись или строка) — уникально идентифицированный объект.

3. Связь — ассоциация между двумя сущностями. Существуют различные типы связей, например: иерархические, родо-видовые и т. п.

4. Атрибут (столбец или поле) — свойство, которым может быть охарактеризована сущность или связь.

Эти элементы следует рассмотреть подробнее. Несмотря на кажущуюся простоту, диаграммы Erd в классической нотации П. Чена предполагают серьезную вариативность при визуализации элементов и проектируемых моделей в целом.

Прежде всего, необходимо рассмотреть сущности. Диаграммы «сущность-связь» могут использовать три вида сущностей: независимые, зависимые и родительские сущности в иерархической связи (рис. 4.11).

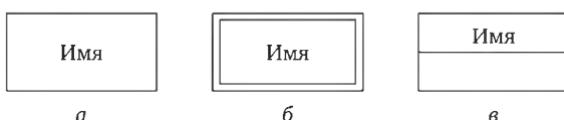


Рис. 4.11. Сущности в нотации Erd:

а — независимая; б — зависимая; в — родительская в иерархической связи

Независимая сущность используется для отображения независимых данных. Таковыми считаются данные, которые всегда присутствуют в системе. Независимая сущность может иметь отношения с другими сущностями в системе, однако это не является обязательным требованием. В свою очередь, зависимая сущность иллюстрирует данные, которые зависят от иных сущностей системы. Соответственно, зависимая сущность обязательно должна иметь отношения с иными сущностями.

Вслед за сущностями необходимо рассмотреть атрибуты нотации ERD (табл. 4.4). В зависимости от сложности модели может быть использовано до пяти типов различных атрибутов. Выделяются обычные атрибуты, первичные ключи, внешние ключи (используются в реляционной модели данных), многозначные атрибуты, получаемые и наследуемые атрибуты (для иерархических связей).

Таблица 4.4

**Атрибуты в нотации ERD**

Графическое отображение	Название
	Атрибут
	Первичный ключ
	Внешний ключ
	Многозначный атрибут
	Получаемый (наследуемый) атрибут

Атрибут в диаграммах «сущность-связь» может быть составным. В таком случае атрибуты-компоненты будут выглядеть как присоединенные к нему эллипсы (рис. 4.12).

Связи используются для иллюстрации отношений между сущностями. Все связи в диаграммах «сущность-связь» отображаются в форме ромба (рис. 4.13), причем имя связи указывается внутри. Двойная рамка ромба обозначает наличие идентифицирующей связи, тогда как одинарная говорит о неидентифицирующей связи.



Рис. 4.12. Атрибуты в нотации Erd

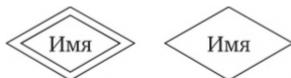


Рис. 4.13. Связи в нотации Erd: идентифицирующая и неидентифицирующая

Важная характеристика связи в Erd — это так называемая *обязательность*. Понятие связи в Erd имеет очевидно выраженный двусторонний характер.

Рис. 4.14 означает, что отдел обязательно должен состоять из сотрудников. Однако каждый конкретный сотрудник не обязательно должен входить в состав рассматриваемого отдела. Двойная линия служит для отображения обязательности, тогда как одинарная линия иллюстрирует ее отсутствие.



Рис. 4.14. Обязательная и необязательная связи в Erd

Важно отметить, что связь также может характеризоваться собственными атрибутами. Атрибуты связи обозначаются как присоединенные эллипсы, и в целом логика их отображения аналогична отображению атрибутов сущностей.

И, наконец, необходимо рассмотреть кардинальность (кратность, мощность) в диаграммах «сущность-связь». Кардинальность иллюстрирует, сколько экземпляров одной сущности может быть связано с каким-либо числом экземпляров другой сущности. Для любых двух сущностей кардинальность может быть задана в виде  $(N;M)$ , где  $N$  — число экземпляров первой сущности, а  $M$  — число экземпляров второй.

На рис. 4.15 показан наиболее простой вариант кардинальности. Приведенная часть диаграммы обозначает следующее: один отдел может состоять из любого числа сотрудников (включая 0, т. е. отсутствие сотрудников). Говоря более строго, один конкретный экземпляр сущности «Отдел» может состоять из различных экземпляров сущности «Сотрудник», причем количество последних варьируется от 0 до бесконечности.



Рис. 4.15. Односимвольное обозначение кардинальности в нотации ERD

Более сложная кардинальность в ERD обозначается диапазоном. Так, на рис. 4.16 показано, что экземпляр родительской сущности (к примеру, отдел маркетинга) должен состоять как минимум из пяти экземпляров дочерней сущности (в данном случае — маркетологов). Цифры (0,1) около дочерней сущности иллюстрируют ее связь с родительской сущностью и означают, что любой сотрудник может входить или не входить в какой-либо конкретный отдел.



Рис. 4.16. Диапазонное обозначение кардинальности в нотации ERD

Иногда классическая нотация П. Чена дополняется элементами нотации Мартина (табл. 4.5). Главная особенность нотации — графическое отображение кардинальности. Однако вариант П. Чена также позволяет отобразить любые возможные варианты кардинальности.

Таблица 4.5

Кардинальность связи в нотации Мартина

Графическое отображение	Значение
—	Кардинальности нет
	1, 1
0	0, 1
-->	M, N
0<=	0, N
--<=	1, N

#### **4.3.5. Информационный инжиниринг**

Подход, получивший название **информационного инжиниринга**, был предложен исследователем Джеймсом Мартином. Данный подход полагает, что создание информационной системы может и должно рассматриваться как процесс создания и развития согласованных моделей, которые охватывают весь жизненный цикл ИС. Такие модели должны охватывать как деятельность всей организации, так и ИС, которую необходимо создать. Иными словами, фундамент информационного инжиниринга — это *совокупность связанных моделей*.

При работе в рамках данного подхода созданные модели сохраняются в специальном **репозитории проекта**, а за счет применения CASE-средств они изменяются, синхронизируются и контролируются. Таким образом, для каждого этапа ЖЦ существуют вполне определенные результаты — полученные модели объектов. В качестве объектов могут выступать организация, требования и пр. Кроме того, стоит отметить, что методология основана на активной работе с заказчиками, которые помогают создать наиболее эффективную систему. За счет доступности и простоты моделей, используемых в данном подходе, ход работы понятен как исполнителю, так и заказчику.

Информационный инжиниринг постулирует, что исходная задача во время создания информационной системы — это построение модели бизнес-процессов, которые существуют в компании и благодаря которым организация достигает поставленных целей. Именно за счет данной модели разработчики получают требования к ИС и, соответственно, начинают работу над ее созданием.

Соответственно, вторым шагом является получение набора моделей, который описывает требования к системе. После того как этот шаг будет закончен, формируется совокупность моделей, которая описывает проект информационной системы целиком. Совокупность моделей будет включать в себя модели архитектуры ИС и модели требований к ИС. Отдельно ведется работа по выделению корпоративных баз данных и формирование требований и пожеланий относительно отдельных приложений.

### **4.4. Описание бизнес-процессов**

Описание бизнес-процессов можно трактовать как описание поведенческой основы информационной системы. Модель бизнес-процессов основывается на функциональной и информационной моделях ИС.

При проектировании ИС набор моделируемых бизнес-процессов определяется, как правило, функциями или процессами, кото-

рые отображаются на последних уровнях диаграмм IDEF0 или DFD. Тем не менее в ряде случаев может оказаться полезным построение более широкого спектра бизнес-процессов организации с целью их дальнейшей оптимизации.

Традиционно описание бизнес-процессов происходило на основе блок-схемы алгоритмов. Сегодня эта нотация во многом считается устаревшей, и для описания бизнес-процессов используют нотации EPC и BPMN.

#### 4.4.1. Бизнес-процессы в анализе и проектировании ИС

Анализ на основе бизнес-процессов используется как для описания деятельности организации, так и для формирования требований к системе. Ситуация такова, что формирование требований к ИС — задача, которую очень тяжело формализовать. При этом цена ошибки очень высока — вплоть до того, что заказчик получит ИС, которая не удовлетворяет его потребности. Кроме того, ошибки, допущенные на этапе формулирования требований, практически невозможно исправить. Поэтому очень высоко значение начальных этапов, на которых формируются требования и на которых требования еще можно изменить.

Бизнес-процессы — это основа работы любой компании. Их содержание определяется целями и задачами организации. Именно бизнес-процессы обеспечивают все виды деятельности компании, которые так или иначе связаны с производством, продажей или поставкой товаров. Любой бизнес-процесс имеет четко определенные начало, конец, интерфейсы и взаимосвязи с другими бизнес-процессами (или с внешним окружением). Каждая работа, входящая в состав бизнес-процесса, имеет свои временные характеристики, благодаря которым такие работы выполняются в нужной последовательности и в заданные сроки. Также бизнес-процесс включает в себя условие собственной инициации и время выполнения.

Процессный подход позволяет объективно описать все виды деятельности организации в контексте ее целей и задач. Поскольку бизнес-процессы направлены на реализацию целей и задач компании, деятельность организации описывается как процесс создания и развития согласованных моделей. Детализация и интеграция использованных моделей предполагает полное сохранение их свойств.

Применение процессного подхода к деятельности организации позволяет сформировать три системы моделей компаний:

- стратегическую;
- укрупненную;
- детальную.

Стратегическая система моделей, создаваемых на основе бизнес-процессов, описывает ключевые особенности организации и виды ее деятельности. Кроме этого, стратегическая система мо-

делей включает в себя элементы, которые не относятся напрямую к организации и ее бизнес-процессам, однако имеют значение при создании ИС. Создание стратегической системы моделей — это первый шаг, с которого начинается описание деятельности организации.

Задачи стратегической системы моделей:

- определить цели и задачи организации;
- сформировать модели бизнес-процессов.

При создании стратегической системы моделей «вертикальная» структура организации не рассматривается. Вместо этого рассматриваются потоки работ, которые происходят независимо от внутренней структуры компаний, отделов, полномочий, иерархии и т. п. Процессный подход рассматривает поток работ в целом, включая внутренние и внешние бизнес-процессы.

Модели, относящиеся к стратегической системе, строятся после обследования организации. Для этого разработчики опрашивают экспертов из числа топ-менеджеров компании, которые определяют стратегию организации. Кроме того, для разработки стратегической системы моделей используются документы компании. Созданные модели сохраняются в специальном репозитории проекта.

После создания стратегической системы моделей строятся другие модели. На этот раз их построение производится на основе обследования деятельности организации, которая теперь рассматривается на уровне подразделений. Эти модели также сохраняются в репозитории проекта, а их построение может происходить параллельно.

**Укрупненная система моделей организации** решает задачу проекции основных бизнес-процессов стратегического уровня на реальную структуру организации. Кроме того, данная система выделяет основные функции подразделений и уточняет состав и характеристики бизнес-процессов. В очередной раз проводится обследование предприятия. В ходе этого обследования выявляются функции, выполняемые подразделениями, а также входы и выходы этих функций. Результатом работы становится уточнение общего списка бизнес-процессов и функций по отделам компании, списки используемых документов и другие характеристики, которые наполняют бизнес-процессы фактическим содержанием.

**Детальная система моделей организации** используется для построения концептуальной модели данных и функциональной модели организации. Для этого производится детальное описание деятельности организации: сначала на уровне описания и реализации общих бизнес-процессов в организации, а затем уровень абстракции уменьшается вплоть до уровня детальных моделей каждого подразделения. В результате выделяются функции каждого подразделения, основные данные, документы, регламенты работ, описывается работа персонала и пр.

Во время обследования каждого подразделения выявляются его функции. В дальнейшем они распределяются по бизнес-процессам подразделения. Тем самым функции отделов наполняются содержанием — конкретными работами, которые осуществляют подразделение. Детализация бизнес-процессов может быть достаточно подробной и включать в себя функции, информационные потоки, входные и выходные документы, взаимодействия внутри компании, данные, бизнес-правила, регламенты, роли персонала, взаимосвязи, временные характеристики и т. п.

В общем и целом процессный подход позволяет сказать, где, кем и когда выполняется каждая функция компании, а также какие данные для этого требуются и откуда они берутся. Система моделей организации может использоваться в следующих целях:

- формирование требований к ИС;
- анализ деятельности организации для ее оптимизации (за счет инжиниринга или реинжиниринга);
- формирование стратегического плана, включающего все этапы ЖЦ информационной системы.

Процессный подход также предполагает создание **системы моделей описания требований к ИС**. Цель создания этой системы — переход от модели описания организации к модели описания информационной системы, которая будет описывать конкретные элементы проекта (БД, приложения, ПО, аппаратные средства и др.). Описание ИС должно также отражать цели и задачи организации.

Для совершения подобного перехода нужно сформировать систему моделей, которая будет описывать требования к ИС, а затем связать эти требования с проектируемыми компонентами. Проводится дополнительная итерация, в ходе которой формируются основные требования к архитектуре ИС, данным, интерфейсу, регламенту работы пользователей, функциям и к управлению системой.

Система моделей требований к ИС включает в себя требования интеграции существующих ИС и БД в новую систему. Итогом создания системы моделей должен стать план создания, развертывания, сопровождения и развития ИС, которая соответствовала бы целям, задачам и стратегии развития организации.

#### 4.4.2. Нотации основных методологий моделирования

**Нотация моделирования бизнес-процессов BPMN (Business Process Modeling Notation)** была впервые представлена публике еще в 2004 г. и описана консорциумом Object Management Group. В основе управления процессами в BPMN лежит идея, что сама стратегия управления опирается на три основные методологии: моделирования бизнес-процессов, анализа и оптимизации. В свою очередь, их поддерживает ряд инструментальных средств, служащих:

- для разработки стратегии, описания, анализа, документирования;
- информационной поддержки бизнес-процессов;
- поддержки потока работ (Workflow management).

BPMN с самого начала создавалась как нотация, подходящая для применения любым пользователем — от бизнес-аналитиков и разработчиков до руководителей, менеджеров бизнес-процессов и просто рядовых сотрудников подразделений. Она объединяет различные точки зрения на бизнес-процесс, тем самым стандартизируя модель.

### Пример

В официальном полном описании нотации BPMN указывается, что для разработки первой версии модели были объединены концепции и некоторые объекты следующих диаграмм и нотаций:

- диаграммы активности UML;
- диаграмма потоков активностей и принятия решений ADF (activity decision flow);
- диаграмма событийных цепочек процесса EPC (event-driven process chain);
- нотация функционального моделирования IDEF (Icam DEFinition for functional modeling);
- другие модели (UMLDOC Business Processes, RosettaNet, LOVeM).

В 2010 г. была опубликована версия BPMN 2.0, созданная при сотрудничестве многих исследовательских групп, в том числе консорциума OMG, Института Хассо Платтнера (Hasso Plattner Institut, Потсдам, Германия), университета Гумбольдта (Берлин, Германия), университетской инициативы Signavio.

В 2013 г. версия BPMN 2.0.1 была принята как международный стандарт ISO/IEC 19510:2013 *Информационные технологии. Модель и нотация процесса менеджмента объекта в групповом бизнесе.*

Основные понятия и группы объектов в BPMN приведены в табл. 4.6.

Таблица 4.6

#### Основные понятия и группы объектов в BPMN

Группы объектов	Описание
Действия	При помощи объектов Действия описываются задачи (бизнес-правила, сценарии, задачи-сервисы, задачи отправки сообщений и др.). Задачи затем детализируются, в том числе за счет маркеров действий (сценариев действий) и определения потоков (порядка и условий выполнения действий)
События	События в BPMN, как и в некоторых других нотациях, относятся к категориям начальных, промежуточных и завершающих. К событиям относятся: отправка и получение сообщений,

Группы объектов	Описание
	таймеры, ошибки, сигналы, остановы и другие виды событий. По сути, событие является индикатором происшествия, требующего дальнейшего участия пользователя, что возможно организовать, НЕ прерывая процесса (в отличие от предыдущих версий нотации)
Логические операторы	Логические операторы определяют порядок наступления событий процесса при ветвлении, слиянии или синхронизации
Данные	Стандартные объекты данных (сообщения, хранилища, коллекции объектов), которые могут использоваться различными действиями
Хореография	Понятие, впервые появившееся именно во второй версии нотации BPMN. Его основная идея в отражении задач взаимодействия (обмена сообщения) между участниками (двумя и более)
Диалоги и взаимодействия	Определение характера информационных взаимодействий: один-к-одному либо один-ко-многим. Отличие от хореографии — в выделении нескольких пулов действий ( <i>swimlanes</i> ) и детальном описании каждого из них (пример таких пулов приведен на рис. 4.17)
Роли	Благодаря группе объектов Роли определяются: <ul style="list-style-type: none"> <li>• распределение обязанностей участников процесса;</li> <li>• информационные потоки между ними;</li> <li>• порядок обмена сообщениями</li> </ul>

Нотация eEPC (Extended Event-driven Process Chain, расширенная событийная цепочка процессов) предполагает описание алгоритма действий, выполняемых отдельными организационными единицами, что позволяет сформировать общий сценарий процесса как последовательность отдельных шагов.

Основными объектами диаграммы eEPC являются элементы, приведенные в табл. 4.7.

eEPC не случайна названа «расширенной» диаграммой — на практике в модели такого вида могут также включаться другие объекты, например:

- товарно-материальные ценности;
- бумажные и электронные документы;
- продукты (используемые и производимые);
- объекты информации;
- информационные системы и их отдельные модули и функции;
- базы данных;
- цели (которые поддерживаются конкретной функцией);
- места выполнения (например, производственный цех № 4);
- другие элементы описания.

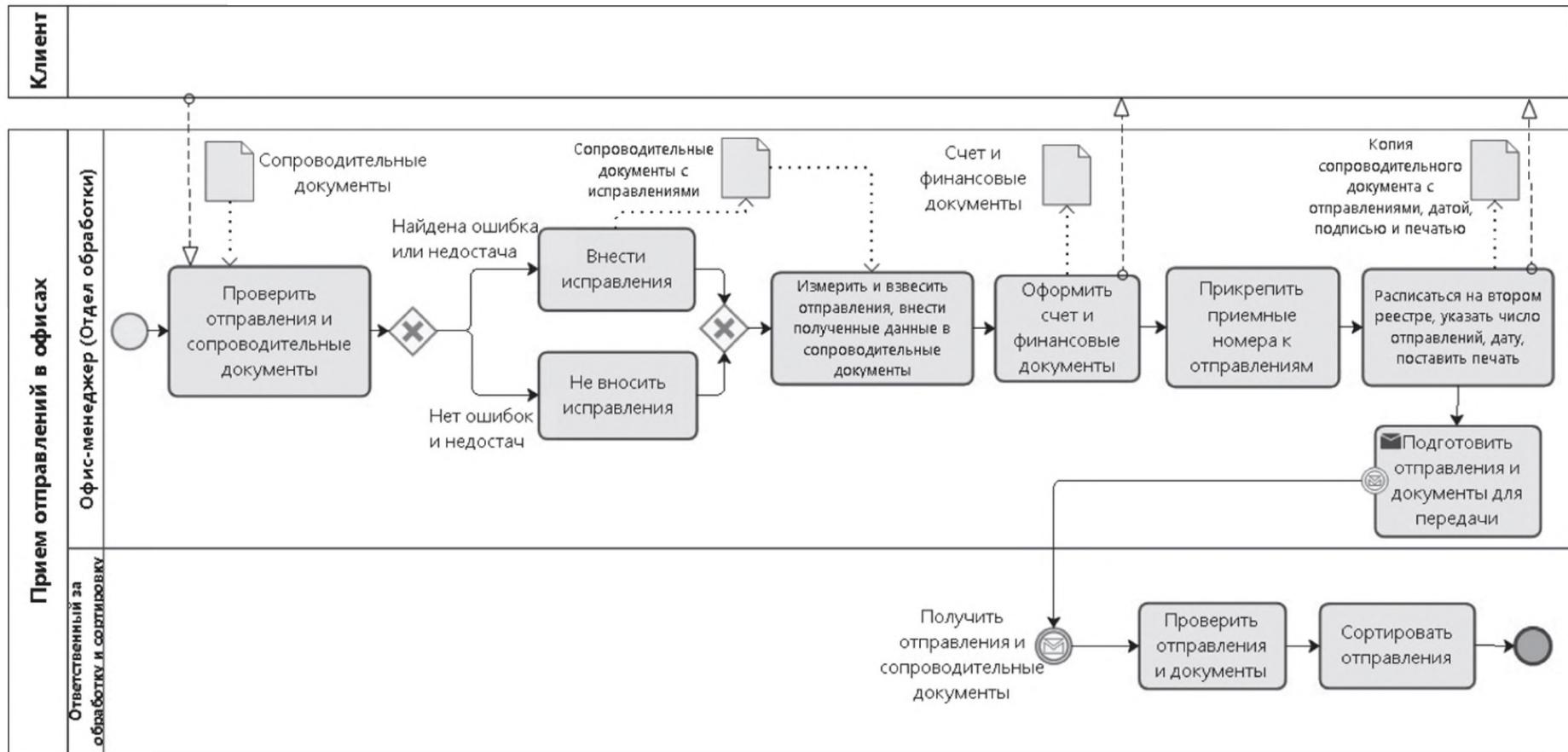


Рис. 4.17. Пример BPML-диаграммы

Таблица 4.7

## Объекты и нотация диаграмм eEPC

Тип объекта	Описание	Графическое обозначение
Событие	Состояние внешней и внутренней среды, являющееся также обязательным условием начала и окончания выполнения функции. Конечные события могут также быть начальными событиями для другого процесса	
Логическое правило	Правила выполнения функции (если наступает только одно из событий или обязательно оба события) и правила наступления событий при выполнении функций (по сути, правила слияния и разделения ветвей процесса)	
Функция	Описание элемента работы, который представляет собой один этап процесса	
Интерфейс процесса	Внешний (по отношению к текущей диаграмме) процесс или функция. Используется для указания взаимосвязи процессов (как для логической последовательности «следующий/предыдущий процесс», так и для обозначения направления передачи объекта)	
Объекты организационной схемы	Обозначение организационных единиц (должностей, подразделений, ролей) — исполнителей, владельцев или участников функций. Бизнес-роли в данном случае рассматриваются как требующие полномочий для управления функциями	

Между всеми объектами в обязательном порядке определяются связи, например, «создает» (документ), «распределяет» (задание между сотрудниками), «использует» (информационную систему 1С), «выполняет» (функцию выполняет менеджер), «принимает решение», «обеспечивает», «является владельцем» и многие другие.

**Пример****Текстовое описание**

На вход процесса поступает запрос от клиента, который должен быть обработан. Ответственность за выполнение данной функции возлагается на отдел продаж. По результатам обработки запроса будет сформирован заказ в системе 1С.

### Графическое описание

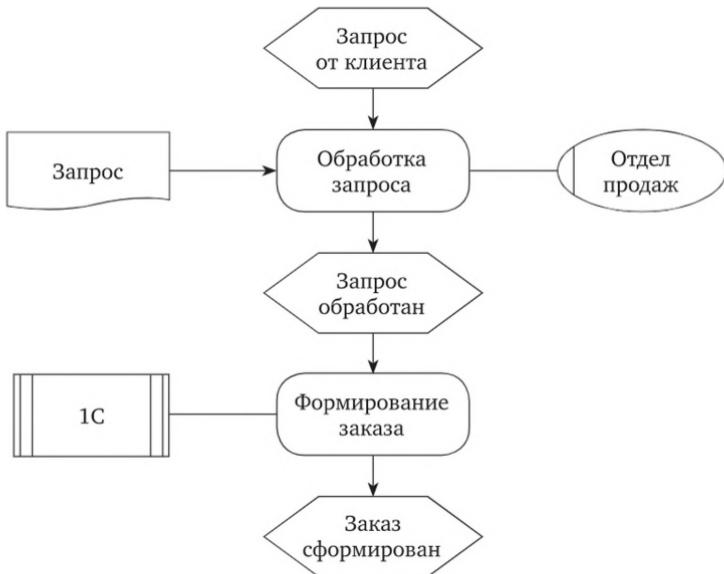


Рис. 4.18. Графическое описание алгоритма

Как видно из рисунка 4.18, графическое описание алгоритма позволяет составить интегрированную картину процесса. Она, в свою очередь, может в дальнейшем использоваться либо как основа для «As-Is»-анализа, либо для последующей доработки и автоматизации как самого процесса, так и управления им с точки зрения достижения целевых показателей эффективности.

**Архитектура интегрированных программных систем ARIS** — инструментальная система моделирования бизнеса, разработанная компанией IDS Scheer AG и ныне принадлежащая Software AG.

ARIS и как методология, и как платформа обеспечивает проектирование, внедрение и управление бизнес-процессами как в процессе повседневной работы, так и для целей анализа, оптимизации и реинжиниринга бизнес-процессов.

Модель организации в ARIS может быть определена как совокупность взаимосвязанных и взаимодополняющих графических моделей, которая адекватно описывает моделируемые предметные области деятельности организации (рис. 4.19).

Таким образом, ARIS как методология позволяет моделировать такие подсистемы предприятия, как организационная, функциональная, информационная, процессная и подсистема входов/выходов. Они формируются на трех уровнях описания, иерархически

идущих от анализа проблем бизнеса к конкретной реализации при помощи ИТ:

- уровень требований (семантические модели);
- уровень спецификации (бизнес-описания с ориентацией на информационные технологии);
- уровень реализации (модели, еще более детализирующие спецификации на уровне информационных технологий).

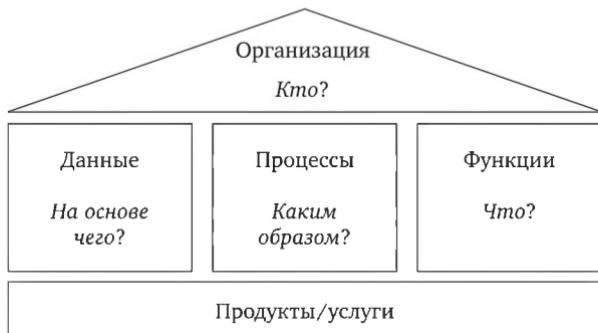


Рис. 4.19. Структура ARIS

Для каждого из представлений программный продукт ARIS предусматривает различные виды диаграмм:

- организационная диаграмма;
- диаграмма данных ERM;
- диаграммы BPMN 2.0;
- процессный ландшафт (цепочка добавленной стоимости VACD);
- системный ландшафт (диаграмма компонентов);
- иерархическая диаграмма активностей (whiteboard);
- диаграмма бизнес-процессов EPC;
- диаграмма ИТ-инфраструктуры (сети);
- диаграмма общего вида.

Методология ARIS исходит из предположения, что деятельность предприятия может быть полностью описана при помощи иерархии моделей. Также используются различные инструменты, которые позволяют получить новые возможности (табл. 4.8).

Таблица 4.8

#### Расширения ARIS

Дополнительные инструменты	Предоставляемые возможности
Процессная аналитика	Визуализация проблем производительности Получение отчетов/оповещений о достижении критических отметок показателей процессов

Окончание табл. 4.8

<b>Дополнительные инструменты</b>	<b>Предоставляемые возможности</b>
	Мониторинг данных, процессов и их ключевых индикаторов (например, функционально-стоимостной анализ затрат)
Управление рисками	Управление бизнес-операциями, рисками и требованиями, контроль исключений
Управление политиками и соответствием требованиям регуляторов	Формирование карт политик в бизнес-контексте с разграничением зон ответственности
	Сочетание политик управления требованиями, рисками и процессами
	Ведение журнала аудита с возможностью формирования сложных отчетов (в том числе по контрольным панелям)
Управление взаимодействием: создание рабочей платформы коллаборации	Организация общих обсуждений процессов/приложений/данных
	Представление моделей процессов в формате web-страниц
	Публикация моделей процессов в интранет-сети компании
	Возможность для специалистов компании предложить свои улучшения в процессах
Симуляция	Возможность «прогона» (симуляции) моделей BPMN 2.0 и EPC для выявления различий моделей As-Is и To-Be
	Получение статистики и сводной информации по результатам симуляции моделей в режиме реального времени
	Возможность проведения сценарного анализа «Что, если» для определения степени зависимости результата и показателей процессов от определенных факторов и групп факторов
Моделирование бизнес-правил	Возможность определения логики бизнес-правил и связывания сформированных с их помощью структурированных моделей с электронными таблицами (например, для анализа стоимости процесса)
Управление оптимизацией бизнеса	Интеграция информации о структуре и значениях ключевых показателей эффективности процессов, моделях бизнес-процессов, информации о центрах затрат для поддержки принятия стратегических решений

#### **4.4.3. Программные продукты моделирования деятельности организации**

Использование программных продуктов, поддерживающих ту или иную методологию описания процессов, в масштабах всей компании дает несколько неоспоримых преимуществ. Во-первых, это помогает сузить как спектр специализированных решений для моделирования, навыки работы с которыми необходимы сотрудникам компании, так и спектр детально изучаемых методологий. Это, в свою очередь, расширяет потенциальные границы использования на предприятии созданных моделей процессов. Во-вторых, программные решения позволяют использовать единый внутренний репозиторий объектов, что способствует связности моделей и проведению их верификации.

Перевод информации о процессах организации в цифровой вид и моделирование описания процессов в конкретной нотации требует выполнения следующих действий:

- организация рабочей группы и проведение ее заседаний в целях анализа имеющихся данных по бизнес-процессам, их объектам и существующим взаимосвязям;
- создание предварительных моделей существующих процессов (в состоянии «как есть»), их последующее обсуждение, доработка и согласование;
- преобразование данных о процессах в вид моделей выбранной методологии описания в виде «как есть»;
- создание, доработка и утверждение моделей вида «как должно быть»;
- непрерывный анализ, мониторинг и оптимизация процессов.

Поддерживать системный и структурный уровни рассмотрения организации, а также модели бизнес-процессов позволяют CASE-технологии, реализованные в том числе в виде отдельного класса программного обеспечения (Computer-Aided Software/System Engineering). Данные системы позволяют системным аналитикам и командам разработчиков после общения с бизнес-заказчиком сформировать проект описания деятельности компании и в дальнейшем перейти к ее автоматизации.

Завершая разговор о различных моделях описания процессов и инструментальных средствах, перечислим возможности основных платформ моделирования.

**CA ERWin Data Modeler.** Комплекс CASE-средств, среда моделирования и коллективной работы с данными, проектирования и генерации баз данных. Интеграция процессов и данных (с IDEF0, DFD, IDEF3), проверка и оптимизация дизайна баз данных, доступ к метаданным и их интеграция.

Среди ключевых функций:

- визуализация структур данных;
- создание проектов баз данных (по графическим моделям);

- определение стандартов (например, шаблонов моделей);
- сравнение моделей и баз данных;
- отчетность и публикация (в том числе в HTML-виде с экспортом изображений графических моделей и метаданных).

Данный программный продукт интегрирует возможности BPWin и ERWin, обеспечивая также создание логических и физических моделей и трансформацию между ними.

**Bizagi BPM Suite.** Bizagi BPM Suite — достаточно широко известная и популярная система для моделирования бизнес-процессов и управления ими. Данная система состоит из трех компонентов:

- Bizagi Process Modeler (дизайнер процессов);
- Bizagi Studio (автоматизация процессов);
- Bizagi BPM Server (исполнение процессов).

Моделирование бизнес-процессов осуществляется в нотации BPMN. В целом система считается достаточно удобной, если не требуется моделировать большое предприятие. В противном случае работа в графическом редакторе может быть затруднена.

**Microsoft Visio.** Один из первых инструментов создания графических диаграмм. Со временем расширяя репозиторий доступных моделей и новых объектов, он существенно упрощает создание наиболее распространенных видов диаграмм, хотя и не предоставляет автоматизированных инструментов их анализа.

**ARIS Express.** Все базовые возможности ARIS реализованы в программном продукте ARIS Express, полнофункциональная версия которого находится в свободном доступе. В ней пользователи могут создавать все девять базовых типов диаграмм.

Кроме этого, ARIS Express предоставляет такие возможности, как печать моделей, экспорт в форматах PDF и EMF, создание собственных шаблонов диаграмм, панель инструментов моделирования, онлайн-поддержку, а также автоматическая генерация моделей из определенных форматов документов и импорт диаграмм MS Visio.

**ARIS Platform.** Гораздо более широкой функциональностью обладает семейство программных продуктов ARIS Platform (в 2013 г. — ARIS Architect и ARIS Designer). Этот продукт исходит из идеи, что любая организация может быть описана с помощью целой иерархии моделей (от верхнего уровня цепочки добавленной стоимости VACD до диаграмм конкретных процедур и ресурсного окружения).

В нем доступны также центральный репозиторий моделей, версионность, поддержка множества языков, многопользовательская работа, конструктор отчетов WYSIWYG, конфигурируемые метамодели и многие другие возможности.

Кроме того, для формирования специфических аналитических отчетов, регламентов процессов, создания базы моделей по спецификациям возможно использовать технологию ARIS Script и писать сценарии (подпрограммы) на языке SAX Basic.

**Software AG ARIS** (ранее **IDS Scheer ARIS**). Моделирование организации, процессов, данных, функций, продуктов и услуг. Поддержка создания ИС от анализа требований до разработки (через интерфейсы экспорта моделей в средства разработки). Дополнительные возможности работы с системой сбалансированных показателей, операционными рисками, стандартами качества.

## **4.5. Объектно-ориентированный анализ**

Объектно-ориентированный анализ и проектирование ИС может рассматриваться и в качестве дополнения к функциональному подходу, и как самостоятельное направление для анализа и проектирования ИС. В первом случае объектно-ориентированные методологии начинают применяться только на стадии проектирования ИС. А во втором они могут охватывать весь жизненный цикл информационной системы самостоятельно или использоваться в комплексе со структурными методологиями.

Методологии, относящиеся к структурному и объектно-ориентированному подходам, серьезно различаются между собой. Перечислим их основные отличия.

**1. Подход к декомпозиции.** Структурный подход предполагает проведение декомпозиции системы на основе функций. В результате разработчики получают иерархическую структуру взаимосвязанных функций, причем по мере движения «вниз» уровень абстрактности снижается. Объектно-ориентированные методологии используют совершенно иной подход. В рамках объектной декомпозиции ИС разбивается на набор объектов, которые условно соответствуют объектам в реальном мире, причем объекты также взаимодействуют друг с другом, но не образуют ярко выраженной иерархии.

**2. Объединение данных.** Из содержания данной темы становится очевидным, что структурные методологии четко разграничивают атрибутивные данные и поведение. Это выражается в создании отдельных функциональных, инфологических и поведенческих моделей. Объектно-ориентированный анализ и проектирование предполагают совместное хранение атрибутов и поведения объектов.

**3. Структурная организация внутри модулей ИС.** Модули, создаваемые в рамках структурного подхода, состоят из функций, имеющих иерархическую связь друг с другом, причем функция может состоять из подфункций и т. д., вплоть до получения требуемого уровня детализации. Объектно-ориентированный анализ и проектирование используют другие виды отношений: композицию и наследование. В результате общая модель объектно-ориентированного подхода представляет собой не иерархию, а граф более общей структуры.

Для проведения объектно-ориентированного анализа и проектирования сегодня используются различные инструменты, методы, подходы и методологии. Чаще всего в рамках объектно-ориентированного анализа и проектирования говорят об унифицированном процессе и унифицированном языке моделирования UML. Они будут подробно рассмотрены в следующей теме.

## **Контрольные вопросы и задания**

1. Зачем применяется структурный анализ?
2. Какие основные нотации используются для описания бизнес-процессов?
3. Какие нотации приняты в качестве международных стандартов?
4. Перечислите программные продукты, которые используются для поддержки нотаций.
5. Каковы основные положения методологии SADT?
6. Каковы характеристики семейства методологий IDEF?
7. В чем заключаются основы методологии ARIS?
8. Каковы основные элементы диаграммы ERD?
9. Каковы основные элементы диаграммы DFD?
10. Когда и зачем применяются диаграммы IDEF1X?

## **Рекомендуемая литература**

1. Грекул, В. И. Проектирование информационных систем // Национальный открытый университет «ИНТУИТ», 2012. — URL: <http://publications.hse.ru/books/74833061> (дата обращения: 17.10.2020).
2. Елиферов, В. Г. Бизнес-процессы: регламентация и управление : учебник / В. Г. Елиферов. — Москва : ИНФРА-М, 2015.
3. Информационные системы и технологии управления : учебник / под редакцией Г. А. Титоренко. — 3-е изд., перераб. и доп. — Москва : ЮНИТИ-ДАНА, 2011.
4. Исаев, Р. А. Банковский менеджмент и бизнес-инжиниринг. В 2 томах / Р. А. Исаев. — 2-е изд., перераб. и доп. — Москва : ИНФРА-М, 2013.
5. Реинжиниринг бизнес-процессов : учебное пособие / под редакцией А. О. Блинова. — Москва : ЮНИТИ-ДАНА, 2010.

# **Тема 5**

## **ПРОЕКТИРОВАНИЕ**

---

Эта тема посвящена подробному рассмотрению фазы проектирования ИС. Приводятся общие требования к методологиям проектирования ИС. Рассматриваются история развития методологий проектирования и основные принципы проектирования. Подробно описаны классы методов и технологий проектирования, среди которых каноническое проектирование, типовое проектирование и проектирование с использованием CASE-средств.

Особое внимание уделяется объектно-ориентированному анализу и проектированию. Тема включает описание логики применения UML (Unified Modeling Language) при проектировании архитектуры ИС. Представления архитектуры, которые отражают высокий уровень абстракции, далее детализируются основными диаграммами UML. Тема также содержит обзор основных CASE-средств объектно-ориентированного анализа и проектирования.

После изучения данной темы студент будет:

**знатъ**

- общие требования к методологиям проектирования ИС;
- историю развития методологий проектирования ИС;
- принципы проектирования ИС;
- основы канонического проектирования;
- основы типового проектирования;
- основы проектирования с использованием CASE-средств;
- основы объектно-ориентированного анализа и проектирования;
- основные CASE-средства объектно-ориентированного анализа и проектирования;

• основные принципы и методы документирования требований;

**уметь**

- производить объективный выбор между различными классами методов и технологий проектирования и использовать их;

- применять основные принципы проектирования ИС;

**владеть навыками**

- моделирования архитектуры ИС на основе модели «4+1 представления» на языке UML;

- создания основных типов диаграмм языка UML (диаграммы прецедентов, классов, деятельности, взаимодействия, состояний, компонентов, развертывания);

- использования основных CASE-средств объектно-ориентированного анализа и проектирования.
-

## **5.1. Общие требования к методологиям проектирования ИС**

Использование методологий в интересах организации преследует простую и вполне понятную цель: организовать процесс создания информационной системы и обеспечить управление этим процессом таким образом, чтобы созданная ИС гарантированно отвечала потребностям организации, а также соответствовала предъявляемым к ней требованиям. Таким образом, любая методология проектирования ИС учитывает те же аспекты, на которых основана работа с любыми проектами: качество ИС, сроки создания ИС и бюджет на создание ИС.

Чтобы создание ИС стало эффективным с точки зрения качества, сроков и бюджета, используются исчерпывающие описания различных процессов на протяжении всего жизненного цикла информационной системы, в результате чего создание становится более простым, удобным и выгодным. Кроме того, методология проектирования корпоративных ИС включает в себя возможности для сопровождения, модификации и наращивания ИС, а также позволяют «вписать» создаваемую ИС в уже имеющуюся ИТ-инфраструктуру.

Задачи методологии проектирования ИС:

- обеспечить создание ИС, отвечающей требованиям заказчика и целям организации;
- гарантировать создание системы при условии соблюдений срока, бюджета и требований по качеству;
- поддерживать дисциплину сопровождения, возможности для модификации и расширения;
- обеспечивать преемственность разработки.

Чтобы обеспечить работу с полным жизненным циклом любой корпоративной ИС, методология реализуется посредством прикладных технологий, стандартов, методик и программно-технических средств<sup>1</sup> (рис. 5.1).

**Технологии проектирования** — основная составляющая методологии проектирования ИС. Сегодняшняя практика позволяет определить технологию проектирования как совокупность трех составляющих.

1. Пошаговые процедуры, которые определяют последовательность операций, связанных с проектированием. Такие процедуры содержат не только сами последовательности, но также условия выполнения операций и их описания.

2. Критерии и правила, при помощи которых оцениваются результаты технологических операций.

---

<sup>1</sup> Вендоров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем // CITForum : Библиотека on-line. URL: <http://citforum.ru/database/case/index.shtml> (дата обращения: 15.06.2020).

3. Нотации, при помощи которых описывается проектируемая ИС.

Для применения технологий проектирования ИС требуется наличие стандартов, которые должны соблюдаться всеми, кто участвует в проекте. Соответственно, **стандарт проектирования** должен устанавливать:

- набор моделей для каждой стадии проектирования и описание степени детализации моделей (или диаграмм);
- правила фиксации решений на моделях (наименование, атрибуты объектов, правила оформления диаграмм, требования к форме и размерам);
- требования к рабочим местам команды разработчиков (настройки ОС, настройки используемых средств, настройки проекта и др.);
- механизм обеспечения совместной работы (правила интеграции подсистем конечной ИС, правила поддержания проекта в одинаковом состоянии, правила проверки действий и решений на непротиворечивость и т. п.).



Рис. 5.1. Представление технологической операции проектирования<sup>1</sup>

В качестве основных **областей проектирования** корпоративных информационных систем выделяют следующие:

- область проектирования объектов данных, которые в дальнейшем будут реализованы в базе данных;
- область проектирования программ, отчетов и электронных форм, которые будут обеспечивать выполнение запросов к данным;
- область учета среды или технологии, например топологии сети, конфигурации аппаратных средств, архитектуры, параллельной обработки и пр.

<sup>1</sup> Вендоров, А. М. CASE-технологии. Современные методы и средства проектирования информационных систем.

Имеющаяся практика создания КИС многократно демонстрировала важность наличия и грамотного использования методологии проектирования ИС. Более того, во многом именно методология проектирования является главным фактором, от которого в дальнейшем будет зависеть успех или неудача проекта.

Методологии проектирования корпоративных ИС постоянно совершенствуются, и происходит это за счет обобщения опыта профессионалов со всего мира и динамичного развития программно-аппаратных средств.

## **5.2. История развития методологий проектирования ИС**

Методологии проектирования корпоративных информационных систем имеют долгую историю. Рассмотрение этой истории имеет большое значение, поскольку позволяет понять прошлое и будущее методологий данного класса.

На сегодняшний день принято выделять четыре этапа развития методологий проектирования корпоративных ИС.

1. Дометодологический этап (*pre-methodology*).
2. Ранний методологический этап (*early methodology*).
3. Методологический этап (*methodology*).
4. Постметодологический этап (*post-methodology*).

**Дометодологический этап** приходится на 60-е гг. XX в. На данном этапе фокус находился на самом программировании, тогда как потребностям конечных пользователей уделялся минимум внимания. В результате слабого развития технологий основной задачей разработчиков становилось хотя бы создание рабочей системы, а также оптимальное использование ресурсов и обход ресурсных ограничений. При этом основные позиции были у разработчиков-«технологов», которые, как правило, не обладали навыками эффективной коммуникации. Как явствует из названия, создание ИС велось без какой-либо четкой методологии, в связи с чем часто нарушались сроки, увеличивался бюджет, ухудшались результаты и пр. Со временем это привело к созданию стандартов, которые регламентируют создание ИС.

**Ранний методологический этап** связан с появлением первых методологий проектирования ИС. Большое значение по мере про текания данного этапа стала приобретать роль системного аналитика, а вопросам анализа и проектирования уделялось все больше и больше внимания. Монолитные системы и приложения, которые разрабатывались ранее, перестали соответствовать потребностям компаний, в результате чего фокус внимания разработчиков сместился на создание узкопрофильных приложений, которые в дальнейшем интегрировались с существующими ИС и другим ПО ком-

пании. На данном этапе активно применялись блок-схемы, популярность которых стала снижаться в связи с началом методологического этапа.

**Методологический этап** характеризуется появлением и активным развитием новых подходов. Так, в 1970-е гг. возникли: каскадная модель, модель «сущность-связь», диаграммы потоков данных, структурные диаграммы и жизненные циклы сущностей. Появились такие инструменты, как ПО управления проектами, ПО словаря данных, системные репозитории, инструменты графического отображения, CASE-средства.

Широкое применение при создании ИС в 1970-е гг. получил системный подход, который возник задолго до методологического этапа. Целью системного подхода было понять природу больших систем высокой сложности. В рамках данного подхода организации рассматривались как системы, взаимодействующие с внешней средой. Главный упор делался на комплексное понимание ситуации, которое позволяло выделить области изменений в работе компаний, возможные на основе внедрения ИС.

В 1980-е гг. стали формироваться и другие подходы. Новые подходы фокусировались на самых разных аспектах ИС и деятельности компаний. Среди этих подходов выделяют стратегический подход, подход участия, прототипирование, структурный подход и др. На данных группах альтернативных методологий необходимо остановиться подробнее.

- **Стратегический подход.** В рамках стратегического подхода делается упор на предварительное планирование при создании информационной системы, а также на разработку ИТ-стратегии компании. Подход предполагает участие топ-менеджеров в анализе и формировании ИТ-целей организации.

- **Подход участия.** Подходы из данной группы фокусируются на пользователях ИС и их ролях, причем сами пользователи вовлекаются в работу по созданию ИС на всех стадиях, включая анализ, проектирование, внедрение и сопровождение.

- **Подход прототипирования.** В рамках данного подхода за счет использования прототипов разработчики демонстрировали пользователям интерфейс системы и ее базовую функциональность. Такая демонстрация является гораздо более понятной и удобной, чем формальная документация, понятная только специалистам.

- **Структурный подход.** Методологии данного подхода основаны на разбиении большой и сложной проблемы на составные части, которые более понятны и просты для разрешения. Данный подход активно использует диаграммы функций, диаграммы процессов, диаграммы потоков данных и др. Большое внимание уделяется бизнес-процессам компаний.

В 90-е гг. ХХ в. стала активно развиваться так называемая вторая волна методологий методологического этапа. Особого внимания заслуживают **объектно-ориентированные методологии**. Ряд исследователей и практиков утверждают, что объектно-ориентированная разработка — наиболее естественный подход к созданию ИС, который опирается на описание предметной области, а не следование логики процессора при обработке данных. Этот подход позволяет унифицировать процесс разработки за счет повторного использования частей программного кода. Объектно-ориентированный подход характеризуется простотой, наглядностью и понятностью, за счет чего облегчается взаимодействие между специалистами из разных областей. В итоге увеличивается согласованность результатов и появляется возможность учитывать изменения.

Также в 1990-е гг. был популярен **инкрементальный (или эволюционный) подход**. Данный подход предлагал не создавать каждый раз ИС с нуля, а развивать и улучшать уже существующую систему. Это позволяет снизить время разработки и решить проблему изменчивости требований. Система, разрабатываемая в рамках данного подхода, делилась на несколько компонентов, каждый из которых разрабатывался отдельно.

**Постметодологический этап:** критика методологий и неметодологические подходы. Сегодня можно констатировать серьезный кризис методологий. Разработчики и заказчики все чаще высказывают скептическое отношение относительно их полезности и необходимости, в связи с чем широкое распространение получили неметодологические подходы. Практика показала, что использование методологии — не панацея, а в ряде случаев применение методологии никак не влияет на получение положительного результата. Кроме того, выбор методологии не всегда осуществляется правильно, что также сказывается на результатах проекта.

Соответственно, сегодня наблюдается снижение спроса на применение формализованных методологий разработки ИС. Более того, в ряде случаев необходимость их использования и вовсе становится под вопрос. Важный аспект, из-за которого существующие методологии подвергаются критике, заключается в недостаточном освещении ряда вопросов и отсутствии возможностей для использования таких важных инструментов, как *визуализация, когнитивные карты, планирование сценариев, анализ кейсов, анализ требований заинтересованных сторон*.

Существует особый подход, который не привязан к явной и формализованной методологии. Этот подход получил название **ситуационной разработки**. В рамках данного подхода разработчики используют только те инструменты, которые являются подходящими, причем при выборе инструментов они опираются на свой опыт и знания, а не на заранее прописанные стандарты. Распространение

подобных подходов иллюстрирует тенденцию аметодологической разработки ИС, которая имеет ряд минусов, характерных для дометодологического этапа. Ситуативный подход, тем не менее, позволяет снизить риски за счет предоставления структуры, при помощи которой разработчики осуществляют выбор инструментов и техник, а также их адаптацию под конкретную ситуацию

На постметодологическом этапе возникли и активно развиваются *гибкие (Agile) подходы к разработке*, при использовании которых требования изменяются и уточняются в процессе создания ИС. Конечные пользователи вовлекаются в процесс создания ИС, при этом приоритет отдается не созданию полного пакета документации, а получению высокоеффективного работающего продукта. В рамках *agile*-подхода продукт создается небольшими шагами, за счет чего результат соответствует целям компании и потребностям конечных пользователей.

Практика применения гибких подходов показала, что в ряде проектов они могут быть очень эффективны. Но точно так же практика продемонстрировала, что во многих проектах наиболее эффективна каскадная модель, которая строится на основе постоянных требований, сформулированных в начале проекта, а также спиральная модель, которая работает с требованиями, которые уточняются в ходе проекта. Подводя итог, автор придерживается мнения, что важно понимание всех достоинств и недостатков каждого из подходов и понимание контекста их применения.

**Принципы проектирования ИС.** Независимо от конкретных условий и выбранной методологии, проектирование ИС всегда основывается на нескольких общепринятых принципах. Различные исследователи и практики выделяют разное число подобных принципов, и нередко их количество исчисляется десятками. Ниже будут приведены основные принципы проектирования ИС.

**Принцип декомпозиции** в общем смысле означает, что решение сложных проблем осуществляется за счет разбиения их на подпроблемы, которые более пригодны для понимания и разрешения. В проектировании это означает, что проектирование ИС осуществляется за счет ее условного разбиения на различные модули, элементы, компоненты и т. п., каждый из которых имеет взаимосвязи с другими модулями и проектируется отдельно.

Другой, не менее важный, принцип — это **принцип иерархического упорядочивания**. Составные части ИС представляются в виде иерархической древовидной структуры, причем каждый последующий уровень характеризуется все большей детализацией. Соответственно, проектирование ИС может рассматриваться как постепенный спуск от абстрактного описания системы к конкретному описанию ее составных частей.

Независимо от подхода и методологии, проектирование ИС всегда подразумевает соблюдение **принципа концептуальной общности**. Данный принцип означает, что проектирование ИС проводится в рамках подхода, который выбирается изначально и не меняется в дальнейшем.

**Принцип унификации** в проектировании ИС означает, что один и тот же элемент должен иметь одинаковое описание и обозначение независимо от конкретной диаграммы, модели и т. п. При этом однотипное обозначение допускает, что детализация описания может варьироваться в зависимости от типа модели или диаграммы.

**Принцип логической независимости** применяется для того, чтобы разработчики могли сконцентрировать силы на логическом проектировании ИС, тогда как ее физическая реализация рассматривается в качестве отдельного вопроса.

Применение **принципа многомодельности** обусловлено тем, что ни одна модель не в состоянии исчерпывающим образом отразить состояние объекта реального мира. В связи с этим допускается и считается необходимым создание множества моделей, которые дополняют друг друга, освещая различные аспекты моделируемого объекта. Схожее значение имеет и **принцип абстрагирования**, который предписывает фокусировать внимание на наиболее важных элементах системы и ограничивать описание несущественных.

**Принцип инкапсуляции (информационной закрытости)** означает, что элементы системы скрывают друг от друга свое внутреннее содержание. В соответствии с этим принципом информационное взаимодействие элементов системы минимизируется и в большинстве случаев сводится к передаче минимально необходимого объема информации, тогда как внутренние алгоритмы и принципы функционирования остаются недоступными для других элементов.

И, наконец, огромное значение имеет **принцип полиморфизма**, согласно которому объекты с одинаковой спецификацией могут иметь различные варианты реализации.

Нередко в качестве самостоятельного принципа исследователи выделяют **принцип формализации**, который постулирует применение строгого методического подхода при проектировании ИС. Тем не менее некоторые методологии сегодня отклоняются от данного принципа без ущерба для получаемых результатов.

**Модели проектирования ИС.** На этапе проектирования информационной системы в соответствии с описанными выше принципами и выбранным подходом создаются различные модели, которые описывают ИС. Эти модели могут быть классифицированы по различным признакам.

1. По отображаемому аспекту.

- Функциональная модель, которая описывает варианты использования ИС, ее функциональность, объекты и субъекты взаи-

модействия и т. п. Функциональные модели бывают как статическими, так и динамическими.

- Информационная модель, которая описывает состав и структуру данных (реляционные БД, базы классов и пр.). Информационные модели относятся к статическим.
- Поведенческие модели применяются для описания состояний системы и переходов между ними, алгоритмов взаимодействия и обработки информации. Поведенческие модели относят к динамическим.
- Компонентные модели, при помощи которых описывается состав и структура программных и аппаратных средств. Такие модели являются статическими.

## 2. По степени отображения динамики процессов.

- Статические модели описывают состав и структуру модели.
- Динамические модели описывают поведение системы или ее элементов. Главная особенность динамических моделей — явное или косвенное использование понятия «время».

## 3. По степени физической реализации.

- Логические модели, которые описывают структуру ИС, ее состав, состояние, поведение, но без привязки к языкам программирования, СУБД, аппаратному обеспечению и пр.
- Физические модели, описывающие элементы ИС в контексте их конкретной физической реализации.

## 4. По строгости описания.

- Неформальные модели, которые не имеют общепринятого структурированного вида и могут дать только наиболее общее представление об ИС или ее элементе. Как правило, неформальные модели не очень информативны и не могут использоваться для дальнейшего анализа, но иногда их использование необходимо — например, для взаимодействия с бизнес-экспертами.

- Формальные модели имеют три подвида. *Описательные* модели основаны на представлении при помощи специальных документов (бланков, отчетов, таблиц и т. п.). *Графические* модели используют схемы, диаграммы, чертежи и пр. *Математические* модели иллюстрируют ИС в виде математических зависимостей, систем уравнений, логических выражений и пр.

Как правило, проектирование ИС начинается с неформальных моделей, и только потом происходит их постепенная формализация. Логическое проектирование в соответствии с изложенными ранее принципами также происходит раньше, чем наступает время физического проектирования. Динамические и статические модели могут создаваться параллельно в одно и то же время, но гораздо чаще первыми создаются статические модели.

**Классы технологий проектирования ИС.** В общем виде можно выделить три класса технологий проектирования ИС: каноническое

проектирование, автоматизированное проектирование (на основе CASE-средств), типовое проектирование. В табл. 5.1 представлены особенности каждого из указанных типов.

Таблица 5.1

**Классы технологий проектирования ИС**

Класс	Степень автоматизации	Степень типизации	Степень адаптивности
Каноническое	Ручное, автоматизированное	Оригинальное	Реконструкция
Автоматизированное	Автоматизированное	Оригинальное	Реконструкция модели
Типовое	Автоматизированное	Типовое	Параметризация, реструктуризация модели

Все три класса технологий проектирования ИС подробно рассмотрены далее.

### **5.3. Каноническое проектирование**

Каноническое проектирование ИС является одним из распространенных методов, получившим статус «классического». Каноническое проектирование применяется в небольших локальных проектах, реализация которых не требует применения тяжеловесных методологий или программных средств. Каноническое проектирование осуществляется на уровне непосредственных исполнителей проекта, которые в ходе своей работы не используют специализированные инструменты или используют их в ограниченном объеме.

Каноническое проектирование осуществляется на основе каскадной модели жизненного цикла информационной системы. Действительно, каскадная модель достаточно проста в использовании, и ее применение совместно с методом канонического проектирования дает стабильный, предсказуемый и достаточно высокий результат. При этом каноническое проектирование чаще всего используется при создании небольших уникальных информационных систем, поскольку отсутствие инструментальных средств или ограниченные возможности их применения затрудняют ведение масштабных проектов.

Принципы канонического проектирования достаточно просты. Основная единица обработки данных — это задача, которую исполнителям предстоит решить. Соответственно, предметная область рассматривается в контексте отдельных задач и комплексов задач, с которыми предстоит столкнуться.

Отдельного внимания заслуживает определение задачи. Метод канонического проектирования рассматривает задачу как совокупность операций, при помощи которых набор исходных данных преобразуется в значимую информацию, которая нужна для принятия управленческого решения или для осуществления управленческой функции.

В рамках метода канонического проектирования для каждой задачи требуется указать:

- наименование задачи;
- сроки или периодичность решения задачи;
- степень формализуемости задачи;
- источники информации, которые понадобятся для решения задачи;
- количественные характеристики и (или) показатели задачи;
- алгоритмы расчета показателей и методы контроля расчетов;
- средства сбора, передачи и обработки данных;
- требуемую точность решения задачи;
- трудоемкость задачи;
- формы представления исходных данных и полученной из них информации;
- конечных потребителей полученной информации.

Метод канонического проектирования в самом общем виде предполагает наличие следующих стадий, входов и выходов (рис. 5.2).



Рис. 5.2. Стадии, входы и выходы канонического проектирования

Подробная детализация всех стадий в данной теме производиться не будет, поскольку стадии ЖЦИС подробно разбираются в соответствующих темах данного учебника.

### **Каноническое проектирование на основе ГОСТ 34.601—90**

Проектная документация при каноническом проектировании обычно формируется на основе требований ГОСТ 34.601—90 «Автоматизированные системы. Стадии создания». Для определения проектной документации могут применяться и другие документы, однако на сегодняшний момент ГОСТ 34.601—90 является наиболее распространенным в России при каноническом проектировании.

ГОСТ 34.601—90 определяет следующие стадии создания информационной («автоматизированной» в терминах документа) системы.

1. Формирование требований к автоматизированной системе.
2. Разработка концепции автоматизированной системы.
3. Техническое задание.
4. Эскизный проект.
5. Технический проект.
6. Рабочая документация.
7. Ввод в действие.
8. Сопровождение автоматизированной системы.

На каждом этапе создания автоматизированной системы выполняется заранее определенный набор работ и формируются специальные документы. В табл. 5.2 представлены основные этапы работ согласно ГОСТ 34.601—90 для каждой стадии создания автоматизированной системы.

Таблица 5.2

#### **Детализация стадий создания автоматизированной системы по ГОСТ 34.601—90**

Стадия	Этапы работы
1. Формирование требований к АС	1.1. Обследование объекта и обоснование необходимости создания АС. 1.2. Формирование требований пользователей к АС. 1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания)
2. Разработка концепции АС	2.1. Изучение объекта. 2.2. Проведение необходимых научно-исследовательских работ. 2.3. Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя. 2.4. Оформление отчета о выполненной работе
3. Техническое задание	Разработка и утверждение технического задания на создание АС

Стадия	Этапы работы
4. Эскизный проект	4.1. Разработка предварительных проектных решений по системе и ее частям. 4.2. Разработка документации на АС и ее части
5. Технический проект	5.1. Разработка проектных решений по системе и ее частям. 5.2. Разработка документации на АС и ее части. 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку. 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации
6. Рабочая документация	6.1. Разработка рабочей документации на систему и ее части. 6.2. Разработка или адаптация программ
7. Ввод в действие	7.1. Подготовка объекта автоматизации к вводу АС в действие. 7.2. Подготовка персонала. 7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями). 7.4. Строительно-монтажные работы. 7.5. Пусконаладочные работы. 7.6. Проведение предварительных испытаний. 7.7. Проведение опытной эксплуатации. 7.8. Проведение приемочных испытаний
8. Сопровождение АС	8.1. Выполнение работ в соответствии с гарантийными обязательствами. 8.2. Послегарантийное обслуживание

Приложение к ГОСТ 34.601—90 содержит детальное описание работ, включая списки требуемых документов. К примеру, при выполнении этапа 1.1 проводится сбор данных об объекте автоматизации и осуществляющей деятельности. Проводится оценка качества функционирования объекта и осуществляемых видов деятельности; происходит выявление проблем, решение которых возможно средствами автоматизации. В завершение формируется оценка (технико-экономической, социальной и пр.) целесообразности создания автоматизированной системы в виде документа.

На этапе 1.2 производится подготовка исходных данных для формирования требований к АС. ГОСТ 34.601—90 определяет основные исходные данные, к которым отнесены:

- характеристика объекта автоматизации;
- описание требований к системе;

- ограничения допустимых затрат на разработку, ввод в действие и эксплуатацию;
- эффект, ожидаемый от системы;
- условия создания и функционирования системы.

Также на этапе 1.2 в документальном виде формируются требования к автоматизированной системе.

Для примера проиллюстрируем также этап 1.3. Основным проектным документом в данном случае является отчет о выполненных работах на первой стадии создания автоматизированной системы. Также оформляется заявка на разработку автоматизированной системы или иного документа с аналогичным содержанием.

ГОСТ 34.601—90 может быть адаптирован под конкретные условия проекта. Допускается:

- исключить стадию «Эскизный проект»;
- исключить отдельные этапы работ на всех стадиях;
- объединять стадии «Технический проект» и «Рабочая документация» в одну стадию «Техно-рабочий проект»;
- выполнять отдельные этапы работ до завершения предшествующих стадий;
- параллельное во времени выполнение этапов работ;
- включение новых этапов работ.

## 5.4. Проектирование с использованием CASE-средств

### 5.4.1. Unified Process и Rational Unified Process

Unified Process — методология разработки ИС/ПО, название которой переводится на русский язык как «унифицированный процесс». Унифицированный процесс разработки программного обеспечения был предложен создателями языка визуального моделирования UML. В дальнейшем для полного раскрытия потенциала объектно-ориентированного подхода был разработан UP — процесс производства программного обеспечения. При этом следует понимать, что если UML был стандартизирован OMG, то стандартизация UP не проводилась, даже несмотря на наличие коммерческих методологий и проработанных подходов. Язык UML будет подробно рассмотрен далее в теме 5.

Если задаться вопросом, какова конечная цель процесса производства программного обеспечения, то можно получить достаточно простой и лаконичный ответ: превращение исходных требований в готовое и реально функционирующее ПО.

История UP. Основой унифицированного процесса стал SEP (Software Engineering Process), который изначально использовался для определения, что, когда, как и кто должен делать при разработке программного обеспечения. В 1967 г. компания Ericsson начала

работу по развитию SEP. Изначально подход рассматривал информационные системы в виде взаимосвязанных блоков, причем меньшие блоки образовывали связи друг с другом и «сплетались» в блоки большего размера, из которых, в конечном счете, и состояла ИС.

Исходной посылкой подхода стал принцип «разделяй и властвуй». Действительно, масштабная информационная система чрезесчур сложна для понимания, поэтому требуется упростить задачу разработки. Одно из оптимальных решений — разбиение системы на блоки, каждый из которых имеет какие-либо интерфейсы и собственное внутреннее устройство, а также связь с другими блоками. Данная логика содержится и в современных версиях UP с той лишь разницей, что большие блоки носят название подсистем, а маленькие именуются компонентами.

Одним из наиболее важных нововведений, предложенных Ericsson, стали так называемые «варианты трафика». С их помощью разработчики демонстрировали способы реального применения информационной системы. Варианты трафика в современной реализации существуют в виде прецедентов в UML.

Процесс имел два представления — статическое и динамическое. Статическое представление содержало описание архитектуры ИС, а динамическое — описание взаимодействия блоков. С этой целью использовались диаграммы последовательностей и автоматов, которые и сегодня применяются в UML.

1980 г. ознаменовал собой новую веху в развитии объектно-ориентированного анализа и проектирования. Язык спецификации и описания (Specification and Description Language, SDL) стал одним из первых языков визуального моделирования, а спустя десять лет возникла его расширенная версия. В итоге SDL-92 стал первым стандартом объектного моделирования. Он до сих пор применяется в сфере телекоммуникационных ИС.

Следующим важным этапом стал 1987 г., когда А. Джекобсон основал компанию Objectory AB. Взяв за основу метод компании Ericsson, А. Джекобсон сумел разработать процесс производства программного обеспечения. Новый процесс получил название «Object Factory» и буквально с первых же месяцев своего существования был хорошо встречен коммерческими организациями. Будучи коммерческим продуктом, процесс содержал всю необходимую документацию, программное средство и возможность получения консультаций от специалистов компании-разработчика.

Основой успеха Object Factory стал принципиально новый подход. Процесс разработки программного обеспечения впервые был рассмотрен как самостоятельная система, а потоки работ были представлены в виде отдельных диаграмм. Тем не менее, данный процесс имел свои минусы. Несмотря на революционные идеи, в каждом конкретном случае требовалась широкомасштабная «настройка»

процесса. Коммерческие версии предоставлялись с небольшим набором шаблонов, но даже они редко когда могли использоваться без доработок.

В 1995 г. ситуация стала меняться. Компания Rational купила компанию А. Джекобсона и приняла решение объединить Object Factory со своими разработками. Первым значимым результатом стало создание Филиппом Крухтеном «4+1 представлений» архитектуры, которые по сей день являются фундаментом UP (хоть и неявным). Далее была выделена последовательность из четырех фаз разработки ИС и ПО: Начало, Уточнение, Построение, Внедрение. Начались активные попытки объединить преимущества водопадного и инкрементного ЖЦПО.

Со временем был сформирован новый подход, получивший название ROP (Rational Objectory Process). В новом подходе использовалось понятие рисков, была детализована архитектура ИС, а также были детально проработаны отдельные области, в которых Object Factory был слаб. При этом UML стал языком, при помощи которого представлялись как отдельные модели ROP, так и весь подход в целом.

В 1998 г. возник Унифицированный процесс компании Rational (RUP). С тех пор регулярно возникали новые версии RUP, каждая из которых учитывала ошибки и недостатки предыдущих версий.

Отдельно стоит отметить выход в 1999 г. книги А. Джекобсона «Unified Software Development Process», в которой подробно описывался Унифицированный процесс (UP, но не RUP!) В конечном счете, UP стал открытym процессом создания программного обеспечения, а RUP — коммерческой версией процесса от компании Rational.

*UP и RUP.* Можно сказать, что RUP — это коммерческая версия открытого подхода UP. Сегодня RUP предоставляется и поддерживается компанией IBM, которая поглотила Rational в 2003 г. RUP от IBM включает множество стандартов и инструментальных средств, которые охватывают все рабочие потоки и все фазы RUP. Таким образом, использование RUP предполагает финансовые затраты, однако существенно экономит время компании на поиск бесплатных инструментов для поддержки бесплатного UP. Кроме того, RUP характеризуется хорошим веб-окружением и наличием качественной службы поддержки от IBM.

Говорить об отличиях RUP от UP достаточно трудно, поскольку оба подхода имеют гораздо больше общего, чем отличий. Если в 1999 г. RUP был фактически коммерческой версией реализации UP, то к сегодняшнему моменту он претерпел некоторые изменения. Однако во многом, как было сказано ранее, отличия сводятся к открытости или коммерческой направленности конкретного варианта унифицированного процесса.

Оба подхода моделируют, кем, когда и что выполняется в рамках создания программного обеспечения. Главные различия сводятся к изображению элементов процесса и их названиям (рис. 5.3). Для ответа на вопрос «Кто?» UP использует понятие «Исполнителя» (Worker), тогда как RUP оперирует категориями Ролей (Role). Также существуют отличия в названиях общих видов деятельности. В UP они называются Рабочие потоки, а в RUP — Дисциплины.

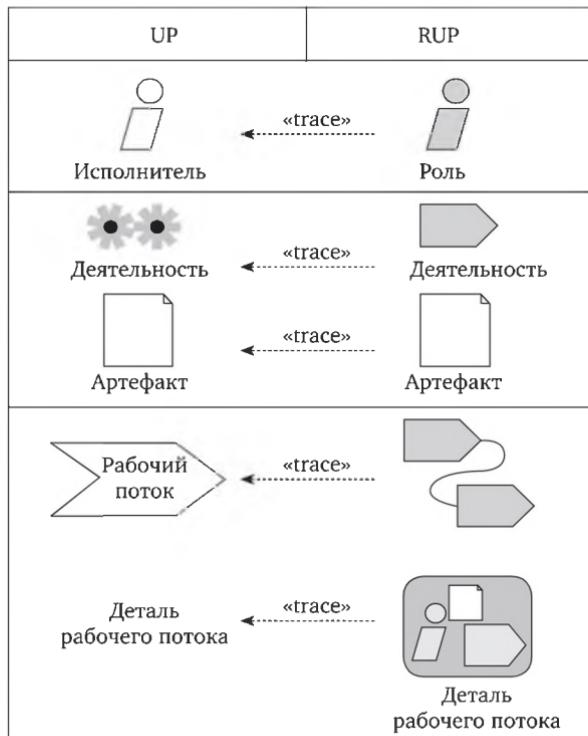


Рис. 5.3. Сравнение UP и RUP

**Основы UP.** Унифицированный процесс основан на трех аксиомах, определяющих его логику, структуру и применение. Согласно этим аксиомам UP:

- управляется требованиями и рисками;
- является архитектурно-центричным;
- является итеративным и инкрементным.

Управление требованиями и рисками — часть фундамента UP. Управление требованиями рассматривается как основа основ, которая позволяет получить по завершению процесса качественную

с точки зрения заказчика информационную систему. Не меньшее значение имеет и управление рисками, поскольку едва ли существуют проекты, реализация которых не была бы сопряжена с какими-либо неприятными неожиданностями, обычно приводящим к крупным финансовым затратам или времененным издержкам.

Фокус на архитектуре в рамках обсуждения UP означает, что применение унифицированного процесса нацелено на создание и развитие надежной архитектуры ИС. Архитектура в данном контексте включает в себя подсистемы, компоненты этих подсистем и их взаимодействие друг с другом, в результате чего обеспечивается итоговая функциональность ИС. Также констатируется, что продуманная архитектура — залог качественной ИС.

**Рабочие потоки.** Рабочие потоки — это основные виды деятельности в UP, которые определяют, что конкретно должно быть сделано и какие навыки или знания необходимо для этого применить. При этом рабочие потоки делятся на два типа — основные и вспомогательные. Основные рабочие потоки таковы:

- *бизнес-моделирование* — производится моделирование деятельности предприятия, формирование предметной области, общий анализ и т. п.;
- *требования* — собираются данные о том, что именно должна делать система, какой функциональностью она должна обладать;
- *анализ и проектирование* — требования уточняются, структурируются, а затем реализуются в системной архитектуре;
- *реализация* — происходит создание программной части;
- *тестирование* — созданная программная часть проверяется на ошибки и на соответствие функциональным и нефункциональным требованиям;
- *развертывание* — система развертывается на рабочих местах конечных пользователей или тестовой группы.

К вспомогательным рабочим потокам относятся:

- *управление конфигурацией и изменениями* — деятельность, направленная на управление версиями системы, внесение изменений, документирование запросов на изменения и т. п.;
- *управление проектом* — общая деятельность проектного управления;
- *среда* — деятельность, связанная с обеспечением рабочей среды для команды разработчиков в целом и управляющих проектом в частности.

В каждой итерации могут присутствовать все рабочие потоки, перечисленные выше. Тем не менее, в ходе отдельной итерации какие-то рабочие потоки могут оказаться ненужными. К примеру, на ранних стадиях проекта практически отсутствуют рабочие потоки Тестирования, Реализации и Развёртывания. А на поздних стадиях нет потребности в Бизнес-моделировании и Требованиях.

**Фазы UP.** Унифицированный процесс создания ИС содержит четыре основных фазы: Начало, Уточнение, Построение и Внедрение. Каждая фаза включает в себя одну и более итераций, а внутри каждой итерации реализуются рабочие потоки (рис. 5.4). Число итераций внутри каждой фазы зависит от контрольных точек, создание которых обусловлено общей логикой проекта. Как правило, рекомендуется делать не более 3—5 итераций внутри одной фазы, а для средних и маленьких проектов рекомендуемое число итераций снижается до 1—2. При этом считается, что средняя длительность итерации не должна превышать 2—3 месяца.

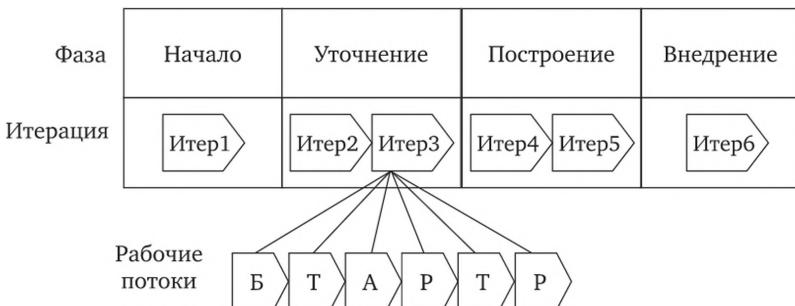
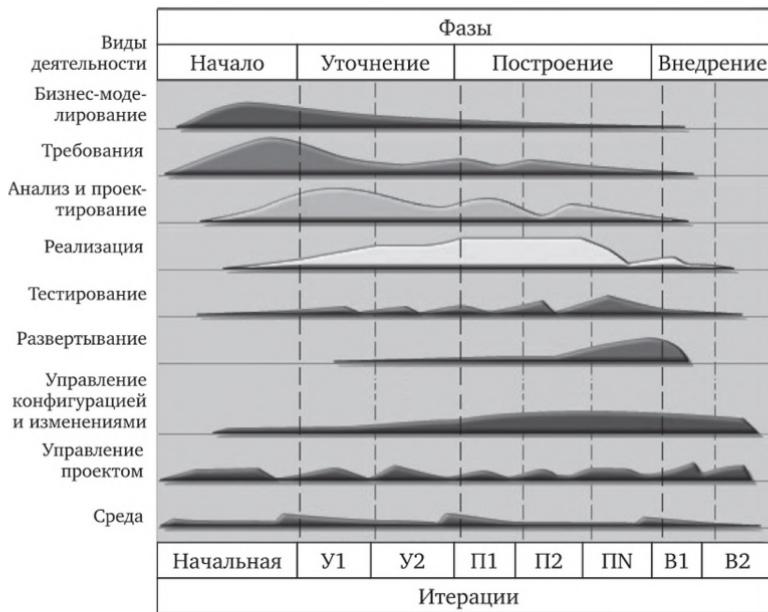


Рис. 5.4. Общая логика UP

В конце каждой итерации команда разработчиков создает промежуточную версию ИС и документации. Версия ИС, созданная в конце последней на текущий момент итерации или фазы, называется базовой. Базовая версия является основой для дальнейшей разработки ИС. При этом изменения в базовую версию могут быть внесены только посредством формальных процедур, которые определяются вспомогательными рабочими потоками. В противном случае качество итоговой ИС может оказаться недостаточным.

На рис. 5.5 отображены все фазы UP, все рабочие потоки и объем работы, который выполняется на каждой фазе. Можно увидеть, что на фазе «Начало» наибольший объем работ приходится на рабочие потоки «Бизнес-моделирование», «Требования» и «Анализ и проектирование». «Управлению конфигурацией и изменениями» уделяется минимальное внимание, как и «Среде». А «Разворачивание» и «Тестирование» практически не выполняются.

**Особенности фаз UP.** При использовании унифицированного процесса важно понимать одну важную деталь: UP нацелен, прежде всего, на достижение целей каждой фазы, а не на создание определенного набора артефактов. Это выражается в наличии контрольных точек в каждой фазе, для достижения которых должны быть выполнены определенные условия. Пока условия не выполнены, фаза не может быть закончена.



*Рис. 5.5. Распределение объема работ по фазам RUP*

**Фаза «Начало».** Фаза «Начало» имеет простую и достаточно понятную цель — заложить основу проекта и сдвинуть его с мертвой точки. Для этого производится обоснование выполнимости проекта. Для этого команда разработчиков и представителей заказчика проверяет, может ли быть реализован проект в существующих условиях и (или) на требуемых платформах, аппаратных мощностях, в рамках выделенного бюджета и т. п. Иногда с этой целью создается концептуальный прототип, чтобы сторона заказчика могла убедиться в возможности удовлетворения бизнес-требований. В этом случае рабочий поток Реализации также будет присутствовать в фазе «Начало». Важно отметить, что даже в этом случае «Тестирование» отсутствует — прототип обычно не тестируется просто потому, что в дальнейшем он не будет использоваться.

Вслед за этим производится разработка экономического обоснования проекта. Организация — заказчик ИС должна быть уверена в том, что новая система окажется коммерчески выгодной. Экономическое обоснование также включает в себя приблизительную оценку ожидаемых издержек владения системой, общих затрат на проект, оценку потерь в случае отказа от разработки ИС и т. п.

Вслед за этим производится определение требований, на основании которых формируется предметная область. Правильно сформулированные требования — это основа успешного создания ИС,

поскольку UP предполагает постепенное движение от требований к конечному продукту.

И, наконец, в рамках фазы «Начало» выявляются основные риски, которые могут существенно сказаться на выполнимости проекта.

**Фаза «Уточнение».** Первая и наиболее ярко выраженная цель фазы «Уточнение» — это создание базовой версии архитектуры информационной системы. Параллельно с этим производится детализованная оценка рисков и определяются основные атрибуты качества, будь то скорость реагирования на выявленные дефекты, максимально допустимое количество дефектов и пр.

В рамках данной фазы должны быть выявлены практически все прецеденты, на основании которых будет создаваться итоговая функциональность системы. Разумеется, выявить их все сразу обычно не удается, но к этому следует стремиться, поскольку внесение крупных изменений на последних фазах может привести к серьезному увеличению стоимости проекта или сорвать сроки сдачи ИС.

Ближе к середине фазы «Уточнение» начинается работа над планом следующей фазы. В случае форсированных проектов работа над планом следующей фазы запускается в самом начале текущей фазы. Это позволяет существенно уменьшить время реализации проекта. Учитывая развитые инструментальные средства поддержки UP (и особенно RUP), менеджер проекта не оказывается перегруженным даже в такой ситуации.

И, наконец, уже на фазе «Уточнение» требуется определить, какие ресурсы потребуются для создания ИС, какой штат сотрудников будет в дальнейшем обслуживать ИС и работать с ней, какие аппаратные ресурсы понадобятся для этого и какова ее конечная стоимость.

К концу фазы «Уточнение» на основе сформированной архитектуры ИС должна быть создана исполняемая система. Важно понимать, что эта исполняемая система — не прототип, а базовая версия, которая в дальнейшем будет развиваться. Соответственно, базовая версия системы должна тщательно тестироваться на основе подхода, предлагаемого UP.

Наибольший объем работ приходится на следующие рабочие потоки:

- требования — на основе требований производится детальное описание предметной области;
- анализ и проектирование — формулируется представление об архитектуре ИС, которая затем проектируется;
- реализация — создание базовой версии архитектуры;
- тестирование — проверка базовой версии на работоспособность;

В конце фазы «Уточнение» должна быть создана надежная базовая версия архитектуры ИС, что и является контрольной точкой.

При этом важно, чтобы созданная архитектура уже на данном этапе учитывала выявленные риски. Параллельно с этим происходит переоценка рисков: выявляются новые риски, детализируются старые и т. п. Экономическая сторона проекта и сроки его выполнения также должны быть окончательно оговорены и заверены всеми заинтересованными сторонами.

**Фаза «Построение».** Определение требований, анализ и проектирование практически завершаются на данной стадии. Главная цель фазы — развить базовую версию системы до полноценной ИС. При этом разработчики часто сталкиваются с проблемой поддержания целостности системы. Как правило, это происходит из-за пренебрежения формальными аспектами UP. Менеджер проекта обязан обеспечить выполнение формальных процедур и сохранить качество ИС на высоком уровне.

Содержание фазы по рабочим потокам выглядит следующим образом:

- бизнес-моделирование — минимальная активность или ее полное отсутствие;
- требования — сбор и выполнение неучтенных ранее требований, если это возможно в рамках определенных ранее проектных ограничений;
- анализ и проектирование — завершение проектной и аналитической моделей;
- реализация — самый важный рабочий поток данной фазы, активности которого нацелены на создание функционирующей версии ИС;
- тестирование — проверка системы на корректное выполнение базовых функций и соответствие наиболее важным нефункциональным требованиям;
- развертывание — не менее важный рабочий поток, активности которого нацелены на развертывание ИС на компьютерах бета-тестеров.

Отдельно стоит отметить возрастающую роль вспомогательного рабочего потока «Управление конфигурацией и изменениями». На данной фазе создается множество версий ИС, а сами разработчики вносят значительное число изменений. Для синхронизации версий у команды разработчиков и учета изменений применяются методы, регламентированные данным рабочим потоком.

Контрольная точка фазы «Построение» считается достигнутой в тот момент, когда информационная система может быть передана пользователям для предварительного тестирования (бета-тестирования). Для этого ИС должна быть стабильной, а заинтересованные стороны должны быть готовы к предварительному внедрению ИС. Разумеется, расходы на создание ИС должны укладываться в план, который был окончательно одобрен на предыдущей фазе.

**Фаза «Внедрение».** Главная цель фазы «Внедрение» — это развертывание созданной информационной системы для использования конечными пользователями. К этому моменту все дефекты, выявленные в рамках бета-тестирования, должны быть устранены, а сама система должна быть подготовлена для внедрения и, если это требуется, для распространения.

На фазе «Внедрение» рабочие места пользователей подготавливаются для развертывания созданной ИС. Настраивается работа системы, а в случае появления новых проблем созданное ПО изменяется для их устранения.

Именно на фазе «Внедрение» создается пользовательская документация и происходит обучение конечных пользователей. Как правило, команда разработчиков предоставляет консультационные услуги для заинтересованных сторон и конечных пользователей.

Довольно часто можно встретиться с практикой постпроектного анализа, в ходе которого разработчики анализируют свою деятельность с целью ее оптимизации и улучшения. Отдельные аспекты постпроектного анализа, касающиеся производительности и эффективного использования ресурсов, могут быть презентованы представителям заказчика для повышения имиджа команды исполнителей, если таковая представлена сторонней компанией.

Рабочие потоки «Бизнес-моделирование», «Требования», «Анализ и проектирование» и «Тестирование» практически не участвуют в последней фазе. Рабочий поток «Реализация» выполняется, только когда требуется изменить созданный программный код для устранения ошибок. «Развертывание», напротив, становится наиболее важным. Возрастает роль вспомогательных рабочих потоков, в особенности «Управления проектом» и «Управления конфигурацией и изменениями».

В качестве контрольных точек фазы «Внедрение» рассматривают завершение бета-тестирования, проведение приемочных испытаний, выпуск продукта и начало его использования. Требуется также согласовать дальнейшую стратегию поддержки продукта.

#### 5.4.2. Рабочий поток «Бизнес-моделирование»

**Представление прецедентов.** Разработка информационной системы начинается с представления прецедентов. Представление прецедентов (модель Scenarios) отражает наиболее существенные функциональные требования к ИС. Данная модель описывает требуемую функциональность системы (в виде прецедентов, англ. Use Cases), системное окружение ИС (в виде акторов, англ. Actors) и связи между ними.

Представление прецедентов состоит из одной или нескольких диаграмм прецедентов. При формировании данного представления может использоваться принцип декомпозиции, согласно которому

сначала строится диаграмма прецедентов с наиболее абстрактными прецедентами, которые детализируются в виде отдельных диаграмм прецедентов по мере необходимости.

Диаграммы прецедентов основаны на небольшом числе элементов графической нотации (табл. 5.3).

Таблица 5.3

**Нотация для моделирования представления прецедентов и построения диаграмм прецедентов**

Компонент или вид связи	Элемент нотации
Актор	 Actor
Прецедент	
Ассоциация	—
Направленная ассоциация	—>
Наследование	—>
Зависимость	.....>
Включение	«include» .....>
Расширение	«extend» .....>

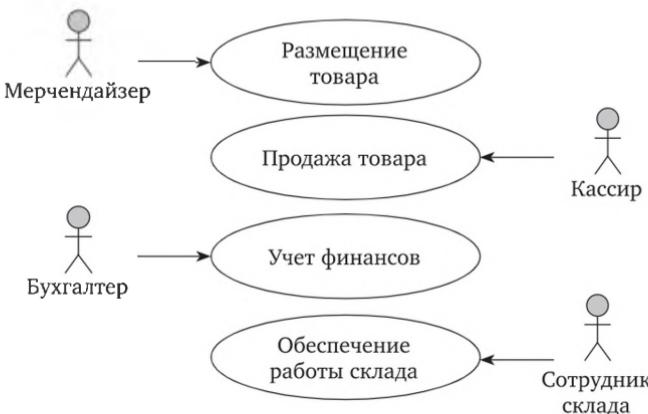
1. Актор (Actor), в качестве которого может выступать человек, устройство или другая ИС. Актор может быть как получателем, так и поставщиком информации. Актор может участвовать в работе прецедента или инициировать его. Для пользователя системы имя актора соответствует его роли. В некоторых случаях Актором может быть время.

2. Прецедент (Use Case) — это последовательность действий, которая инициируется актором, варианты взаимодействия актора с ИС. Графически прецедент отображается в виде овала, внутри которого содержится название прецедента.

3. Различные виды связи.

На начальной фазе отображаются прецеденты, основанные на ключевых функциональных требованиях к ИС (рис. 5.6). В дальнейшем в модель представления прецедентов будут добавлены и другие прецеденты. Логика построения архитектуры ИС, управляемая прецедентами, изложена далее.

Представление прецедентов, как и любое другое, формируется итерационно. Результатом конечной итерации должно стать формирование стабильной архитектуры ИС, учитывающей основные архитектурные компоненты. В этом случае начинается детализация моделей архитектурных представлений при помощи профильных диаграмм UML.



*Рис. 5.6. Пример моделирования представления прецедентов на начальной фазе создания архитектуры ИС*

К представлению прецедентов обычно относят также диаграммы последовательности и диаграммы кооперации, графические нотации которых будут подробно рассмотрены в дальнейшем.

**Формирование диаграмм прецедентов.** На более поздних этапах рабочего потока «Бизнес-моделирование» детализируются прецеденты, отраженные в представлении прецедентов.

Приведем пример диаграммы прецедентов для ИС условного магазина, занимающегося розничной торговлей. После изучения деятельности магазина может быть получена диаграмма прецедентов, которая отражает основные виды деятельности магазина (рис. 5.7).

На рис. 5.7 приведен упрощенный пример диаграммы прецедентов. Пунктирная линия со словом «extend» означает, что прецедент «Заказ товаров» может быть включен в прецедент «Учет запасов» (например, если новая ИС будет автоматически формировать заказ поставщикам на основании данных о запасах). Однако на текущий момент заказ товаров не является составной частью учета запасов.

Пунктирные линии со словом «include», идущие от прецедента «Продажа товаров», означают, что прецеденты «Продажа по карте» и «Продажа наличными» являются составными. При этом настолько подробная детализация может не производиться.

Квадратной рамкой выделена область автоматизации.

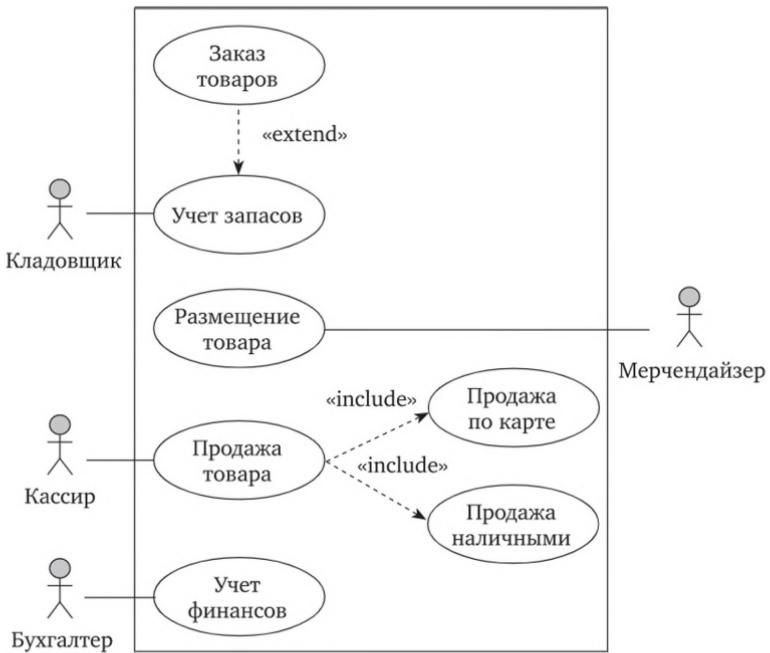


Рис. 5.7. Пример диаграммы прецедентов

#### 5.4.3. Рабочий поток «Анализ и проектирование»

**Процессное представление.** На ранних этапах рабочего потока «Анализ и проектирование» формируется в процессное представление. Процессное представление — это вариант процессной декомпозиции применительно к объектно-ориентированному подходу. Процессное представление принимает во внимание нефункциональные требования к ИС, включая производительность и доступность. Процессная архитектура охватывает вопросы согласованности, распределения, системной интеграции, устойчивости к сбоям. Процессная модель показывает, какую форму принимают основные абстракции логического представления в процессном представлении.

Процесс в рамках данной модели — это набор коммуницирующих друг с другом элементов. Каждый компонент определяется на основе выполнения им какой-либо задачи. Иными словами, процесс — это группировка задач, выраженных в виде единых исполняемых компонентов.

Условно выделяют два вида задач:

- главные задачи (обязательны для отражения), выражаемые в форме архитектурных элементов, имеющих однозначное представление;

- второстепенные задачи (не обязательны для отражения), которые используются на местном уровне по причинам, важным для реализации ИС.

Нотация для моделирования процессного представления архитектуры ИС показана в табл. 5.4. Процессное представление может включать и другие элементы графической нотации диаграммы деятельности. Однако, как правило, большинство из них не имеют архитектурной значимости и не отражаются на высоком уровне. При этом их использование считается допустимым.

Таблица 5.4

#### Нотация для моделирования процессного представления архитектуры ИС

Компонент или вид связи	Элемент нотации
Процесс	Activity
Ассоциация	—
Зависимость	----->
Реализация	.....>

**Диаграммы взаимодействия.** Для детализации элементов процессного представления на поздних фазах рабочего потока строятся диаграммы взаимодействия.

На рис. 5.8 показан пример диаграммы последовательности на примере процесса оплаты банковской картой в магазине. Следует обратить внимание, что подробная детализация здесь, как и в других моделях представлений, не производится. Процесс отражается в виде задач ИС, которые выполняются для реализации функциональности системы.



Рис. 5.8. Пример процессного представления архитектуры

Пример на рис. 5.9 показывает, как аналитик получает отчеты, доступные в соответствующей подсистеме. Аналитик вводит в соответствующую подсистему запрос доступных отчетов. Подсистема проверяет ID пользователя, отправляя соответствующее сообщение в подсистему безопасности. Подсистема безопасности «возвращает» подсистеме генерации отчетов результаты проверки ID, и после этого производится поиск доступных отчетов. В завершение отчеты представляются аналитику.

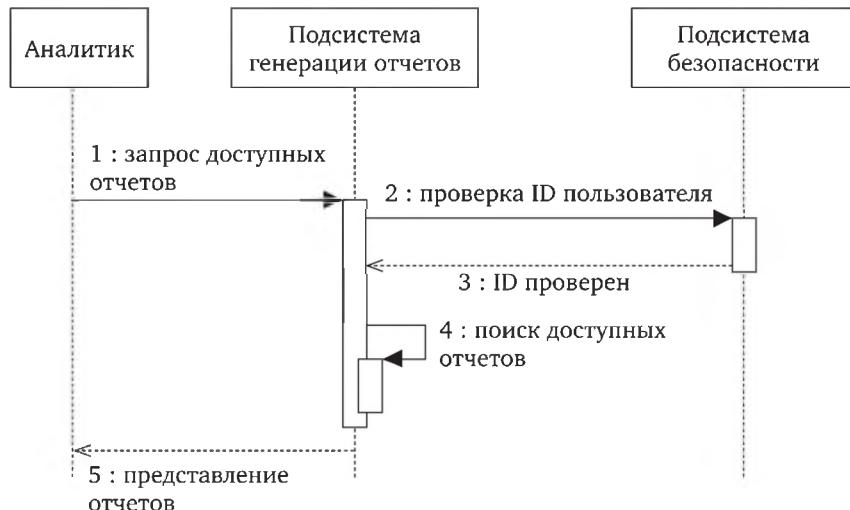


Рис. 5.9. Пример диаграммы последовательности

Вместо диаграмм последовательности могут использоваться диаграммы кооперации, пример которой представлен на рис. 5.10.



Рис. 5.10. Пример диаграммы кооперации

Как можно видеть на рис. 5.10, все объекты, сообщения и отношения абсолютно идентичны таковым на диаграмме последовательности. Разница заключена лишь в их изображении. Диаграмма кооперации позволяет лучше проиллюстрировать, какие отношения существуют между различными объектами внутри одного прецедента.

Как правило, необходимость построения сразу двух типов диаграмм взаимодействия отсутствует. Чаще всего используются диаграммы последовательности за счет их высокой информативности и наглядности, но в ряде случаев разработчики принимают решение построить диаграммы кооперации вместо диаграмм последовательности или наряду с ними.

**Диаграммы деятельности.** Диаграммы деятельности также детализируют процессное представление. В качестве примера будет детализирован прецедент «Размещение товаров», приведенный ранее на диаграмме прецедентов (рис. 5.11). Как правило, диаграмма деятельности, спроектированная в самом начале создания ИС, иллюстрирует текущий вариант деятельности. Однако в дальнейшем, по мере продвижения по этапам жизненного цикла ИС, диаграмма будет неоднократно дополняться и изменяться.

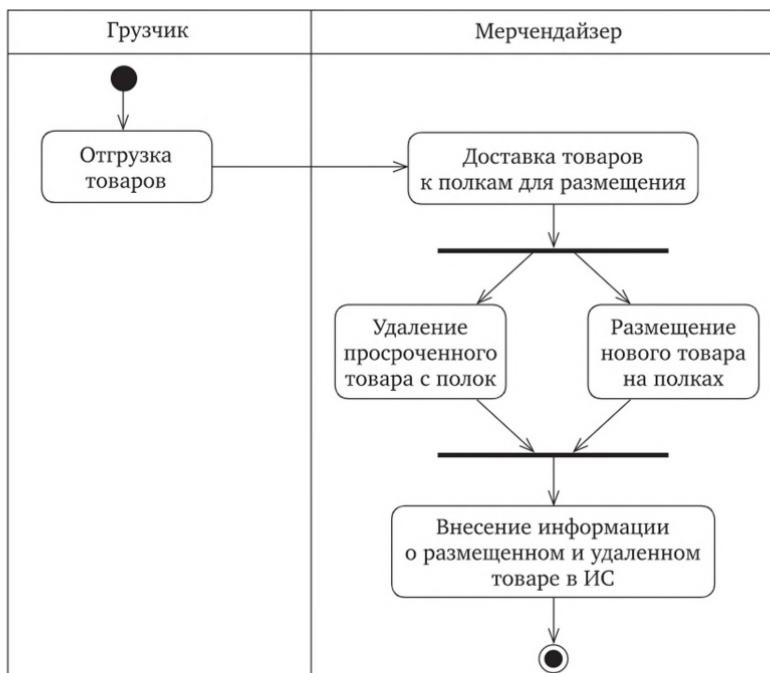


Рис. 5.11. Пример диаграммы деятельности для прецедента «Размещение товаров»

Приведенная на рис. 5.11 диаграмма деятельности иллюстрирует деятельность мерчендайзера по размещению товара на полках магазина. Когда новый товар от поставщиков поступает на склад, мерчендайзер доставляет его к торговым стеллажам и начинает размещение. Параллельно с этим мерчендайзер проверяет наличие просроченного товара. По завершению данных действий мерчендайзер вносит в ИС информацию о количестве и наименовании просроченных товаров, а также указывает, какой товар и в каком количестве был размещен на полках.

Отдельно следует обратить внимание на такой элемент диаграммы деятельности, как «плавательная дорожка» (англ. *Swimlane*). Название происходит из-за внешней схожести с плавательными дорожками в бассейне. *SwimLane* используется для повышения удобства при работе с диаграммой. Каждая «плавательная дорожка» содержит в себе действия одного и только одного актора. На рис. 5.11 видно, что левая плавательная дорожка содержит действия актора «Грузчик», а правая — действия актора «Мерчендайзер».

Диаграммы взаимодействия используются для уточнения диаграммы прецедентов. Диаграммы взаимодействия предоставляют детальное описание логики функционирования прецедента с акцентом внимания на времени или отношениях.

**Логическое представление.** Логическое представление архитектуры ИС — это объектно-ориентированный вариант принципа декомпозиции, который используется в структурном подходе. Логическая архитектура служит прямым отражением функциональных требований к системе. Для создания логического представления архитектуры используются UML-диаграммы классов, объектов и состояний.

На архитектурном уровне логического представления обычно используются элементы графической нотации, приведенные в табл. 5.5.

Таблица 5.5

**Нотация для моделирования логического представления архитектуры ИС**

Компонент или вид связи	Элемент нотации
Класс	
Состояние	
Ассоциация	
Направленная ассоциация	

Окончание табл. 5.5

Компонент или вид связи	Элемент нотации
Агрегация	— ◇
Композиция	— ●
Наследование	— ▷
Зависимость	..... ➤
Реализация	..... ▷

При разработке логического представления архитектуры необходимо избегать преждевременной специализации классов. Классы рассматриваются как наиболее абстрактные объекты без излишней детализации. Подсистемы рассматриваются с точки зрения реализации классов.

Слева на рис. 5.12 можно увидеть, как классы используются в логическом представлении архитектуры. Обратите внимание, что атрибуты и операции для классов не указываются, если только они не имеют архитектурного значения. Необязательно сразу указывать необходимые типы связей, хотя это допускается. Для обозначения большинства связей можно использовать ассоциацию. Справа приведен пример использования состояний в логическом представлении архитектуры. Как и в предыдущем случае, детализация состояний не производится, если только этого не требуют особые условия проекта.

Важно помнить, что избыточная детализация архитектурного уровня не только бесполезна, но и вредна, поскольку архитектурный уровень в любом случае будет детализироваться UML-диаграммами в дальнейшем.

**Формирование диаграммы классов.** Диаграммы классов детализируют классы, отраженные в логическом представлении. На рис. 5.13 приведен фрагмент диаграммы классов, который включает в себя описание класса «Сотрудник» и двух наследуемых классов: «Кассир» и «Мерчендейзер». В качестве кассира и мерчендейзера рассматриваются не конкретные люди, а роль, которая соотносится с одним или несколькими сотрудниками предприятия.

Диаграмма классов является одной из наиболее важных в объектно-ориентированном анализе и проектировании. Диаграмма классов позволяет создать своеобразный «мост» между архитекторами ИС, для которых наибольшее значение имеет предметная область, и программистами, использующими классы для понимания поведения объектов.

**Формирование диаграмм состояний.** Производится также детализация состояний, отраженных в рамках логического представления.

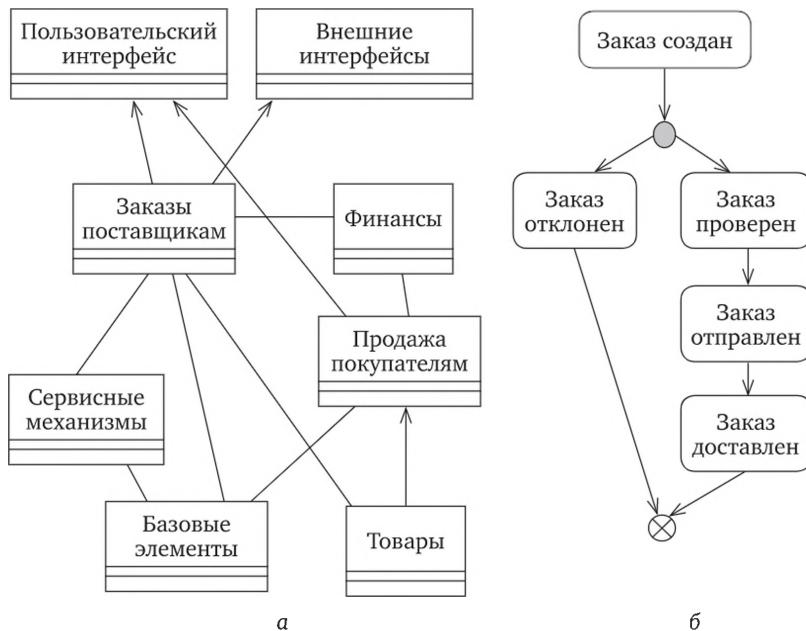


Рис. 5.12. Пример использования классов (а) и состояний (б)  
в логическом представлении архитектуры ИС

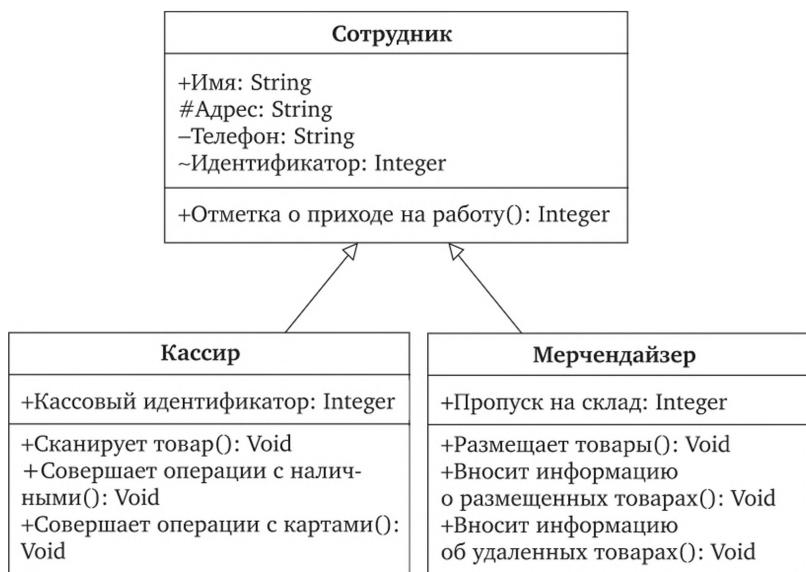


Рис. 5.13. Фрагмент диаграммы классов

На приведенном примере (рис. 5.14) можно увидеть фрагмент диаграммы состояний, который описывает вход в систему кассового аппарата при заступлении кассира на смену.



Рис. 5.14. Пример композитного состояния

#### 5.4.4. Рабочий поток «Реализация»

**Представление реализации.** Представление реализации фокусируется на декомпозиции ИС вплоть до компонентов. Если процессное представление рассматривает компоненты как задачи, выполняемые системой и участвующие в процессе, то представление реализации рассматривает компоненты как небольшие программные библиотеки или функции, реализованные в виде программного кода. Каждый компонент может создаваться одним человеком или маленькой командой разработчиков.

Нотация для моделирования представления реализации архитектуры ИС аналогична нотации для процессного представления (см. выше).

Большое значение для построения грамотного представления реализации имеют Пакеты (*Package*). Пакеты используются в самых разных UML-диаграммах для группировки однотипных элементов. В модели представления реализации пакеты используются для группировки компонентов в подсистемы (рис. 5.15).

Два нижних пакета-«слоя» («Базовые элементы» и «Распределенная виртуальная машина») являются независимыми от предметной области. С их помощью ИС может быть адаптирована под различные программно-аппаратные платформы, установлена на требуемые операционные системы и т. п.

Пакеты «Функциональная зона ИС» и «Интерфейсы» основаны на предметной области ИС. Пакет «Функциональная зона ИС» обеспечивает систему необходимой функциональностью, а пакет «Интерфейсы» — ее взаимодействие с офлайн-продуктами и конечными пользователями.

По мере развития архитектуры ИС нередко выделяется несколько десятков компонентов, сгруппированных по различным пакетам «слоям». В случае очень масштабных ИС их количество может достигать 50—70. На рис. 5.12 отражены лишь некоторые компоненты и слои, которые могли бы быть в ИС магазина розничной торговли.



Рис. 5.15. Пример представления реализации архитектуры ИС

**Диаграммы компонентов.** Представление реализации детализируется за счет построения диаграмм компонентов. Рассмотрим простой пример использования диаграммы компонентов на примере магазина розничной торговли. Рис. 5.16 иллюстрирует взаимосвязь компонентов подсистемы оплаты.

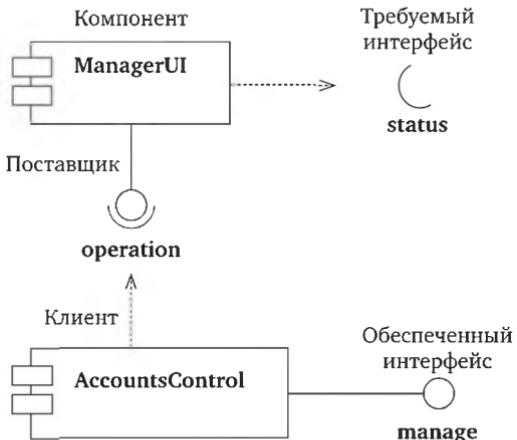


Рис. 5.16. Пример диаграммы компонентов

#### 5.4.5. Рабочий поток «Развертывание»

**Рабочий поток «Тестирование».** Тестирование является самостоятельным рабочим потоком в RUP, однако в учебнике оно не рассматривается детально. На практике в рамках данного рабочего потока осуществляется непосредственное тестирование программных компонентов информационной системы. Обычно для этого используются специализированные инструментальные средства, позволяющие автоматизировать основные виды деятельности внутри рабочего потока.

**Представление развертывания.** На ранних этапах рабочего потока «Развертывание» формируется представление развертывания, которое устанавливает соответствие программных компонентов и аппаратного обеспечения, на котором и будет развернуто ПО. Представление развертывания фокусируется, прежде всего, на нефункциональных требованиях к ИС.

Модель, отражающая представление развертывания, включает в себя основные узлы и экземпляры узлов, если указания последних требуют обстоятельства проекта (рис. 5.17). В дальнейшем на основе этих элементов моделируется физическая реализация системы на архитектурном уровне, которая затем будет детализирована в диаграммах развертывания.

На рис. 5.18 изображено архитектурное представление развертывания для ИС магазина. В таком виде система может быть представлена серверами ИС (основным и запасным), автоматизированными рабочими местами кассиров, сканерами штрих-кодов и терминалами оплаты при помощи банковских карт. Детализация серверов производится по усмотрению в зависимости от потребностей про-

екта. К примеру, можно указать наличие отдельного сервера для подсистемы учета товара и наличие мобильных сканеров для работников склада и (или) мерчендайзеров. Отдельно может быть указан сервер финансовой подсистемы, АРМ бухгалтеров и т. п.

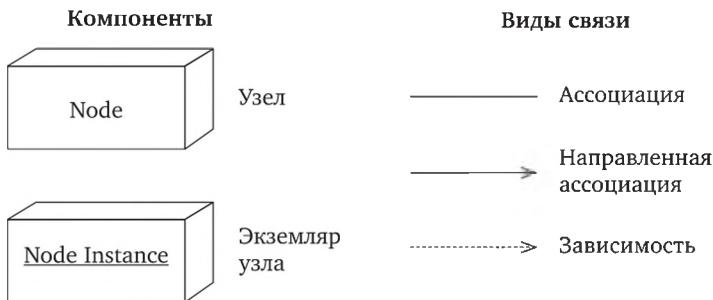


Рис. 5.17. Нотация для моделирования представления развертывания архитектуры ИС

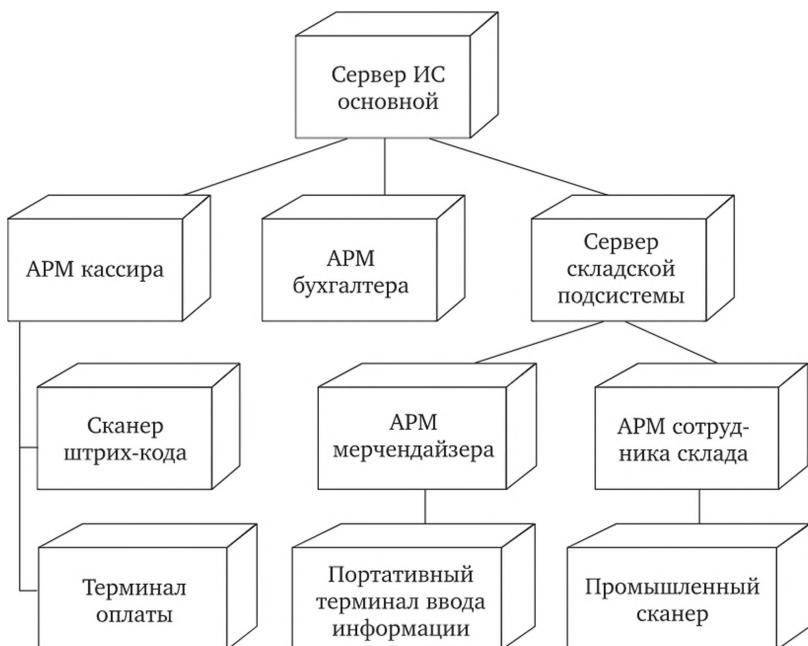


Рис. 5.18. Пример глобального отображения представления развертывания

**Диаграммы развертывания.** Представление развертывания детализируется путем построения диаграмм развертывания.

На рис. 5.19 можно видеть, как артефакты размещены на узлах, а также взаимосвязи между узлами и между артефактами. В даль-

нейшем диаграмма развертывания используется для закупки необходимого аппаратного обеспечения и развертывания на нем созданной информационной системы.

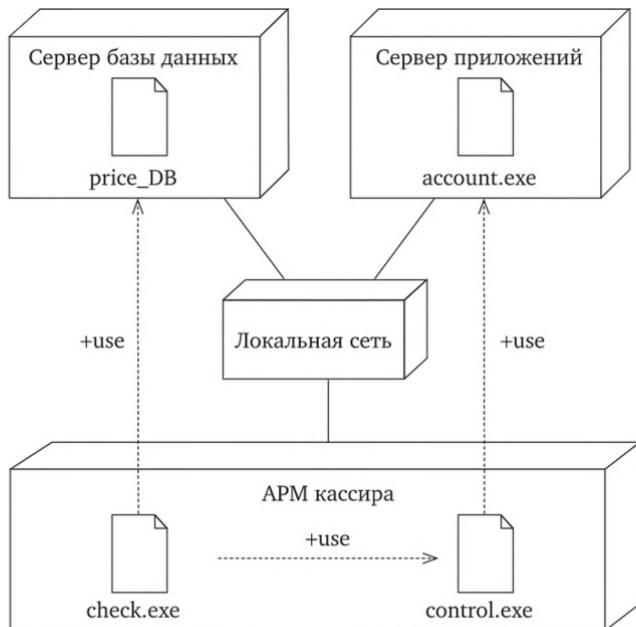


Рис. 5.19. Узлы и артефакты на диаграмме развертывания

#### 5.4.6. CASE-средства объектно-ориентированного анализа и проектирования

Визуальное моделирование на языке UML может осуществляться при помощи различных CASE-средств. Далее будут рассмотрены основные из них.

**Star UML.** Довольно часто в российской практике применяется программное средство StarUML, разработанное компанией MKLab. Изначально целью проекта была попытка создать открытое решение, способное заменить коммерческие приложения для визуального моделирования, наиболее популярные на тот момент — Rational Rose и Borland Together. Можно констатировать, что цель была достигнута, и StarUML стал очень распространенным инструментом для визуального моделирования.

Сегодня существует несколько версий средства StarUML. Изначально StarUML существовал как открытый и бесплатный инструмент визуального моделирования. В таком виде средство развивалось вплоть до версии 5.0.2.1570. На текущий момент не осущест-

вляются поддержка и развитие этого продукта, однако продукт доступен для загрузки по адресу: <http://staruml.sourceforge.net/v1/>. Кроме того, следует отметить, что StarUML характеризуется большим числом пользователей и наличием широкого комьюнити.

Позже на свет появился продукт StarUML 2. Этот продукт является коммерческим. На официальном сайте продукта доступны четыре варианта лицензии: персональная, коммерческая, образовательная и версия для учебных аудиторий. Существует trial-версия программы, предназначенная для ознакомительных целей. Стоимость лицензии варьируется от 40 до 70 долл. США. Последняя версия StarUML 2—2.6.0. Загрузить коммерческую версию можно по ссылке: <http://staruml.io/download>.

Отдельно следует отметить проект WhiteStarUML, который занимается дальнейшим развитием продукта StarUML, начиная с версии 5.0.

Все рассмотренные выше CASE-средства поддерживают UML 2.0, однако существенно различаются по своим возможностям, функциональности, интерфейсу, названиям диаграмм/элементов и т. п. Тем не менее, все средства поддерживают основные диаграммы UML и обеспечивают взаимосвязь диаграмм в рамках единой модели.

**Microsoft Visio.** Microsoft Visio — это векторный графический редактор диаграмм и блок-схем, предназначенный для использования на ОС Windows. Microsoft Visio входит в пакет Microsoft Office. Сегодня MS Visio является достаточно распространенной программой.

Довольно часто при помощи данного средства осуществляется визуальное моделирование с использованием UML. Однако следует понимать, что в случае масштабных проектов требуется применение более узкоспециализированных средств и решений.

При помощи MS Visio можно произвести построение всех основных диаграмм UML.

**IBM Rational.** В 2003 г. IBM поглотила компанию Rational Software. С этого момента IBM предоставляет многочисленные коммерческие решения и средства для визуального моделирования.

- Rational Software Architect — среда разработки и моделирования, использующая UML. Решение поддерживает преобразования типов «модель-код» и «код-модель». Возможны прямые преобразования диаграмм UML в Java, C#, C++, EJB, WSDL, XSD, COBRA IDL, SQL. Обратные преобразования возможны из Java, C++, .NET-языков.

- Rational Rhapsody — семейство программных продуктов, предназначенных для моделирования, проектирования и разработки ПО.

- Rational Rose — продукт для разработки приложений на основе UML. Именно в данном продукте UML стал базовой технологией визуализации и разработки ИС. Функциональность продукта включает моделирование на UML, бизнес-моделирование, разработку ПО

и его архитектуры, кодирование, проектирование, моделирование данных, анализ и визуализацию.

- Rational Modeler — бесплатное средство для разработки программного обеспечения на основе UML. Rational Modeler позволяет создавать законченные приложения. Содержит полноценную среду моделирования. Поддерживает автоматизацию документирования. Может быть обновлен до коммерческого продукта или интегрирован с другими продуктами.

- Rational Tau — среда для разработки сложных ИС и ПО на основе моделей, основанная на языке UML. Rational Tau включает Tau, Tau/Architect и Tau/Developer. Для моделирования на UML предназначен, прежде всего, компонент Tau.

- Rational Statemate — решение для моделирования, проектирования и создания прототипов при быстрой разработке встроенных систем. На основе языка UML производится разработка имитаций для тестирования проектов.

IBM выпускает и другие программы, поддерживающие моделирование на языке UML.

## 5.5. Типовое проектирование

В настоящий момент типовое проектирование является одним из наиболее привлекательных для небольших и средних предприятий, во многом из-за возможности быстрого и удобного выбора решения в каталоге решений вендоров. Такие каталоги обычно включают множество решений с учетом предметной области организации, ее масштаба, потребностей и пр.

Потребность в типизации проектных решений обуславливается двумя основными факторами:

- сложность обеспечения высокого научно-технического уровня разработки при индивидуальном проектировании;
- существенное снижение затрат на проектирование при внедрении типовой системы.

Если предприятие собирается воспользоваться методом типового проектирования, то оно должно удовлетворять некоторым условиям. Так, управление этим предприятием должно быть как можно более прозрачным и осуществляться на основе единых правил, положений и документов. При этом сама структура управления организации должна быть типичной для всех отделов, а технические средства должны быть стандартизированы.

Для выбора конкретного типового проектного решения могут быть использованы различные критерии, среди которых:

- срок разработки ИС;
- финансовые возможности предприятия;
- текущее состояние ИТ-инфраструктуры;

- текущие ИС предприятия;
- используемое ПО;
- квалификация персонала и т. п.

Типовое проектирование не предполагает формирования сложных артефактов и документов, описывающих архитектуру предприятия, бизнес-процессы, информационные системы. Типовой характер проектирования предполагает, что для успешной реализации проекта достаточно высокоуровневого описания. К примеру, при типовом проектировании может быть построена упрощенная диаграмма компонентов типового решения, на основании которой будет производиться выбор типового решения и его дальнейшая адаптация и настройка. На рис. 5.20 представлен пример такой диаграммы, выполненный при помощи языка UML.

Главной целью создания подобной диаграммы является получение общего представления у проектной команды и основных заинтересованных сторон. При этом внедряемое проектное решение может не описываться, поскольку поставщик решения обычно обладает всей необходимой документацией. Гораздо большее значение играет сбор бизнес-требований, обучение бизнес-пользователей и последующее развертывание. Основные артефакты проектирования формируются проектной командой непосредственно для облегчения этих этапов, тогда как основная документация остается стандартной (но адаптированной к изменениям реализуемого решения относительно базового решения).

Существует несколько видов типовых проектных решений (ТПР), классификация которых основана на уровне декомпозиции:

- элементные ТПР;
- подсистемные ТПР;
- объектные ТПР.

Для каждого вида типового проектного решения применяется соответствующий метод типового проектирования. Достоинства и недостатки каждого метода приведены в табл. 5.6.

Таблица 5.6

#### Методы типового проектирования

Метод	Достоинства и недостатки
Элементный метод типового проектирования	<p>Достоинства: применение модульного подхода к проектированию и документированию ИС</p> <p>Недостатки:</p> <ul style="list-style-type: none"> <li>— недостающие компоненты ИС придется разрабатывать вручную;</li> <li>— элементы плохо адаптируются к особенностям предприятия;</li> <li>— огромные временные затраты на доработку и интеграцию элементов</li> </ul>

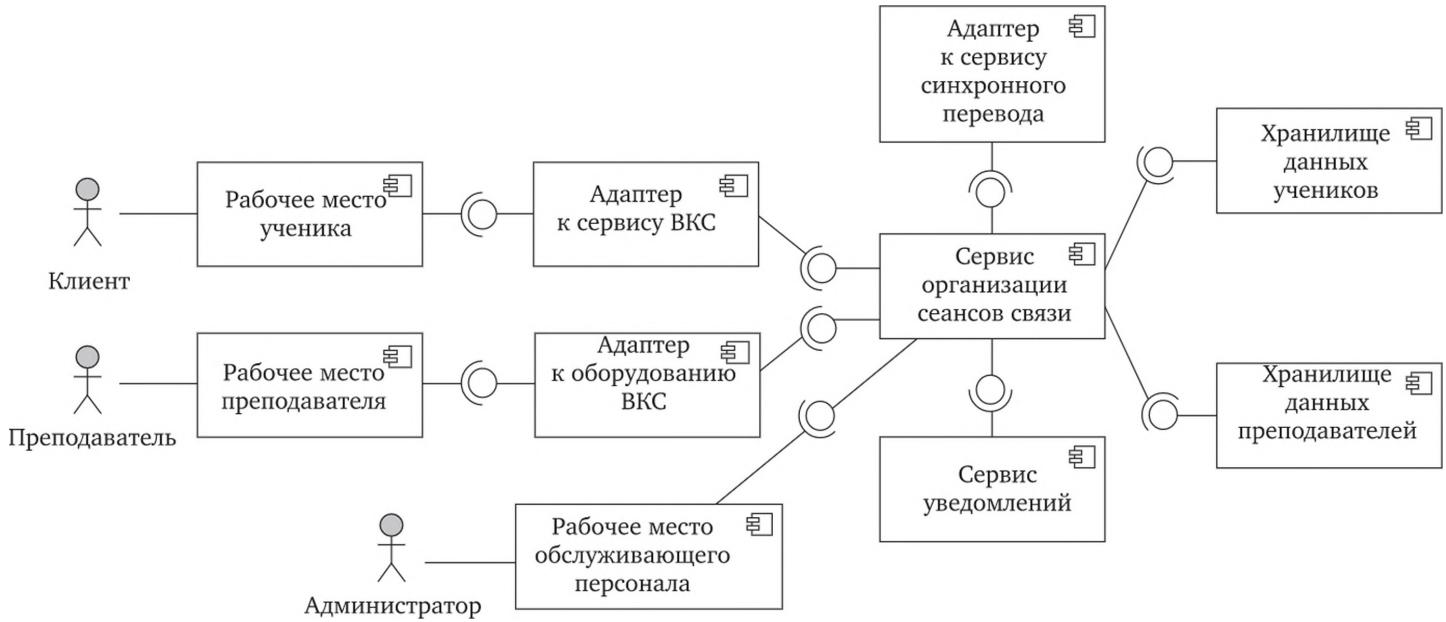


Рис. 5.20. Упрощенная диаграмма компонентов типового решения, выполненная на языке UML

Метод	Достоинства и недостатки
Подсистемный метод типового проектирования	<p>Достоинства:</p> <ul style="list-style-type: none"> <li>— применение модульного подхода к проектированию и документированию ИС;</li> <li>— возможность параметрической настройки ПО для каждого объекта управления;</li> <li>— снижение затрат на проектирование и разработку;</li> <li>— высокая степень документированности процессов информационной обработки</li> </ul> <p>Недостатки:</p> <ul style="list-style-type: none"> <li>— адаптивность подсистемного метода все еще достаточно низкая;</li> <li>— сложность представляет комплексной использование функциональных подсистем от разных производителей.</li> </ul>
Объектный метод типового проектирования	<p>Достоинства:</p> <ul style="list-style-type: none"> <li>— возможность комплексного использования всех компонентов ИС;</li> <li>— открытость архитектуры;</li> <li>— масштабируемость;</li> <li>— конфигурируемость</li> </ul> <p>Недостатки: типовой проект сложно привязать к конкретному предприятию</p>

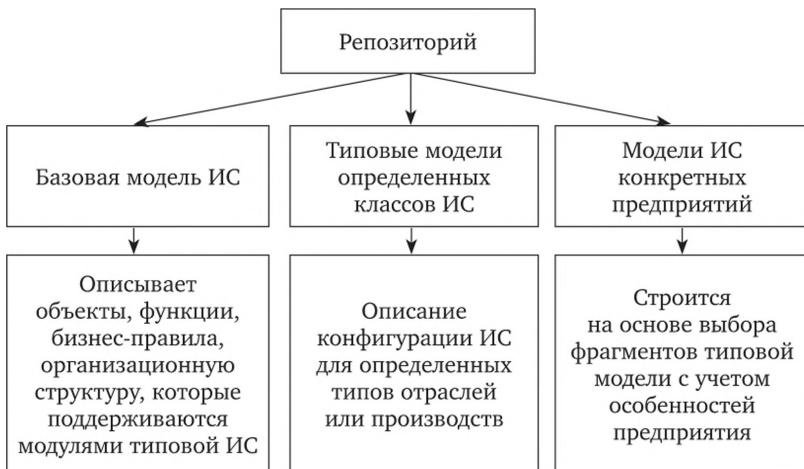
На объектном (модельном) методе следует остановиться подробнее. Этот метод сводится к адаптации структуры, состава и характеристик некоторой типовой ИС к потребностям предприятия. Выделяются два варианта данного метода.

1. Вариант 1, при котором создается модель предприятия, а затем под эту модель проектируется типовая ИС на базе уже существующей конфигурации. При этом используется специальный инструментарий.

2. Вариант 2, при котором ИС создается на основе типового варианта из репозитория. Репозиторий поставляется вместе с программным продуктом (рис. 5.21).

При этом этапы проектирования в любом случае будут достаточно типичны. К ним относят:

- формулирование и анализ требований;
- оценка и выбор ТПР, который наиболее полным образом удовлетворяет требованиям;
- построение предварительной модели ИС на базе типовых моделей;
- выбор типовой модели системы;
- определение перечня компонентов, которые будут реализованы.



*Рис. 5.21. Репозиторий в объектном методе типового проектирования*

Вслед за этим производится внедрение типовой системы. Устанавливаются основные параметры ИС, задается структура компании, определяется структура ключевых данных. Формируется перечень функций и процессов. Затем производится описание интерфейсов и отчетов, настраивается авторизация доступа.

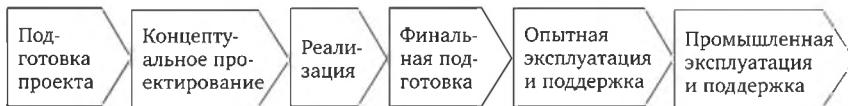
### 5.5.1. Типовое проектирование на основе Accelerated SAP

На сегодняшний день SAP является одним из крупнейших в мире поставщиков ИС и ПО, которые применяются бизнесом. Многие решения SAP очень сложны, поскольку охватывают все предприятия (к примеру, SAP ERP, охватывающая все направления деятельности компании). В связи с этим компания разработала методологию внедрения своих информационных систем. Однако эта методология может использоваться и при внедрении других ИС. Методология получила название Accelerated SAP, или сокращенно ASAP.

Методология ASAP включает шесть основных фаз, начиная с подготовки проекта и заканчивая промышленной эксплуатацией и поддержкой. На этих фазах выполняются активности, относящиеся к представленным на рис. 5.22 рабочим потокам (управление проектами, управление организационными изменениями и др.).

На фазе «Подготовка проекта» происходит начальное планирование и общая подготовка к проекту (табл. 5.7). Определяются границы проекта, формулируются цели, обозначаются приоритеты.

Следующая фаза — Концептуальное проектирование. Как следует из названия, здесь производится создание концептуального проекта ИС, который описывает все требования и модель системы (табл. 5.8).



**Управление проектами** — методология планирования, исполнения и контроля проектов

**Управление организационными изменениями** — управление заинтересованными сторонами и «выравниванием» бизнеса

**Обучение** — развитие навыков команды проекта и пользователей

**Управление данными** — миграция «унаследованных данных» и архивация

**Управление бизнес-процессами** — проектирование процессов и сборка системных решений

**Управление техническими решениями** — управление архитектурой и техническими аспектами систем

**Управление интеграцией решения** — артефакты, тестирование, процедуры, процессы

Рис. 5.22. «Дорожная карта» методологии ASAP

Таблица 5.7

**Цели, этапы и результаты фазы «Подготовка проекта»**

Цели	Этапы	Результаты
Определить цели проекта с учетом общей стратегии предприятия Рассмотреть и понять область внедрения Сформировать план и порядок внедрения Подготовить стандарты и процедуры Провести организацию проекта Определить необходимое ресурсное обеспечение	Инициация Создание стратегии/стандартов Планирование ИТ-инфраструктуры Планирование обучения пользователей	Завершена подготовка проекта Определена область действия проекта

Таблица 5.8

**Цели, этапы и результаты фазы «Концептуальное проектирование»**

Цели	Этапы	Результаты
Создать концептуальный проект и бизнес-модель ИС	Управление проектом Обучение персонала и создание документов	Концептуальный проект ИС разработан

Цели	Этапы	Результаты
Определить формат управления изменениями Пересмотреть стартовые цели/ задачи проекта Выделить предметную область проекта Пересмотреть расписание проекта и порядок внедрения Пересмотреть технический проект ИС	Первичное определение требований Управление изменениями в организации Детализация требований Расширение функциональности и разработка изменений Определение требований по безопасности Проектирование ИТ-инфраструктуры Установка ИС для разработки	Бизнес-модель ИС разработана Конфигурация, модификации и функциональность ИС определены ИС установлена для разработки

Вслед за этим наступает фаза Реализации (табл. 5.9). Концептуальный проект, созданный ранее, реализуется. Все требуемые изменения внедряются в систему. В завершении фазы производится тестирование и формирование релизной версии.

Таблица 5.9

## Цели, этапы и результаты фазы «Реализация»

Цели	Этапы	Результаты
Провести конфигурирование ИС Расширить функциональность ИС Протестировать ИС Определить стратегию сдачи ИС Сформировать и установить роли и уровни доступа	Управление проектом Управление изменениями Обучение персонала и создание документов Разработка основных элементов Конфигурирование основных элементов Инсталляция ИС для проведения тестов Планирование сдачи ИС Внедрение средств безопасности Итоговое конфигурирование Тестирование интеграции Инсталляция производственной версии Нагрузочное тестирование	ИС реализована ИС протестирована Производственная версия ИС успешно установлена

После реализации наступает фаза «Финальная подготовка». На этой фазе завершается подготовка ИС к промышленной эксплуатации, обучение персонала и работы по тестированию (табл. 5.10). Требуется, чтобы были разрешены все критически значимые проблемы и недостатки системы.

Таблица 5.10

## Цели, этапы и результаты фазы «Финальная подготовка»

Цели	Этапы	Результаты
Закончить работы над ИС Установить системы / процедуры поддержки конечных пользователей ИС	Управление проектом Завершение подготовки заказчика к приему ИС Завершение подготовки ИС Установка системы поддержки пользователей Сдача ИС	ИС готова к промышленной эксплуатации

И, наконец, все завершается на фазе «Опытная эксплуатация и поддержка» (табл. 5.11). Здесь производится окончательный переход к промышленному применению ИС. Совершенствуется поддержка и основные операционные виды деятельности. Фаза включает два отдельных периода: завершение проекта и усовершенствование. В период завершения проекта решаются все новые проблемы параллельно с вводом команды технической поддержки в курс дела. Происходит формальное закрытие проекта и передача формализованных знаний. В период усовершенствования решаются новые проблемы в ИС и производится ее мониторинг.

Таблица 5.11

## Цели, этапы и результаты фазы «Опытная эксплуатация и поддержка»

Цели	Этапы	Результаты
Произвести формальное закрытие проекта Обеспечить работу системы поддержки Завершить проект фактически	Управление проектом Поддержка ИС на ранних стадиях Закрытие проекта	ИС внедрена в промышленную эксплуатацию и применяется Система поддержки эффективно налажена

Таким образом, за счет выстроенной методологии управления проектами внедрения своих программных продуктов, сочетающейся с программным инструментом для ее практического применения, SAP способствует структурированию большого количества задач, которые необходимо выполнять на каждом этапе.

Среди других преимуществ данной методологии — уменьшение времени между инсталляцией и запуском системы в производство в режиме реального времени, а также создание общей модели управления проектом, которая может впоследствии применяться и для других внедряемых компанией продуктов или обновлений системы.

**ASAP и PMBoK.** Методология ASAP основывается на собственном опыте SAP в области разработки технических решений и бизнес-консалтинга, со значительным фокусом на уже проверенных мето-

диках управления проектами и инфраструктуре PMBoK. Так, пять ключевых этапов маршрутной карты SAP достаточно четко соотносятся с этапами проектного менеджмента PMBoK (рис. 5.23).

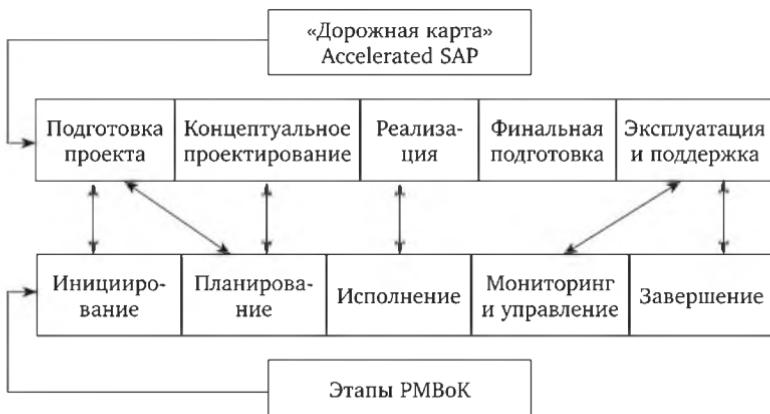


Рис. 5.23. Взаимосвязь ASAP и PMBoK

В целом же корпоративная методология Accelerated SAP предлагает комплексный инструментарий, сочетающий технологическую и управленческую части в виде трех основных компонентов:

- синхронизация бизнес-приоритетов клиента (в зависимости от индустрии) с конкретными решениями SAP через отраслевые карты бизнес-ценностей (value maps) инструмента Solution Explorer;
- карты управления проектами (маршрутные карты) с основными этапами и активностями по ним;
- проектирование, настройка, тестирование и эксплуатация решения при помощи менеджера решений SAP Solution Manager.

### 5.5.2. Типовое проектирование на основе Accelerated SAP (ASAP) 8 Agile

Для типового проектирования при реализации решений SAP может применяться методология Accelerated SAP 8 Agile, которая включает весь необходимый методологический контент: процессы, процедуры, контрольные списки, ускорители и ссылку на типовую документацию SAP. ASAP 8 Agile основана на ASAP 7 и включает материалы, необходимые проектной команде для управления проектами, решениями, организационными изменениями, а также для обучения, осуществления концептуального проектирования, реализации, тестирования, планирования и внедрения изменений.

В методологии ASAP 8 Agile используется стандартизированный подход. На рис. 5.24 представлена дорожная карта методологии, основанная на применении SCRUM-подхода.

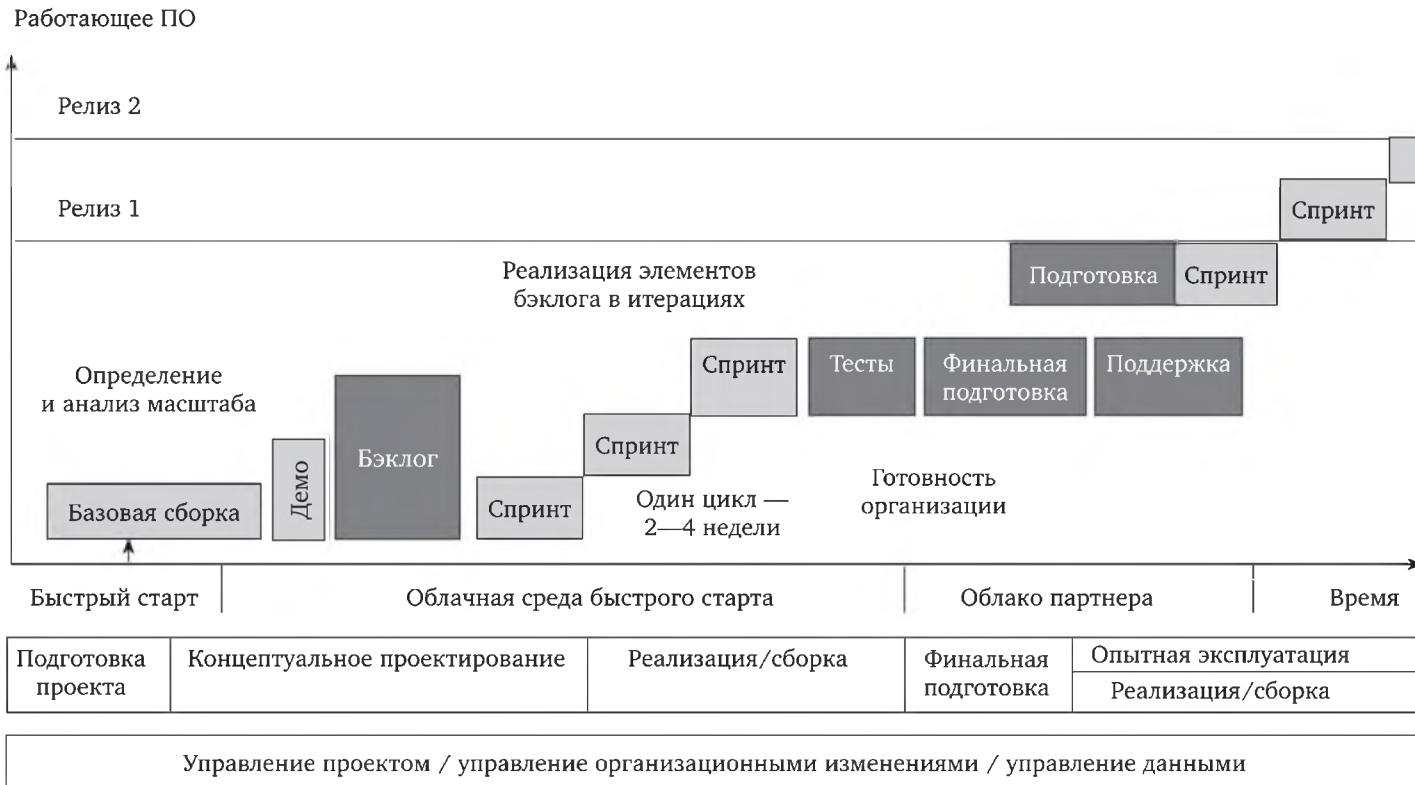


Рис. 5.24. Методология ASAP 8 Agile

В методологии ASAP 8 Agile можно выделить несколько фаз, которые будут рассмотрены далее.

**Фаза 1. Подготовка проекта.** На первой фазе осуществляется первичное планирование и подготовка к реализации типового проекта SAP. Несмотря на множество фактических отличий между двумя даже в высокой степени схожими проектами, каждый из проектов обычно имеет уникальные цели, объемы, приоритеты. Выполнение первой фазы ASAP 8 Agile позволяют эффективно выполнить первичное планирование в соответствие с уникальными особенностями проекта.

**Фаза 2. Концептуальное проектирование.** На второй фазе необходимо обеспечить общее понимание запуска решения SAP для поддержки своей деятельности. Концептуальный проект, формируемый на данной фазе, состоит из нескольких документов, которые иллюстрируют ведение бизнеса компании с использованием решения SAP. По результатам концептуального проектирования также формируют техническую документацию с выявленными требованиями.

Ключевыми документами-результатами выполнения фазы являются Baseline Build и Project Backlog, включающие требования к решению, ранжированные на основании бизнес-требований.

**Фаза 3. Реализация.** Целью фазы реализации является выполнение требований, включенных в Baseline Build и Project Backlog. В рамках ASAP 8 Agile фаза реализации предполагает выпуск нескольких конфигураций на итерационной основе, причем каждая итерация приводит к увеличению функциональных возможностей. Для сравнения, не-agile версии ASAP подразумевают, что конфигурирование решения производится в двух рабочих пакетах — в базовой конфигурации и окончательной конфигурации (основная и оставшаяся области соответственно).

Цели фазы реализации включают:

- формирование системного ландшафта;
- внедрение окончательного решения в системе разработки;
- общее тестирование решения в тестовой среде;
- выпуск решения для продуктивной системы;
- подготовка к обучению пользователей;
- подготовка к миграции данных;
- нагружочное тестирование.

**Фаза 4. Подготовка к корпоративной эксплуатации.** На данном этапе завершается подготовка к эксплуатации решения. Завершаются тестирование, обучение конечных пользователей, ввод и миграцию данных и иные переходные действия. Формируются практики управления системой и внедряются соответствующие инструментальные средства. Решаются оставшиеся критически значимые вопросы. Завершение четвертой фазы означает, что предприятие начинает опытно-промышленную эксплуатацию решения.

*Фаза 5. Продуктивная эксплуатация и поддержка.* На пятом этапе происходит переход от опытно-промышленной к продуктивной эксплуатации. Этап включает планирование и осуществление дополнительного обучения и поддержки конечных пользователей, внедрение процессов мониторинга решения, реализацию практик технической поддержки. В завершение пятого этапа процессы поддержки решения реализованы и интегрированы в деятельность предприятия, и в дальнейшем будут служить фундаментом как для регулярной поддержки, так и для непрерывного улучшения текущего состояния.

*Фаза 6. Работа.* Обеспечение работоспособности является основной целью шестой фазы.

### 5.5.3. Типовое проектирование на основе SAP Activate

SAP Activate позволяет развернуть решение в несколько этапов, за счет чего повышается степень контролируемости системы. SAP Activate включает собственную дорожную карту, определяющую порядок проектирования, реализации и внедрения типового проекта. На рис. 5.25 представлена дорожная карта SAP Activate.



Рис. 5.25. Дорожная карта методологии SAP Activate

На этапе *Обнаружения* (Discover) происходит определение ценностей и преимуществ для конкретного предприятия от использования типового решения SAP. Определяются, формулируются и согласовываются стратегия и дорожная карта проекта.

На этапе *Подготовки* (Prepare) осуществляется запуск проекта. Происходит планирование проекта, осуществляется сбор и подго-

товка проектной команды. Участники команды ознакамливаются с лучшими практиками, документацией и рекомендациями SAP.

На этапе *Исследования* (Explore) внедряется стандартное решение SAP. Формируются требования бизнеса, которые должны быть реализованы в стандартном решении.

На этапе *Реализация* (Realize) осуществляется настройка, формирование и тестирование интегрированной среды. Производится загрузка данных. Осуществляется первичное использование решения бизнес-пользователями.

На этапе *Развертывания* (Deploy) производится настройка системы. Проектная команда получает подтверждение от бизнеса о готовности к эксплуатации решения. Происходит переход бизнес-процессов организации в новый формат, предполагающий использование развернутого решения.

На этапе *Эксплуатации* (Run) обеспечивается поддержка эффективного применения решения в интересах бизнеса. Производится мониторинг бизнес-потребностей и их изменений. Анализируется потребность бизнеса в дальнейших инновациях.

## 5.6. Документирование требований

В зависимости от выбранной модели разработки могут отличаться формы документирования требований и атрибутов качества. Так, при каскадной модели разработки чаще всего применяется ГОСТ 34.602—89 «Техническое задание на создание автоматизированной системы». При использовании спиральной модели разработки (например, в рамках Rational Unified Process) требования специфицируются в виде прецедентов (Use Case model) на основе классификации требований FURPS+. В свою очередь, спецификация требований к ПО на основе SRS и IEEE 830 может применяться как при каскадной, так и при спиральной моделях разработки. На рис. 5.26 представлено соотношение форм требований с различными моделями разработки.

### 5.6.1. Документирование требований в ГОСТ 34.602—89

При использовании ГОСТ 34.602—89 формируется техническое задание (или частное техническое задание) на выполнение работ, которое и является основным документом, определяющим порядок создания, настройки, доработки и внедрения модуля системы (или системы в целом). Техническое задание систематизирует все функциональные и нефункциональные требования, тем самым обеспечивая базу для дальнейших активностей по созданию и внедрению системы.

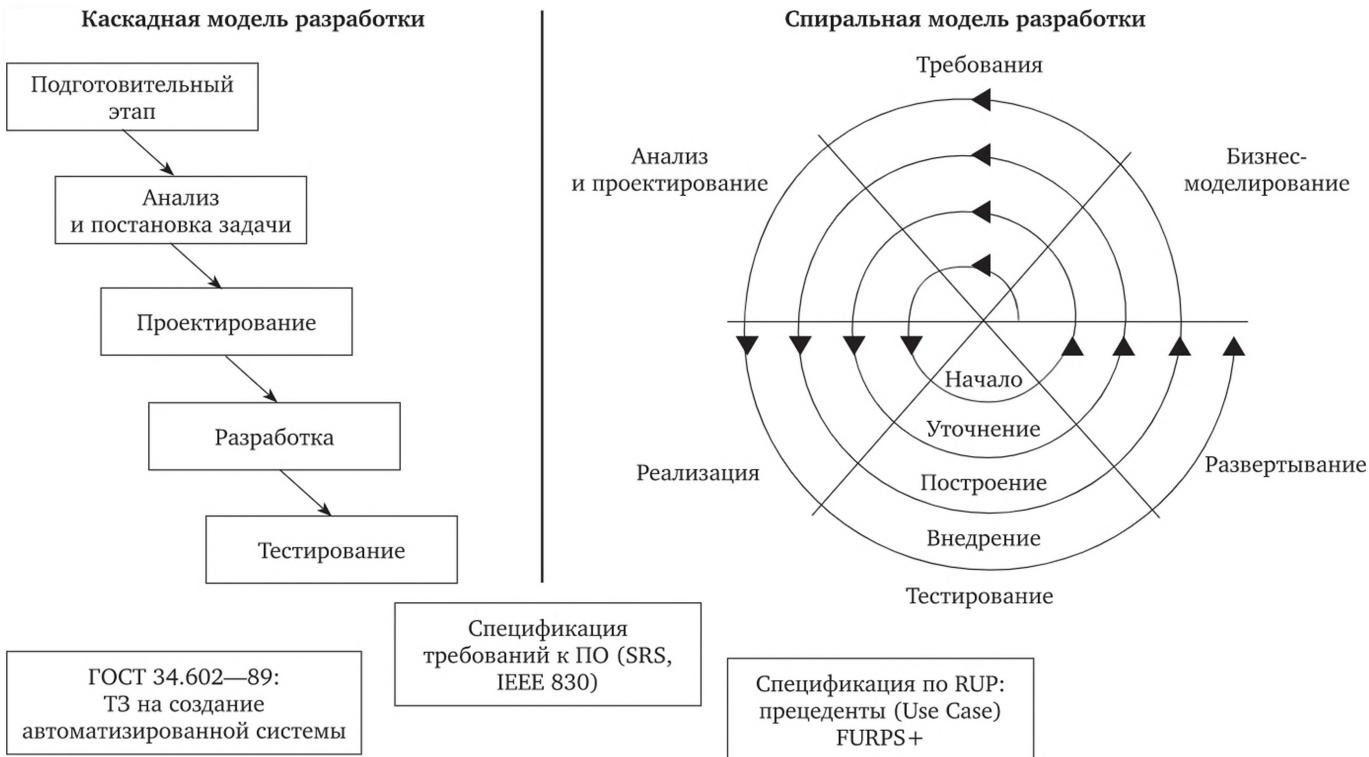


Рис. 5.26. Соотношение форм требований с различными моделями разработки

Помимо ГОСТ 34.602—89 «Техническое задание на создание автоматизированной системы» также может использоваться ГОСТ 19.201—78 «Техническое задание. Требования к содержанию и оформлению». Если ГОСТ 34.602—89 представляет собой комплекс стандартов на автоматизированные системы, то ГОСТ 19.201—78 является единой системой программной документации. Фактически при разработке информационной системы рекомендуется использовать ГОСТ 34.602—89, тогда как при разработке отдельного программного обеспечения стоит воспользоваться ГОСТ 19.201—78.

С точки зрения рассматриваемых ГОСТ автоматизированная система комплексна и включает цель, обслуживающий и эксплуатационный персонал, программные средства и комплекс технических средств. Программа же может существовать как часть автоматизированной системы, так и отдельно.

ГОСТ 34.602—89 предлагает следующую типовую структуру технического задания.

1. Общие сведения.
2. Назначение и цели создания (развития) системы.
3. Характеристики объектов автоматизации.
4. Требования к системе.
5. Состав и содержание работ по созданию системы.
6. Порядок контроля и приемки системы.
7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие.
8. Требования к документированию.
9. Источники разработки.

При этом раздел 4 «Требования к системе» детализируется более подробно в табл. 5.12.

Таблица 5.12

**Требования к системе в ГОСТ 34.602—89**

Требования к системе в целом	Требования к структуре и функционированию системы; Требования к численности и квалификации персонала; Показатели назначения; Требования к надежности; Требования безопасности; Требования к эргономике и технической эстетике; Требования к транспортабельности Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов; Требования к защите информации от несанкционированного доступа; Требования по сохранности информации при авариях; Требования к защите от влияния внешних воздействий; Требования к патентной чистоте; Требования по стандартизации и унификации; Дополнительные требования
------------------------------	---

Требования к функциям (задачам)	Перечень функций (задач) и их комплексов для каждой подсистемы, подлежащей автоматизации; Временной регламент реализации каждой функции, задачи (или комплекса задач); Требования к качеству реализации каждой функции (задачи или комплекса задач), к форме представления выходной информации, характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций, достоверности выдачи результатов; Перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.
Требования к видам обеспечения	Требования к математическому обеспечению; Требования к информационному обеспечению; Требования к лингвистическому обеспечению; Требования к программному обеспечению; Требования к техническому обеспечению; Требования к метрологическому обеспечению; Требования к организационному обеспечению; Требования к методическому обеспечению; ... и другим видам обеспечения

### 5.6.2. Документирование требований по Software Requirement Specification (SRS)

Спецификация требований по Software Requirement Specification (SRS) представляет собой законченное описание поведения программы, которую надо разработать. Спецификация требований по SRS может применяться как при каскадной, так и при спиральной моделях разработки. Стандарт IEEE 830 рекомендует следующую структуру SRS:

- введение:
  - цели,
  - соглашения о терминах,
  - предполагаемая аудитория и последовательность восприятия,
  - масштаб проекта,
  - ссылки на источники;
- общее описание:
  - видение продукта,
  - функциональность продукта,
  - классы и характеристики пользователей,
  - среда функционирования продукта (операционная среда),
  - рамки, ограничения, правила и стандарты,
  - документация для пользователей,
  - допущения и зависимости;

- функциональность системы:
  - функциональный блок X (блоков может быть несколько):
  - описание и приоритет,
  - причинно-следственные связи, алгоритмы,
  - функциональные требования;
- требования к внешним интерфейсам:
  - интерфейсы пользователей (UX),
  - программные интерфейсы,
  - интерфейсы оборудования,
  - интерфейсы связи и коммуникации;
- нефункциональные требования:
  - требования к производительности,
  - требования к сохранности (данных),
  - критерии качества программного обеспечения,
  - требования к безопасности системы;
- прочие требования:
  - приложение А: Глоссарий,
  - приложение Б: Модели процессов и предметной области и другие диаграммы,
  - приложение В: Список ключевых задач.

### **5.6.3. Документирование требований по Rational Unified Process**

В RUP пик активности в работе с требованиями приходится на начальный этап и начало этапа уточнения. В последующем требования могут корректироваться, однако серьезные изменения не предусматриваются.

Подход RUP допускает также использование классификации требований FURPS+, которая была рассмотрена в параграфе 3.2.3.

В рамках Rational Unified Process спецификация требований к программному обеспечению обычно включает Модель требований (содержащую функциональные и нефункциональные требования) и Модель прецедентов, которая формируется в виде одной или нескольких диаграмм прецедентов.

Перед формированием непосредственно Модели требований в рамках RUP может быть сформирован Документ о концепции и границах, который включает:

- бизнес-требования:
  - исходные данные,
  - возможности бизнеса,
  - бизнес-цели,
  - критерии успеха,
  - положение о концепции проекта,
  - бизнес-риски,
  - предположения и зависимости;

- рамки и ограничения проекта:
  - основные функции,
  - объем первоначально запланированной версии,
  - объем последующих версий,
  - ограничения и исключения;
- бизнес-контекст:
  - профили заинтересованных лиц,
  - приоритеты проекта,
  - особенности развертывания.

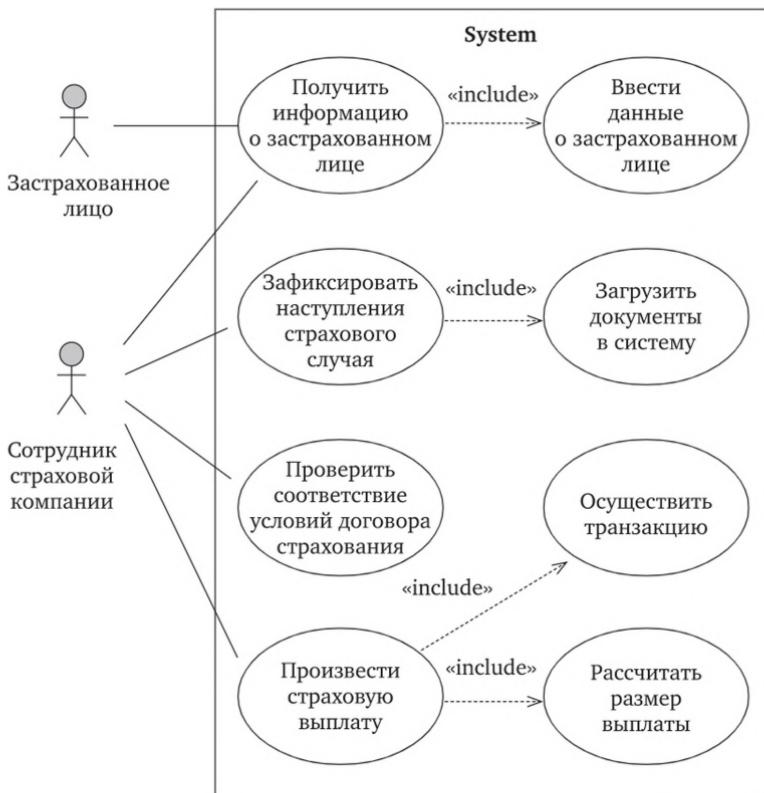


Рис. 5.26. Упрощенный пример Модели precedентов

Непосредственно документ «Модель требований» включает следующие разделы:

- обзор системы:

— обзор прецедентов. Подраздел включает названия и краткие описания прецедентов и акторов с иллюстрациями в виде диаграмм прецедентов,

— предположения и зависимости. Подраздел содержит описание основных технических возможностей, компонентов, подсистем, связанных проектов;

- описание требований:

— описание вариантов использования. Подраздел содержит описание вариантов использования и связанные с ними нефункциональные требования или ссылки на артефакты,

— специальные требования. Подраздел включает описание функциональных требований, которые не описаны как прецедент, а также описание нефункциональных требований общего характера, которые не относятся к прецедентам из прошлого подраздела,

— вспомогательная информация. Подраздел включает информацию, которая облегчает понимание документа (оглавление, приложения, описание прототипов интерфейса и пр.).

При построении Модели прецедентов на языке UML формируется диаграмма прецедентов. Диаграмма включает прецеденты, акторов, связи между ними и границы информационной системы. Прецеденты фактически отражают функциональные требования к информационной системе. На рис. 5.27 представлен упрощенный пример Модели прецедентов, выполненный в программном средстве StarUML.

#### 5.6.4. Выявление атрибутов требований

Независимо от типа модели разработки и способов документирования, в итоговых документах в том или ином виде будет представлен реестр требований. Реестр требований включает перечень требований, сгруппированных по категориям и имеющих различные атрибуты. К наиболее типичным атрибутам требований относятся следующие.

- Абсолютная ссылка — уникальные идентификатор требования, который не подлежит изменению или повторному использованию даже при перемещении, изменении или удалении требования.
- Автор требования — тот, кто выдвинул требование.
- Владелец — сотрудник или группа сотрудников, которым нужно требование и которые будут владеть им после реализации в информационной системе.
- Источник — тот, с кем нужно консультироваться при изменении требования или для получения дополнительной информации; тот, кто выдвинул требование.
- Приоритет — очередность реализации.
- Срочность — информация о том, как скоро нужно реализовать требование.
- Статус — этап жизненного цикла требования (предложено, принято, проверено, отложено, отменено и пр.).

- Сложность — оценка того, насколько сложно выполнить требование.
- Стабильность — оценка степени зрелости требования и возможности работать с ним.

При использовании подхода Rational Unified Process может быть использован набор атрибутов требований RUP, представленный в табл. 5.13.

Таблица 5.13

**Набор атрибутов требований RUP**

Атрибут	Семантика
Статус	Proposed (предложенные) — еще не утверждены, но уже обсуждаются. Approved (одобренные) — утверждены для реализации. Rejected (отклоненные) — решено не реализовывать. Incorporated (включенные) — реализованы в предыдущей версии
Полезность	Critical (критическое) — обязательно к реализации. Important (важное) — можно не реализовывать, но лучше реализовать. Useful (полезное) — можно опустить без больших потерь полезности
Трудоемкость	Оценка времени и ресурсов для реализации возможности, выраженная в человеко-часах или другими методами
Риск	Риск, связанный с добавлением этой возможности (высокий, средний, низкий)
Стабильность	Оценка вероятности того, что требование будет изменено (высокая, средняя, низкая)
Целевая версия	Версия продукта, в которой требование должно быть реализовано

Значения атрибута «Приоритет требований» может формироваться на основе набора MoSCoW. Набор MoSCoW включает следующие значения приоритета требования:

- Must Have — обязательное, фундаментальное требование к информационной системе;
- Should Have — важное требование, которое может быть опущено;
- Could Have — требование, которое реализуется только при наличии времени и средств;
- Won't Have — требование, которое возможно будет реализовано в следующих версиях системы.

Определение приоритета также возможно на основе обычной пятибалльной шкалы, где 1 — критическое требование, без реализации которого невозможен запуск системы в эксплуатацию,

а 5 — возможно полезное требование, полезность которого требует проверки или незначительна.

## **Контрольные вопросы и задания**

1. Каковы основные принципы проектирования ИС?
2. Что такое каноническое проектирование?
3. Что такое проектирование с использованием CASE-средств?
4. Что такое Rational Unified Process?
5. Какие выделяют рабочие потоки в рамках RUP?
6. Что такое типовое проектирование?
7. Назовите особенности методологий Accelerated SAP, SAP Activate, SAP 8 Agile.
8. Зачем применяются методологии проектирования ИС?
9. Какие этапы были пройдены в развитии методологий разработки ИС?
10. Как выбрать подходящую методологию разработки ИС?
11. Какие CASE-средства используются для проектирования?
12. Каковы основные принципы и методы документирования требований?

## **Рекомендуемая литература**

1. Грекул, В. И. Проектирование информационных систем // Национальный открытый университет «ИНТУИТ», 2012. — URL: <http://publications.hse.ru/books/74833061> (дата обращения: 17.10.2020).
2. Вендрев, А. М. Проектирование программного обеспечения экономических информационных систем : учебник для вузов / А. М. Вендрев. — 2-е изд., перераб. и доп. — Москва : Финансы и статистика, 2005.
3. Абдиев, Н. М. Корпоративные информационные системы управления : учебник / Н. М. Абдиев — Москва : ИНФРА-М, 2014.
4. Избачков, Ю. С. Информационные системы : учебник для вузов / Ю. С. Избачков [и др.]. — 3-е изд. — Санкт-Петербург : Питер, 2011.

## Тема 6

# РАЗРАБОТКА

---

Эта тема посвящена вопросам разработки ИС. Тема описывает традиционные и гибкие подходы к разработке информационных систем. Произведено их сравнение, приведено краткое описание традиционных подходов. Подробно рассмотрены гибкие подходы к разработке и их ключевые особенности. Представлены CASE-средства разработки. Тема также подробно описывает тестирование, включая роли, задачи, виды, правила и результаты тестирования. Произведен обзор инструментальных средств тестирования.

После изучения данной темы студент будет:

**знатъ**

- основы традиционных подходов к разработке;
- гибкие методологии и подходы к разработке;
- разницу между жесткими и гибкими подходами к разработке ИС;
- основные CASE-средства разработки;
- роли и результаты тестирования;
- задачи тестирования;
- основные виды тестирования;
- правила и рекомендации проведения тестирования;
- инструментальные средства тестирования;

**уметь**

- в зависимости от контекста производить обоснованный выбор методологии или подхода к разработке;
- определять роли и задачи тестирования;
- обоснованно определять основные виды тестирования и ожидаемые результаты;
- формировать правила и рекомендации для проведения тестирования;

**владеть навыками**

- использования CASE-средств разработки;
  - использования инструментальных средств тестирования.
- 

### 6.1. Жесткие и гибкие подходы к разработке ИС

Все существующие методологии и подходы разработки можно условно разделить на две группы. К первой группе будут относиться жесткие подходы, ко второй — гибкие. Жесткие подходы характеризуются формализованными и четко обозначенными границами и целями. В рамках таких подходов социальные факторы считаются стабильными, повторяемыми и предсказуемыми. Гибкие подходы,

напротив, фокусируются на динамических факторах, к которым относят поведение людей, их действия — таким образом, пристальное внимание уделяется людям. Кроме того, гибкие подходы принимают во внимание неопределенность, непредсказуемость и хаос.

Для понимания, к какой группе относится какая-либо методология, следует последовательно ответить на ряд вопросов.

1. Определение: есть ли знание и согласие о том, что является проблемой?

2. Граница: позволяет ли определение проблемы понять, что является проблемой, а что — не является?

3. Разделение: исчезает ли потребность в вопросе «Что не является проблемой?» после проведения границы?

4. Ответственность: позволяют ли определение и разделение проблемы сказать, кто должен быть вовлечен в решение проблемы, а кто — нет?

5. Информация: если есть ясность в понимании проблемы и ответственности, то знаем ли мы, какая информация потребуется для решения проблемы?

6. Описание: известно ли, чем будет и чем не будет создаваемое решение, если предыдущие критерии были удовлетворены?

Если на все вопросы был дан ответ «Да», то речь идет о жесткой проблеме и, соответственно, жесткой методологии. Если хотя бы на один вопрос дан ответ «Нет», то речь идет о неопределенной проблеме и, соответственно, гибкой и нечетко определенной методологии.

## 6.2. Традиционные подходы к разработке

Методологии, относящиеся к традиционным (или жестким), могут быть классифицированы по нескольким группам:

- методологии, ориентированные на данные;
- процессно-ориентированные методологии;
- смешанные методологии;
- объектно-ориентированные методологии.

**Методологии, ориентированные на данные (Data-Oriented),** основное внимание уделяют данным и учету их потоков. Например, к данной группе относится *методология информационного инжиниринга (IE)*, разработанная Клайвом Финкельштейном и Джеймсом Мартином. В рамках методологии IE процесс создания информационной системы представляется как процесс построения и развития моделей. Соответственно, процесс включает в себя следующие шаги.

1. Информационное стратегическое планирование, которое включает в себя анализ ситуации, анализ потребностей, определение архитектуры и сбор информационно-стратегического плана.

2. Анализ области, который включает в себя анализ атрибутов, анализ взаимодействия, анализ существующих систем, проектирование, планирование.

3. Разработка и программирование системы: логическая разработка и техническая разработка.

4. Создание и реализация: производство, оценка и реализация системы, а также ее защита.

**Процессно-ориентированные** (*Process-Oriented*) методологии отличаются от предыдущей группы, поскольку акцент делается на уже существующих процессах и попытке использовать их для создания новой ИС. В качестве примера можно привести методологию Yourdon Systems Method (YSM), которая разделяет процесс разработки ИС на четыре фазы, каждая из которых состоит из нескольких шагов. Методология выделяет следующие фазы:

- технико-экономическое обоснование;
- моделирование информационных потоков предприятия (с построением DFD и ERD);
- построение основной модели предприятия и ИС;
- непосредственная разработка ИС.

**Смешанные** (*Blended*) методологии рассматривают одновременно и процессы, и данные. В качестве примера можно привести методологию Structured Systems Analysis and Design (SSADM), которая включает в себя ряд шагов.

Шаг 0: осуществимость.

Шаг 1: исследование текущего окружения.

Шаг 2: варианты бизнес-системы.

Шаг 3: определение требований.

Шаг 4: варианты технической системы.

Шаг 5: логическое проектирование.

Шаг 6: физическое проектирование.

**Объектно-ориентированные** (*Object-Oriented*) методологии основаны на создании объектов, которые описывают предметную область. Объектно-ориентированные методологии фокусируют внимание на том, что система должна делать, вместо того, чтобы рассматривать как система должна это делать. Методологии этой группы подразумевают, что стадия разработки не будет начата до тех пор, пока все имеющиеся объекты не будут тщательно проанализированы и спроектированы.

Пример объектно-ориентированной методологии — Rational Unified Process (RUP), которая состоит из нескольких циклически повторяющихся в рамках ЖЦ стадий. К таким стадиям относятся:

- начальная стадия (*Inception*);
- уточнение (*Elaboration*);
- построение (*Construction*);
- внедрение (*Transition*).

Каждая стадия включает набор рабочих потоков — шесть основных и три вспомогательных:

- бизнес-моделирование;
- требования;
- анализ и проектирование;
- реализация;
- тестирование;
- развертывание;
- управление конфигурацией и изменениями;
- управление проектом;
- среда.

При этом каждая стадия завершается созданием некоего рабочего релиза ИС. В целом же RUP базируется на наборе «строительных блоков». Эти блоки описывают, что должно быть произведено, требования к необходимым навыкам и пошаговое объяснение того, как цели разработки могут быть достигнуты.

Методология RUP была подробно рассмотрена ранее.

#### 6.2.1. Microsoft Solution Framework (MSF)

Руководствуясь стремлением снизить риски от неуспешных ИТ проектов, компания «Майкрософт» выпустила пакет руководств по эффективному проектированию, разработке, внедрению и сопровождению решений, основанных на ее технологиях. Знания, заложенные в эти руководства, базировались на опыте «Майкрософта» в реализации проектов крупного масштаба по разработке и сопровождению программного обеспечения, на опыте консультантов «Майкрософта» и достижениях IT-индустрии в целом. Эти технологии представлены в виде двух взаимосвязанных и дополняющих друг друга областей знаний: Microsoft Solutions Framework (MSF) и Microsoft Operations Framework (MOF).

MSF состоит из двух моделей и трех дисциплин. Они подробно описаны в пяти документах.

MSF содержит:

- модели:
  - модель проектной группы,
  - модель процессов;
- дисциплины:
  - дисциплина управление проектами,
  - дисциплина управление рисками,
  - дисциплина управление подготовкой.

Первая версия MSF появилась в 1994 г. Текущая версия — MSF 4.0 была представлена в 2005 г. Необходимо отметить, что сейчас существуют отдельные версии MSF, которые могут применяться для поддержки как традиционных, так и гибких методологий.

### **6.2.2. Rapid Application Development (RAD)**

При использовании спиральной модели жизненного цикла часто применяется методология быстрой разработки приложений RAD (Rapid Application Development). Базовые принципы методологии таковы:

- приложения разрабатываются в несколько итераций;
- полное завершение работ для перехода от одного этапа к другому не требуется;
- пользователи в обязательном порядке вовлекаются в процесс создания ИС;
- для выяснения потребностей пользователей используется принцип прототипирования;
- в обязательном порядке используются CASE-средства, отвечающие за целостность проекта;
- желательным является использование средств управления конфигурацией, которые облегчают процедуру внесения изменений в проект (и сопровождение системы, когда она уже создана);
- в обязательном порядке используется генератор кода;
- разработка и тестирование проекта происходят параллельно;
- разработка осуществляется небольшой командой профессионалов;
- велика потребность в грамотном разделении полномочий и руководстве.

ЖЦПО в методологии RAD состоит из четырех фаз:

- 1) анализа и планирования требований;
- 2) проектирования;
- 3) разработки;
- 4) внедрения.

Первая фаза, *анализ и планирование требований*, служит для определения функций, которые будет выполнять система. Выделяются наиболее важные функции, которые будут прорабатываться в первую очередь. Происходит понимание (и дальнейшее описание) информационных потребностей пользователей. Требования к системе ставятся самими пользователями под руководством экспертов из числа разработчиков. В процессе постановки требований ставятся ограничения проекта по масштабу, времени, средствам. Результат первой фазы — список функций с расстановкой приоритетов, а также предварительные модели ИС (функциональные и информационные).

Следующая фаза, *проектирование*, подразумевает участие части пользователей в проектировании системы. При помощи специальных CASE-средств удается оперативно получить качественные прототипы. За счет работы пользователей с ними уточняются и дополняются требования к системе. При необходимости корректируется функциональная модель системы. Точно так же при необходимости

создается рабочий прототип для каждого элементарного процесса (экран, диалог, отчет), устраниющий неясности. На данной фазе разграничивается доступ к данным и определяется набор требуемой документации.

Завершением фазы проектирования становится определение числа функциональных элементов. Как правило, принимается решение о разбиении ИС на подсистемы, каждая из которых будет реализована командой разработчиков в приемлемое время (60—90 дней для методологии RAD). С помощью CASE-средств проект распределяется между командами. Результатом фазы проектирования становятся:

- общая информационная модель системы;
- функциональные модели системы в целом и модели подсистем, реализуемых отдельными командами разработчиков;
- точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
- прототипы экранов, отчетов, диалогов.

Уже на этом этапе требуется использовать те и только те CASE-средства, которые в дальнейшем будут использоваться для создания ИС. Требование к унификации подобных средств вызвано проблемой, свойственной традиционным подходам: нередко возникало искажение данных при переходе с этапа на этап. Соответственно, единый набор CASE-средств позволяет избежать искажений.

Методология RAD рассматривает каждый прототип как часть будущей ИС. Прототипы не утилизируются после решения возникающих вопросов. За счет подобного шага на каждом следующем этапе используется все более полная, детализованная и удобная информация.

Третья фаза, *разработка*, предполагает быструю разработку приложения. В несколько итераций строится реальная система, основанная на моделях предыдущих фаз и нефункциональных требований. Часть программного кода генерируется в автоматическом режиме из репозиториев CASE-средств. В рамках фазы разработки конечные пользователи оценивают результат и вносят в него изменения, если по каким-то причинам система вдруг не соответствует их требованиям. Тестирование системы производится параллельно с разработкой.

После того, как команды разработчиков заканчивают работы над подсистемами, происходит интеграция подсистем. Возникает полный программный код, производится тестирование совместной работы подсистем, а затем тестируется вся система. Физическое проектирование системы завершается следующими действиями:

- определяется необходимость распределения данных;
- производится анализ использования данных;
- производится физическое проектирование базы данных;

- определяются требования к аппаратным ресурсам;
- определяются способы увеличения производительности;
- завершается разработка документации проекта.

Результат фазы построения — готовая система, которая соответствует требованиям конечных пользователей и заказчика в целом.

И, наконец, наступает время четвертой фазы — *внедрения*. На фазе внедрения специалисты-разработчики обучают пользователей. Производятся организационные изменения. При этом работа со старой ИС продолжается до тех пор, пока новая система не будет полностью внедрена. Поскольку фаза разработки достаточно короткая, планирование фазы внедрения должно начинаться заранее. Обычно подготовка начинается еще на фазе планирования системы.

Схема из четырех фаз не является обязательной. Работа в рамках методологии RAD зависит от начальных условий. Так, может разрабатываться совершенно новая система; может иметься готовый анализ организации; может существовать прототип ИС, который надо интегрировать с разрабатываемой системой и т. п.

Методология RAD может быть эффективнее всего применена для небольших проектов. В рамках данной методологии практически невозможно создать сложные расчетные программы, операционные системы и большие проекты (с объемом в сотни тысяч строк кода). Методология не применяется также для создания приложений, не имеющих интерфейсов, с помощью которых можно наглядно показать конечным пользователям логику системы. RAD не применяется в системах, которые связаны с жизнью и здоровьем людей, поскольку по умолчанию считается, что первые несколько версий не будут полностью работоспособными.

### 6.3. Гибкие методологии и подходы

В переводе с английского *Agile* означает «гибкий» — именно поэтому такое название получило семейство методологий разработки, которые не содержат четких инструкций и «лучших практик», и представляют собой сборник основных принципов и ценностей в предметной области разработки ПО. К тому же сам процесс гибкой разработки является адаптивным по отношению к постоянно изменяющимся условиям, что достигается разработкой за короткие итерации, после каждой из которых происходит пересмотр требований и, в случае необходимости, — изменение практик коммуникаций и работы команды. Кроме этого, есть еще несколько не менее важных и требующих рассмотрения идей Agile:

- приоритет взаимодействия людей над процессами и традиционными инструментами управления;
- приоритет получения работающего продукта над исчерпывающей всеобъемлющей документацией;

- приоритет сотрудничества с потребителями (заказчиком) над формальными вопросами контрактов;
- приоритет быстрого реагирования на изменения над неоступным следованием плану.

Исходя из этих ценностей, у команд, использующих Agile-методологию, почти не встречается одинаковых практик документирования или координации, так как гибкая методология предполагает самоорганизацию, самостоятельное определение объема, содержания и ограничений каждого элемента управления разработкой. Со временем своего появления в 1990-х гг. Agile-практики получили распространение во множестве компаний, в то время как первоначально они позиционировались для управления ИТ-проектами.

В качестве одного из важнейших элементов Agile-подхода к разработке выступают **пользовательские истории** (*user stories*). Это описанные одним или несколькими предложениями основные аспекты функциональности системы, необходимые для выполнения пользователем своей работы. В достаточно сжатом виде они должны отвечать на вопросы: «Кто?», «Что?» и «Почему?» и могут создаваться как менеджером проекта по результатам обследования как один из форматов фиксирования требований, так и разработчиками для выражения нефункциональных требований (например, к безопасности, производительности, качеству решения).

## Пример

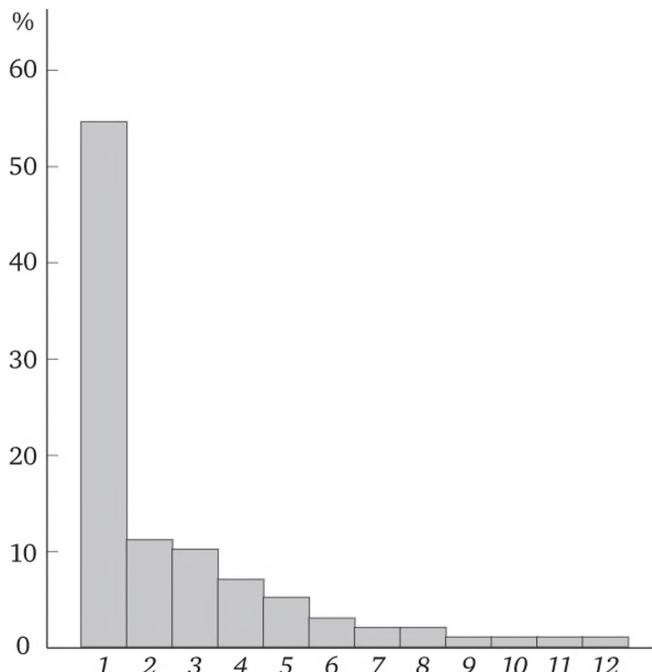
### Отдельные требования, представленные в виде user stories.

- Как ответственный за проведение рассылок я должен вести мониторинг адресатов по разным категориям.
  - Как ответственный за проведение рассылок я бы хотел давать права администратора моим подчиненным в случае необходимости.
  - Как ответственный за проведение рассылок я хочу видеть, сколько процентов пользователей проходят по ссылке в письме.
  - Как ответственный за проведение рассылок я должен иметь возможность отправить адресатам, не открывшим письмо в течение двух недель, дополнительное письмо.
  - Как ответственный за проведение рассылок я бы хотел создавать их план в перспективе на две недели вперед.
  - Как ответственному за проведение рассылок мне необходим удобный формат визуального представления и выгрузки статистических данных.
  - Как ответственный за проведение рассылок я должен знать источник получения контактных данных каждого адресата.
  - Как ответственный за проведение рассылок я должен получать недублированные данные, т. е. одному пользователю может соответствовать только один текущий электронный адрес.

### Итоговая user story на «техническом» языке.

Как ответственному за проведение рассылок пользователю необходимо обеспечить следующую функциональность:

- иметь конфигурацию статусов каждого адресата: адрес электронной почты, реакцию на каждую из проведенных рассылок (красная иконка для открывших письмо и не прореагировавших на него, желтая — для неоткрытых сообщений, зеленая — для прочитавших и выполнивших требование в рассылке действие);
  - иметь несколько уровней прав доступа к рассылкам (на просмотр статистики, на выгрузку e-mail адресов, на отправку рассылок, ...);
  - создать персональные ссылки в отправляемых сообщениях, через которые будет осуществлен подсчет числа пользователей, пришедших на страничку портала по результатам рассылки;
  - ...
- 



*Рис. 6.1. Статистика популярности гибких методологий за 2013 г.<sup>1</sup>:*  
 1 — Scrum; 2 — Scrum/XP Hybrid; 3 — Custom Hybrid; 4 — Scrumban;  
 5 — Kanban; 6 — Lean; 7 — FDD; 8 — Other; 9 — XP; 10 — AgileUP;  
 11 — Agile Modelling; 12 — DSDM

В то же время гибкая методология почти не ограничивает команду проекта: так, если необходим больший уровень детализации технического задания, спецификаций или другой документации, то можно его написать. Если необходимы дополнительные приемо-

<sup>1</sup> Масленников В. В., Крылов В. Г. Процессно-стоимостное управление бизнесом // Cfin.ru: Корпоративный менеджмент, библиотека управления. URL: [http://www.cfin.ru/management/practice/manage\\_business.shtml](http://www.cfin.ru/management/practice/manage_business.shtml) (дата обращения: 30.05.2020).

ные критерии — можно их составить, если нужны прототипы — можно их сформировать в виде макетов (мокапов, *mockups*), в виде блок-схем или в виде реальной модели прототипа. Уровень детализации и формализации всегда зависит от специфики системы и проектной команды, от их компетенций и опыта реализации подобных проектов.

В 2013 г. было проведено восьмое ежегодное исследование «State of Agile», которое выявило наиболее распространенные гибкие методологии управления проектами, используемые в сфере создания ИС и ПО (рис. 6.1).

В 2013 г. методология Scrum оказалась самой популярной, на нее пришлось больше половины проектов. Остальные гибкие методологии используются в 11 % случаев и менее. К ним относятся (XP), Feature Driven Development (FDD), Kanban, Lean, Scrumban (комбинация Kanban и Scrum).

### 6.3.1. Scrum

В переводе с английского языка Scrum означает «схватка». Впервые подобная методология была введена в употребление профессорами японского Hitotsubashi University в их статье «Новая игра в производстве продукта» (*The New Product Development Game*), написанной в 1986 г. В ней разработка продукта была сравнена со схваткой вокруг мяча в регби — приемом, называемым «скрам». Тем не менее авторами концепции считаются разработчики из США, Кен Швабер и Джек Сазерленд, впервые определившие и описавшие основополагающие принципы Scrum — «гибкой» методологии управления проектами (чаще всего применяющейся в области разработки ПО).

По своей сути Scrum — это набор принципов разработки, которые позволяют представлять конечному пользователю (и заказчику) действующий продукт или прототип, обладающий новыми функциями и возможностями с наивысшим приоритетом. Работа в этом случае проводится в четко фиксированные недельные (в среднем от 1 до 6 недель, чаще от 2 до 4) итерации (спринты) (рис. 6.2). Ожидаемые результаты спринта определяются командой под руководством Scrum-мастера в самом его начале и НЕ могут изменяться до завершения. К ним под руководством Scrum-мастера (по сути руководителя проекта) и product-мастера (заказчика, владельца проекта) создается список задач проекта (*project backlog*), в котором содержатся упорядоченные по важности требования. Из этого журнала владелец проекта выбирает наиболее важную функциональность, которая затем переходит в Журнал пожеланий спринта (*sprint backlog*). Вошедшая в него функциональность, подлежащая реализации в рамках спринта, разбивается на задачи, время выполнения которых оценивается командой.

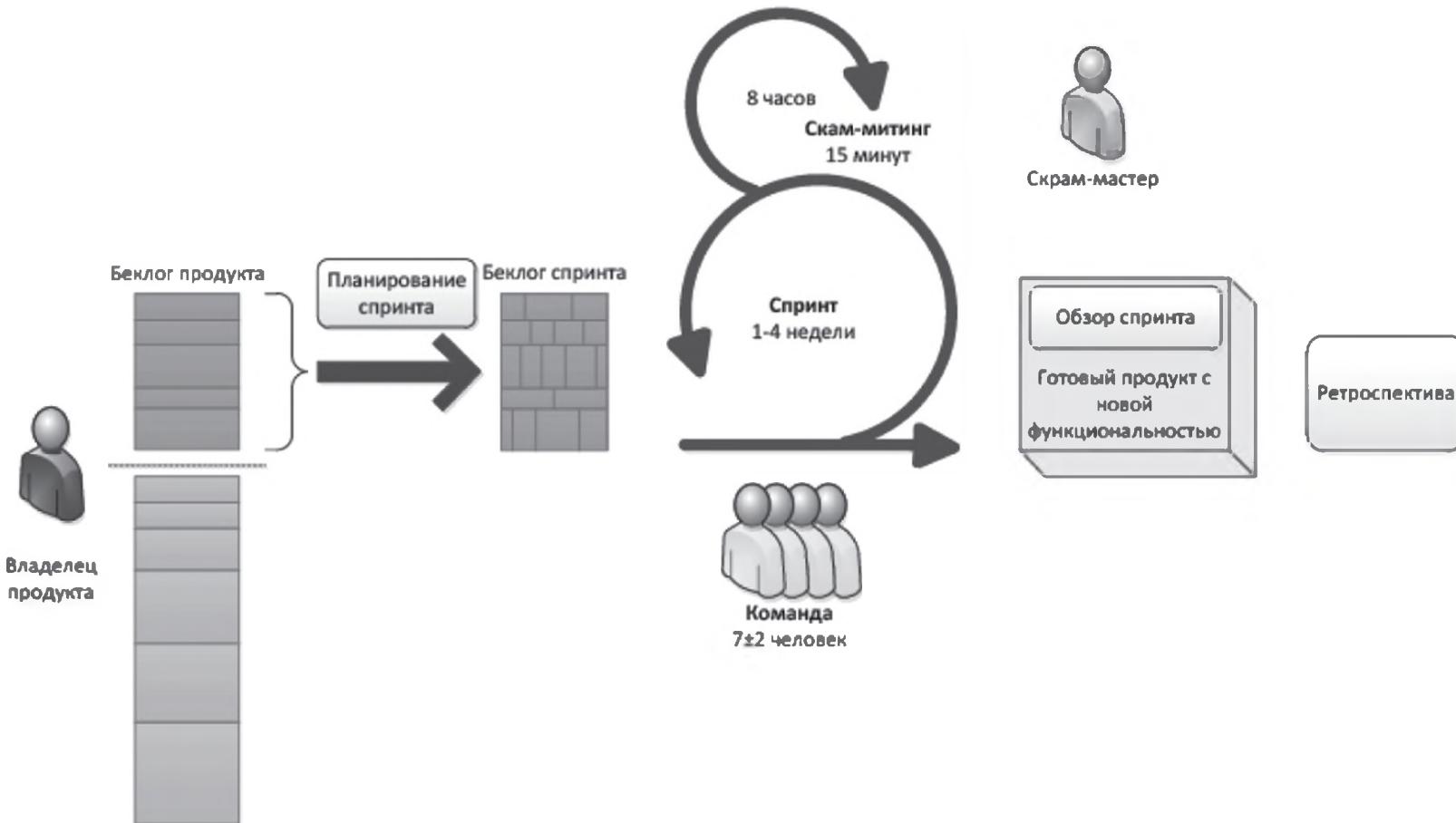


Рис. 6.2. Общая схема Scrum<sup>1</sup>

<sup>1</sup> Вольфсон Б. Гибкие методологии разработки. 2012. URL: <http://adm-lib.ru/books/10/Gibkie-metodologii.pdf> (дата обращения: 30.05.2020).

- Работу над проектом ведут небольшие команды с пересекающимися функциями, причем каждая команда включает всех нужных специалистов. Скрам-мастер (руководитель проекта) отвечает за создание продуктивной атмосферы и соблюдение процессов. Обязанность скрам-мастера — постоянное нахождение с проектной командой для понимания текущей ситуации.

- Требования заказчиков разбиваются на небольшие функциональные части, которые как можно меньше зависят друг от друга и ориентируются на конечных пользователей. Результатом становится получение списка задач для продукта, элементы которого ранжируются в зависимости от важности и оценивают их объем.

- За требования и приоритеты отвечает владелец продукта. Чаще всего им является представитель заказчика, который и замыкает на себе всех заинтересованных в проекте лиц.

- Короткие и фиксированные итерации длительностью от 1 до 4 недель — основа работы проекта. Такие итерации называют спринтами. В конце каждого спринга разработчики предоставляют законченную функциональность — инкремент продукта. При желании его даже можно вывести на рынок.

- У каждой команды проекта есть своя скорость. В зависимости от скорости команда выбирает себе задачи на итерацию, которая не изменяется по времени. На ежедневном скрам-митинге синхронизируется работа команд и обсуждаются возникшие вопросы/проблемы. По мере работы каждая команда берет в работу элементы списка задач в зависимости от приоритета и собственных возможностей.

- Обзор спринга — логическое завершение итерации. Во время обзора спринга получают отклик от владельца продукта и проводят ретроспективу спринга для оптимизации собственных процессов. Также на обзоре спринга владелец продукта может изменить требования и приоритеты, а также запустить новый спринг.

Однако при более глубоком рассмотрении Scrum имеет очень важные основы: методология предполагает циклический и очень активный процесс, минимизирующий неопределенность требований за счет коротких итераций и предоставляющий возможность детального прототипирования. Долгие, нестандартные, динамические проекты с расплывчатым представлением о конечном ожидаемом результате и постоянной сменой приоритетов не являются проблемой для Scrum, и именно поэтому он часто применяется на первых этапах, а затем в целях экономии средств и ресурсов производится переход на другую методологию. Соответственно, Scrum может применяться практически любым типом организаций — от стартапов с их высокой неопределенностью до госзаказчиков с частым изменением требований и частой необходимостью демонстрации текущей версии для отчета о результатах.

При такой быстрой и динамичной работе требуется минимизировать трудозатраты на коммуникацию между разработчиками. Для этого, как правило, используются видеоконференции или личное общение, которое в идеале происходит с использованием маркерной/грифельной доски для заметок. Для сплочения команды первые спринты нередко проводят совместно. Также рекомендуется общение с командой после работы в неформальной обстановке. Существуют особые игры, которые проводятся в команде для построения доверительных отношений. Среди таких игр наибольшей популярностью пользуются «Scrum из ада», «XP игра по планированию», «Производство бумажных шляп», «Холст доверия» и т. п.

Иногда, в силу различных причин, проектные команды могут быть распределены территориально. В таком случае применяется политика «амбассадоров», когда один член команды временно переходит к другой группе, чтобы перенять их опыт, знания и навыки.

Предполагается, что команда работает самоорганизованно, поэтому механизмы и инструменты контроля за разработкой отсутствуют. Даже скрам-мастер не имеет возможности для широкого контроля деятельности команд. Низкая формализация и отсутствие всякой документации приводят к тому, что скрам-мастер может контролировать процесс только на митингах и во время обзоров. При наличии потребности в контроле иногда применяются практики визуального менеджмента, распространенные в методологии Kanban.

С одной стороны, в силу того, что релизы выпускаются часто, минимизируется вероятность ошибок за счет постоянной обратной связи с потребителями при демонстрации новой функциональности, а также ежедневных (!) встреч команды проекта. С другой стороны, увеличиваются временные (и денежные) затраты на проведение презентаций, а также на исправление выявленных проблем и осуществляемый ретроспективный анализ.

Главный недостаток Scrum-подхода — отсутствие практик по организации тестирования и разработки. Для этого также допустимо использовать подходы из других методологий. Часто Scrum «скрещивают» с экстремальным программированием, и возникает особый подход «XP Driven by Scrum».

При принятии решения о переходе на Scrum необходимо принимать во внимание возможное сопротивление со стороны некоторых специалистов, не привыкших к подобным подходам или предпочитающих индивидуальную работу, так как именно в этой методологии крайне важна сплоченность команды, ее уровень ответственности и кросс-функциональность для ежедневного поиска оптимальных путей совершенствования продукта. Также высока зависимость успеха Scrum-проектов от мастерства и компетенций Scrum-мастера, который должен обладать достаточным авторитетом и умением

управлять командой для успешной организации выполнения работ и контроля проекта на всех этапах.

### 6.3.2. Kanban

Применительно к ИТ Kanban следует рассматривать не как полноценную методологию, а как инструмент визуального менеджмента. В основе Kanban лежит несколько принципов:

- 1) визуализируйте процесс;
- 2) ограничивайте объем незавершенной работы;
- 3) фокусируйтесь на процессе;
- 4) совершенствуйте процесс.

Доска — главный инструмент в Kanban. На доске размещаются задачи, которые ранжируются в зависимости от этапа выполнения. Как и в Scrum, в Kanban ежедневно проводятся встречи. На встречах команда обсуждает проблемы, узкие места, а также корректирует информацию на доске. Еженедельно проводится ретроспектива. Благодаря такому менеджменту Kanban позволяет решить проблему отсутствия контроля за разработкой ПО.

Как и Scrum, Kanban уделяет внимание чатам, видеоконференциям, неформальному общению, встречам сотрудников, ретроспективам и т. п. Констатируется, что самостоятельные команды по возможности должны располагаться в одном офисе. Каждая команда имеет свои задачи и свою доску, а также проводит свои встречи каждую неделю. При этом каждый член команды имеет «товарища» из другой команды, с которым он обменивается опытом и информацией о ходе работ. Это позволяет установить доверительные отношения в команде, поскольку каждые две недели товарищ меняется.

Важно, однако, понимать, что Kanban нельзя использовать для управления ЖЦ ИС и ПО. Практики визуального менеджмента следует применять вместе с другими процессами разработки, чтобы сделать его управляемым, гибким и удобным. Практики контроля качества и создания качественного продукта должны быть заложены в процесс его создания изначально, поскольку Kanban не предлагает самостоятельных методик и практик для этого.

### 6.3.3. eXtreme Programming

Основная особенность методологии экстремального программирования (*eXtreme Programming*) состоит в ее эффективности в условиях неопределенных или нечетких требований. Популярность данного подхода увеличивалась по мере роста числа разработчиков, недовольных традиционным подходом с множеством формальных требований и постоянно готовивших документацию, собиравших информацию о показателях проекта. Напротив, новая методология предлагала простой дизайн, переработку (рефакторинг) кода для контроля затрат, постоянное присутствие заказчика, разработку

через тесты и другие аспекты, выгодно отличавшие ее от классических методологий.

Экстремальное программирование опирается на итеративность, взаимодействие внутри команды, взаимодействие с заказчиком, смелость и готовность рисковать. Подход основывается на 12 ключевых практиках.

1. Игра в планирование (*Planning Game*).
2. Тестирование (*Testing*). Практики разработки на основе тестов (*Test-Driven Development, TDD*), т. е. написание тестов до реализации.
3. Повторное кодирование (*Refactoring*).
4. Коллективное владение кодом (*Collective Code Ownership*).
5. Простота разработки (*Simple Design*).
6. Парное программирование (*Pair Programming*).
7. Небольшие релизы (*Small Release*).
8. Постоянная интеграция (*Continuous Integration*).
9. Заказчик в команде разработки (*On-site Customer*).
10. Стандарты кодирования (*Coding Standards*).
11. Метафора (*Metaphor*).
12. 40-часовая рабочая неделя (*40-hour Work Week*).

Для работы в подобных условиях необходима постоянная связь с заказчиком, которая обеспечивается полноценным участием в работе проектной команды квалифицированного, находящегося в курсе проекта представителя заказчика.

При разработке в большинстве случаев выбираются наиболее простые методы исходя из того принципа, что легче в будущем вносить дополнения в базовую версию, чем перестраивать усложненную (хотя, разумеется, бывают исключения, когда изначально учет многих факторов в системе может помочь избежать в дальнейшем многих сложностей). **Принцип простоты** также важен и в интерфейсном решении, когда более интуитивный пользовательский дизайн интерфейса обладает исключительно необходимыми (но не излишними!) функциями. Однако предварительное детальное проектирование интерфейса согласно исходной версии методологии не осуществляется, он создается лишь по мере работы команды в течение проекта.

Принцип коллективной работы по написанию кода и внесению изменений в настройки имеет два аспекта. В первую очередь, это **коллективное владение кодом**, когда любой участник команды разработчиков может внести изменения благодаря единым правилам оформления кода и использованию стандартов для ПО. С другой стороны, применяется метод **парного программирования**, в соответствии с которым двое разработчиков используют один компьютер, чередуя написание кода и доработку настроек, реализацию требований и прочие вопросы.

Данный подход по-своему решает проблему контроля и координации работ. В отличие от Kanban и Scrum, в XP производится подробное планирование работ и техника небольших релизов. Коммуникационные проблемы решаются за счет формализованных правил, где определяется уровень доступности команды. XP также использует видеоконференции и различные коммуникационные программные продукты, при помощи которых можно получить доступ к рабочему столу. Время от времени проводятся реальные встречи участников команды.

В данном подходе особо остро встает проблема недопонимания между членами команды. Проблема решается за счет парного программирования, коллективного владения кодом, метафор и использования стандартов кодирования. Контроль качества осуществляется представитель заказчика, который является членом команды разработки. Если заказчик территориально удален, то контроль осуществляется через онлайн-мессенджеры.

При наличии ряда достоинств, XP содержит важное ограничение. Применение данного подхода невозможно, если в команде разработчиков более пяти человек. Кроме того, практики менеджмента отсутствуют, а потому заказчик вынужден самостоительно ставить рамки и контролировать качество готового продукта.

#### 6.3.4. Feature Driven Development (FDD)

Еще одна известная гибкая методология разработки — это Future Driven Development (FDD), которая совмещает преимущества XP и SCRUM с модельно-ориентированными подходами. Подходу FDD удалось решить главную проблему предшественников: возможность работы только с небольшими командами. FDD, напротив, используется для работы над большими проектами. Первое использование произошло в целях создания банковской системы высокого уровня.

В рамках методологии FDD все проектные работы требуется разделить на пять процессов (рис. 6.3). Во время нулевой итерации (первичной проектной деятельности в терминах FDD) происходит реализация трех процессов. Разрабатывается общая модель, составляется иерархический список необходимых функций, производится оценка функций с точки зрения трудозатрат.

После выделения основных функций для клиента начинается процесс их реализации, который происходит итеративно (рис. 6.4). Ранее созданная модель системы может корректироваться и уточняться. В конце каждой итерации результатом становится работающая функция, которая уже установлена на промышленный сервер. Особый акцент в FDD делается на проектировании модели системы, в связи с чем высоко значение так называемого «моделирования в цвете» при помощи UML.

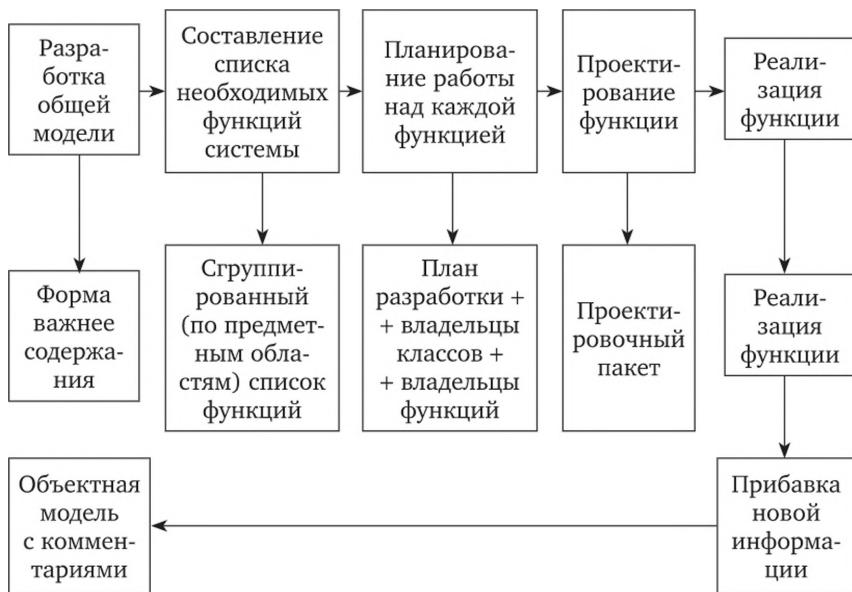


Рис. 6.3. Схема методологии FDD

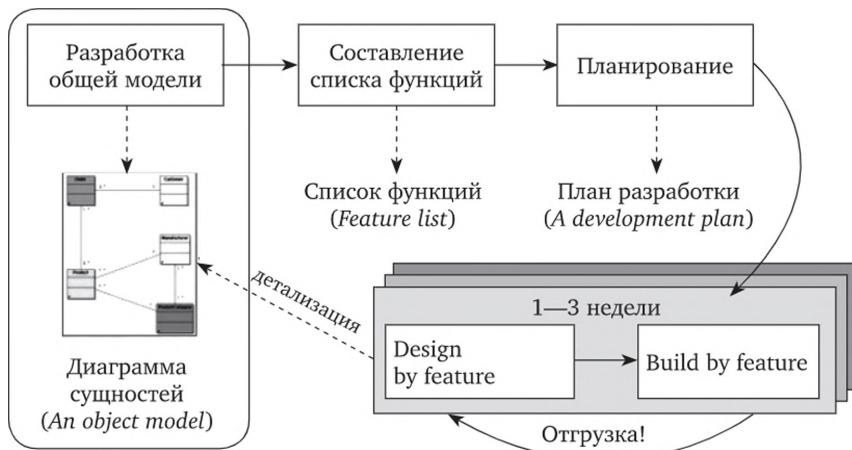


Рис. 6.4. Схема процесса FDD

На этапе моделирования создается несколько конкурирующих моделей за авторством разных команд, независимых друг от друга. После этого совместными усилиями строится общая модель, которая содержит лучшие решения из каждой модели. За каждую отдельную итерацию рассматривается только один кусок модели, который предварительно уточняется и детализируется (производятся так называемые дизайн-сессии).

Процесс разработки контролируется за счет плана, который составляется заранее. Помимо этого, для контроля определяют ответственных лиц. Методология FDD предполагает, что результат должен быть видимым, иначе менеджеру будет тяжело понять, на каком этапе находятся работы по проекту.

Наличие ответственных лиц снижает коммуникационную проблему внутри команды и позволяет эффективно работать географически распределенным группам. Ответственным за последовательность выполнения считается ведущий разработчик. Он планирует работы, на основании которых ответственные по классам самоорганизуются с целью получить требуемый результат.

FDD уделяет пристальное внимание проверкам на этапе разработки. Использование проверок несет в себе следующие преимущества:

- обмен опытом и культурой разработки;
- соответствие стандартам.

Несмотря на очевидные плюсы, методология FDD обладает ярко выраженными недостатками. Прежде всего, в методологии отсутствует практика проведения встреч между разработчиками. В результате страдает доверие к коллегам.

Другой недостаток — наличие правил только по вопросам управления разработкой и проектированием. Для управления требованиями потребуется использовать иной подход. Сравнение подходов Scrum, FDD и XP приведено на рис. 6.5.

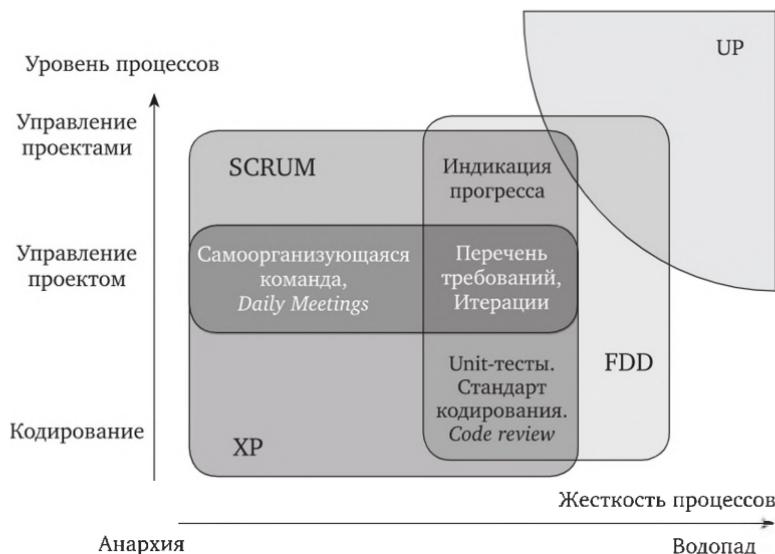


Рис. 6.5. Сравнение методологий Scrum, FDD, XP

### **6.3.5. Domain-Driven Design (DDD)**

Термином «Domain Driven Design» обозначают разработку, которая основана на некоей модели предметной области. DDD — это своеобразное развитие ставшего уже относительно традиционным подхода MDD. DDD развивалась (и развивается) в Agile-сообществе и, соответственно, рассчитана на гибкое использование. По сути своей, DDD — это MDD, в которой, однако, делается акцент на реализуемости модели предметной области в программном коде.

DDD содержит несколько принципов, первый из которых — итеративное построение модели предметной области. Модель должна использовать единый язык, понятный обеим сторонам — и заказчику, и разработчику. Для этого в DDD применяется единый язык разработки и написания модели. Модель системы представляется сразу в нескольких проекциях, описанных ниже.

- Метафора системы, представляющая собой неформальное описание наиболее крупных элементов модели.
- Диаграмма классов, показывающая структуру объектов и методов. При построении диаграммы классов изначально строятся укрупненные классы без излишней детализации. Далее по ходу разработки добавляются те или иные подробности, включая иерархию типов, атрибуты, методы, структуру и т. п.
- Диаграмма учета, которая является учетной моделью системы. Под учетом в DDD понимается отражение потоков обобщенных ресурсов.
- Диаграмма состояний документов, отражающая документооборот. При помощи диаграммы состояний документов происходит связывание учетной и объектной моделей. Документы в данной диаграмме имеют всего один атрибут, в качестве которого выступает состояние. Состояние определяет этап обработки документа, его отражение в учете, ответственность за документ, права на совершение действий с документом. Построение такой диаграммы базируется на модели бизнес-процессов компании.

Затем все проекции связываются воедино при помощи методов переходов на основе такой Agile-практики, как регулярный рефакторинг.

Модель предметной области в DDD создается для того, чтобы в дальнейшем на ее основе создать структуру хранения данных, программный код бизнес-логики и дизайн GUI.

## **6.4. CASE-средства (Computer Aided Software Engineering)**

Методология — важный, но не единственный залог создания качественной информационной системы. При работе с любой методологией важно правильно пользоваться CASE-средствами, при по-

моши которых будут автоматизированы процессы для всех этапов ЖЦИС. CASE-средства, которые применяются сегодня, позволяют существенно облегчить разработку ИС. Различные средства включают в себя инструменты для анализа, документирования, автоматизации и т. п. Грамотно подобранное CASE-средство покрывает весь жизненный цикл ИС или ПО, а также позволяет сэкономить время и ресурсы при их разработке, создании и внедрении.

CASE-средства значительно облегчают разработку ИС на самых сложных этапах. Так, благодаря CASE-средствам на этапе разработки удается добиться получения слаженных технических решений, а также минимизировать усилия по созданию проектной документации. CASE-средства позволяют осуществлять визуальное представление информации разработчикам за счет построения различных диаграмм, удобных для восприятия. А графические средства моделирования позволяют наглядно рассматривать ИС в текущем состоянии и изменять ее в соответствии с требованиями, целями и ограничениями.

Всякое CASE-средство имеет специфическую архитектуру. Такая архитектура включает в себя репозиторий, верификатор диаграмм, сервисы, документатора, администратора и графический редактор диаграмм (рис. 6.6).

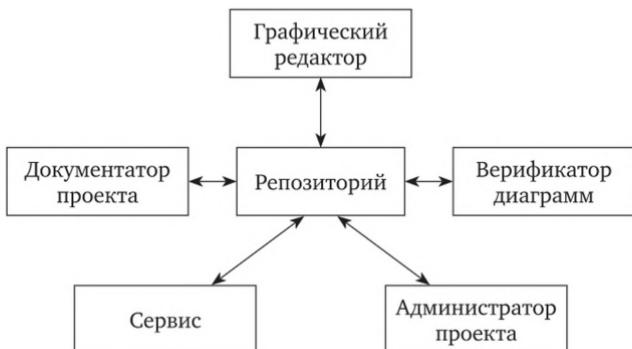


Рис. 6.6. Архитектура CASE-средства

Существуют различные классификации современных CASE-средств. Первая и наиболее распространенная классификация производится по **типам**. В рамках данной классификации выделяются:

- средства анализа (Upper CASE), которые используются в процессах анализа и построения модели предметной области;
- средства анализа и проектирования (Middle CASE), рассчитанные на поддержку основных методологий проектирования; средства данного типа используются, когда необходимо облегчить работу по созданию проектных спецификаций;

- средства проектирования/разработки/реинжиниринга (Low CASE), обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций.

Вторая, не менее полезная, классификация CASE-средств — это классификация по категориям. Такая классификация позволяет определить, в какой степени средство интегрировано по выполняемым функциям. В зависимости от категории, CASE-средство будет включать в себя:

- отдельные средства, которые применяются для решения небольших обособленных задач (*tools*);
- набор средств, которые охватывают большую часть ЖЦИС, причем набор является лишь частично интегрированным (так называемый *toolkit*);
- полностью интегрированные средства с общим репозиторием, которые используются для поддержки всего ЖЦИС.

Далее будут приведены конкретные примеры CASE-средств с различной специализацией:

- графические средства, применяющиеся для анализа и проектирования; с их помощью обеспечивается создание и редактирование иерархических диаграмм (к примеру, DFD, ERD), которые обращают модель информационной системы:
  - CA Erwin Data Modeler (ранее Erwin), ныне являющаяся частью объединенного пакета CA Erwin Modeling Suite,
  - Casewise — средство моделирования, позволяющее анализировать, визуализировать, документировать и оптимизировать бизнес-процессы предприятия и информационной системы,
  - Enterprise Architect, Umbrello, ArgoUML, Umodel и другие средства, применяемые в целях моделирования на основе UML,
  - XTG Data Modeler (XTGDM), которое используется при структурном моделировании данных;
- средства планирования и управления проектом (Microsoft Project и др.);
  - CASE-средства, применяющиеся для хранения версий проекта, его диаграмм, компонентов, синхронизации информации, контроля полноты и непротиворечивости метаданных. Общее название данной группы ПО — системы управления версиями (VCS, англ. *Version Control System*). К данной группе относятся ClearCase, VSS, CVS, Subversion, средства конфигурационного управления PVCS<sup>1</sup>, StarTeam, SVN;
    - Visual Studio, Eclipse, NetBeans — средства для разработки приложений, включающие генерацию программного кода;

---

<sup>1</sup> Включающие PVCS Version Manager, PVCS Tracker, PVCS Configuration Builder и PVCS Notify.

- IBM Rational SoDA — средство документирования;
- средства тестирования:
  - JIRA — система для отслеживания неполадок в случае, когда над проектом работает команда разработчиков,
  - HP Quality Center — полная интегрированная система, при помощи которой обеспечивается контроль качества на всех этапах создания ИС и ПО,
  - Bugzilla — система для отслеживания ошибок с встроенным web-интерфейсом,
  - MantisBT — система отслеживания ошибок, распространяющаяся свободно,
  - Seapine TestTrack Pro — управление жизненным циклом приложений за счет работы с качеством,
  - Testlink — свободно распространяющаяся система управления тестами,
  - Centercode — платформа для разработки бета-версий различных приложений,
  - Microsoft TFS/VSTS. Team Foundation Server (сокр. TFS) — комплексный продукт, который объединяет систему управления версиями, средства сбора данных, инструменты построения отчетов, средства отслеживания проектного статуса и изменений,
  - IBM Rational ClearQuest, представляющее собой ПО для управления ЖЦ приложений, которое содержит средства для гибкого управления дефектами и изменениями, настройки процессов, создания отчетов и мониторинга жизненного цикла.

Отдельно требуется рассмотреть интегрированные CASE-средства. Такие средства объединяют несколько различных компонентов, использующих единый репозиторий. Репозиторий является особой базой, в которой хранится состояние разрабатываемой ИС на различные моменты времени, информация о составных частях ИС, используемых объектах и их взаимосвязях. За счет общего словаря происходит синхронизация объектов, которые вводятся в различных диаграммах.

Приведем примеры интегрированных CASE-средств.

- SAP PowerDesigner — набор инструментов, при помощи которых можно создавать бизнес-приложения. Набор включает средства моделирования бизнес-процессов, а также предоставляет разработчикам широкие возможности для работы с баз данных — как в концептуальном, так и в физическом виде.
  - CA Erwin Modeling Suite — набор средств для моделирования баз данных, бизнес-процессов и компонентов ПО. Включает в себя:
    - Erwin Process Modeler;
    - Erwin Data Model Validator;
    - CA Erwin Model Manager;
    - CA Erwin Data Modeler;

- CA Erwin Data Profiler;
- CA Erwin Model Navigator.

- Embarcadero ER/Studio Enterprise — комплексный набор средств и инструментов для управления данными, который позволяет создавать и обслуживать БД и хранилища данных. В состав входят:

- ER/Studio Data Architect;
- ER/Studio Business Architect;
- ER/Studio Software Architect;
- ER/Studio Repository;
- ER/Studio Team Server;
- MetaWizard.

- IBM Rational Software — платформа для разработки гибкого ПО. Платформа основана на принципе коллективного сотрудничества. Продукт включает следующие модули: Rational Team Concert, Rational Doors, Rational Quality Manager, Rational ClearQuest, Rational Software Architect, Rational System Architect, Rational ClearCase, Rational Rhapsody Developer, Rational Requisite Pro и т. д.

- IBM Rational Requisite Pro — средство для управления требованиями в рамках всего ЖЦ проекта. С его помощью выполняется определение, систематизация и отслеживание требований, а также их изменения. Продукт используется при наличии потребности сократить риски проекта и повысить качество создаваемого продукта.

- IBM Rational Software Architect — это среда для разработки и моделирования, которая использует UML для проектирования архитектуры приложений и веб-сервисов. Используется для создания устойчивых приложений и веб-служб за счет архитектурного анализа кода. Поддерживает возможности MDD.

- IBM System Architect — это инструмент для создания архитектуры предприятия, который используется и бизнес-, и технологическим департаментами для моделирования бизнес-процессов и систем, приложений, баз данных. Также применяется для оперативной поддержки. Следует отметить, что в 2015 г. IBM System Architect был продан фирме UNICOM.

- SILVERRUN Professional & Enterprise Series (Grandite, Business Modeling and Software Engineering Company) включает Enterprise Process Architecture/Data Flow Diagram Modeling, Enterprise Data Architecture/ Entity Relationship Diagram Modeling, Database and Application Interfaces, Network Team Repository.

## 6.5. Тестирование

Тестирование — важная часть процесса создания ИС. Тестирование служит для оценки и проверки качества системы. Общие методы тестирования включают в себя:

- выявление и документирование ошибок;

- подтверждение допущений, сделанных в спецификациях и (или) проектной модели;
- проверку надлежащего функционирования ИС/ПО;
- проверку надлежащей реализации требований к ИС/ПО.

Следует четко понимать, что тестирование не ставит своей целью доказать отсутствие ошибок в программе. Точно так же нельзя использовать тестирование для презентации работоспособности программы. Главная цель тестирования — обнаружение слабостей программного продукта.

Основа подхода в тестировании — это деструктивность. Следует задаваться вопросом: как можно вывести из строя эту программу? При этом подход должен эффективно демонстрировать недостатки и слабости программы. Нельзя впадать в крайность и исходить из попытки любой ценой вывести систему из строя.

**Основные ошибки тестирования.** Нередко констатируется, что тестирование — это очень дорогостоящий процесс, на который тратится 30—50 % всех средств, затраченных на разработку. У таких высоких затрат на тестирование существуют вполне определенные причины, устранив которые можно существенно снизить стоимость тестирования.

1. *Тестирование проводится эпизодически и только на поздних стадиях жизненного цикла.* В результате стоимость отладки увеличивается, поскольку ошибка, допущенная в ранних итерациях, может свести на нет разработку поздних модулей.

2. *Удобство тестирования не принимается во внимание.* Тестирование — затратный по времени и финансам процесс. Тестирование нужно в обязательном порядке автоматизировать при помощи специальных инструментальных средств.

3. *Планирование тестов осуществляется без связи с ИС.* На деле требуется планирование тестов с учетом опыта предыдущих итераций.

Стоимость создания и поддержки ИС может быть снижена только за счет регулярного тестирования в процессе разработки и стремления к получению качественного продукта.

Важно понимать, что в случае итерационного подхода тестирование должно проводиться в рамках каждой итерации. В случае последовательных каскадных подходов тестирование должно осуществляться на каждом этапе, когда его проведение возможно. При этом в обоих случаях тестирование включает в себя множество групп тестов, которые проверяют как функциональные, так и нефункциональные аспекты деятельности ИС.

**Миссия тестирования.** Миссия тестирования определяет задачи и продукты тестирования. Соответственно, правильно выбранная миссия тестирования служит основой для успешного и эффективного тестирования в дальнейшем.

Миссия — это простое утверждение, которого следует придерживаться для фокусирования внимания на общей цели. Миссия тестирования может быть сформулирована следующим образом:

- найти максимальное число дефектов;
- определить существующие проблемы;
- определить имеющиеся риски для качества;
- подтвердить соответствие заданному стандарту;
- проверить соответствие продукта спецификациям и требованиям;
- и т. п.

Если миссия тестирования отсутствует, то команда тестировщиков скажет о своей работе «мы тестируем все» или «мы просто занимаемся тестированием», отразив тем самым непонимание своих целей и задач. В результате тесты могут быть просто подогнаны под контекст проекта.

#### 6.5.1. Роли и результаты тестирования

**Роли в тестировании.** Проведение тестирования предполагает наличие в команде нескольких ролей. Как правило, роли оказываются совмещёнными, т. е. один человек имеет 1—2 роли. Роли в тестировании таковы:

- *тест-менеджер (Test Manager)* производит управленческий контроль тестирования;
- *тест-дизайнер (Test Designer)* разрабатывает план тестирования, а также оценивает эффективность тестирования;
- *тестировщик (Tester)* производит тесты и фиксирует результаты, а также восстанавливает тесты и систему после сбоев;
- *администратор тестовой группы (Test Administrator)* управляет тестировщиками;
- *разработчик тестов (Test Developer)* занимается созданием конкретных тестов, которые в дальнейшем будут использованы тестировщиками.

Наличие всех ролей в каждом проекте по созданию ИС не обязательно, но приведенный выше список так или иначе отражается на практике каждый раз при проведении грамотного тестирования.

**Этапы и результаты тестирования.** В первую очередь следует рассмотреть *План тестирования*. План тестирования определяет сроки, миссию и цели для каждого конкретного тестового цикла. Цель тестирования должна быть выражена как можно более простыми словами.

Как правило, указывают 1—2 цели для каждого тестирования. Цели не должны конфликтовать друг с другом и затруднять тестирование.

Следующий этап — это создание *Комплекта тестов*. Комплект тестов представляет собой набор взаимосвязанных тестов, которые,

будучи исполнены вместе, дают оценку той или иной области тестирования. Комплект тестов включает в себя как *Тестовые сценарии*, так и *Тестовые данные*.

*Тестовые сценарии* представляют собой пошаговые инструкции по выполнению теста. Тестовые сценарии выполняются вручную или автоматически. Как правило, современные инструментальные средства автоматизируют тестовые сценарии. *Тестовые данные* — это входная информация и ожидаемый результат тестирования.

*Прецеденты тестирования* содержат ответ на вопрос: что мы собираемся тестировать? Если в проекте использовалось визуальное моделирование на языке UML, то прецеденты тестирования строятся на основе Диаграммы прецедентов. Прецедент тестирования — это абстрактный мотив проведения теста, который нужен для понимания целей тестирования и общей логики. Во многих случаях данный этап может быть пропущен. Его применение обязательно только для сложных ИС, в которых принципиально важны качество и безопасность.

И, наконец, существует еще один важный документ — *Модель рабочей нагрузки*. Этот документ содержит описание типичных и исключительных нагрузочных условий ИС. Система должна выдерживать рабочие условия, указанные в данном документе. Его создание обязательно, поскольку он служит основой для проведения тестирования на соответствие нефункциональным требованиям.

Результат тестирования — это *Дефект*. Дефект может рассматриваться как объективно существующий запрос на изменение. Все выявленные дефекты собираются в Сводную оценку результатов тестирования для последующего устранения (в рамках следующей версии или следующей итерации).

По завершении каждой итерации тестирования формируется Сводная оценка результатов тестирования. Документ также создается для каждого выпуска программы. Сводная оценка результатов тестирования содержит дефекты и аномалии, найденные при тестировании, а также производит их ранжирование. Сводная оценка результатов тестирования содержит объективную оценку качества тестируемого модуля, ПО или ИС.

### 6.5.2. Задачи тестирования

В общем виде можно выделить шесть задач тестирования:

- 1) определение миссии тестирования;
- 2) проверка подхода к тестированию;
- 3) проведение «теста на герметичность» (проверка стабильности ИС);
- 4) тестирование и оценка;
- 5) достижение приемлемого результата миссии;
- 6) совершенствование средств и методов тестирования.

Задачи тестирования должны выполняться на каждой итерации или в каждом тестовом блоке. Далее они будут рассмотрены подробнее.

**Задача 1. Определение миссии тестирования.** В самом начале тестирования требуется понять, на чем будет сделан акцент при тестировании. Выбранный акцент должен быть согласован с заинтересованными сторонами. Как правило, миссия тестирования концентрируется на следующих аспектах:

- формулирование целей тестирования (пример миссии — «Проверить устойчивость ИС при пиковых нагрузках»);
- тестирование использования системой аппаратных ресурсов (пример миссии — «Проверить использование системой аппаратных ресурсов»);
- определение границ тестирования и области тестирования (пример миссии — «Проверить работоспособность финансового модуля ИС»);
- определение подхода и инструментов автоматизации (пример миссии — «Протестировать модуль ИС при помощи инструмента X»).

Для каждой итерации и для каждого этапа тестирования миссия изменяется.

**Задача 2. Проверка подхода к тестированию.** На первом этапе решения этой задачи формируется подборка методов и инструментов тестирования. Далее следует убедиться, что их применение:

- 1) облегчает процесс тестирования;
- 2) является полезным.

Соответственно, целью становится получение представления об ограничениях всех выбранных методов и инструментов. Если метод или инструмент оказывается хорошим, то он используется. Если нет, начинается поиск альтернативы или от использования инструмента просто отказываются.

Для выполнения данной задачи выполняется следующее:

- проверка работоспособности подхода или инструмента и его способности предоставлять полезные результаты; выполняется на ранних стадиях;
- развертывание базовой инфраструктуры, обеспечивающей применение подхода к тестированию и (или) отдельно взятого инструмента;
- определение области, границ, пределов и ограничений для каждого инструмента и метода;
- обеспечение содействия разработчиков в осуществлении выбранного подхода или в применении выбранных инструментов.

Чаще всего данная задача выполняется на ранних этапах тестирования. В дальнейшем она может реализовываться только в случае появления потребности в новых инструментах или при вынужденном изменении подхода.

**Задача 3. Проведение теста на герметичность.** Решение этой задачи позволяет понять, является ли программный продукт (ИС, текущая версия ПО/ИС) стабильным. Проверка на герметичность осуществляется еще до начала тестирования. Иногда эту задачу называют «санитарной проверкой» или «приемом к тестированию». Проведение теста на герметичность очень важно, поскольку позволяет не тратить время и средства на бесполезные усилия.

Для решения данной задачи выполняются следующие виды деятельности:

- оценка стабильности и тестируемости: нужно понять, может ли текущая версия быть установлена, загружена и запущена;
- получение исходной информации: нужно узнать, какие компоненты были добавлены (в случае итеративной разработки) или какие модули ИС существуют;
- принятие решения о стабильности или нестабильности.

Проведение теста на герметичность — быстрый процесс, который позволяет в ряде случаев существенно снизить затраты на тестирование.

**Задача 4. Тестирование и оценка.** Данная задача — основная для тестирования. Собственно говоря, для ее решения и проводится тестирование. Деятельность, относящаяся к данной задаче, концентрируется:

- на проверке и оценке тестируемых элементов;
- фиксировании информации для диагностики и решения выявленных проблем;
- достижении требуемой широты и глубины оценки;
- установлении обратной связи с самыми важными с точки зрения рисков и качества областями.

Как правило, тестирование и оценка производятся один раз за тестовый цикл. Сюда включены: реализация тестов, оценка тестов, документирование результатов. На данном этапе как никогда важно применение средств автоматизации.

**Задача 5. Достижение приемлемого результата миссии.** Результаты тестирования должны быть полезными относительно сформулированной миссии. Только в этом случае тестирование считается успешным и эффективным. Для этого требуется:

- сформулировать минимально необходимый набор оценок для достижения целей миссии;
- получить результаты, которые помогут улучшить качество продукта;
- выявить отступления от качества;
- обеспечить полезной информацией команду проекта.

**Задача 6. Совершенствование средств и методов тестирования.** Процесс тестирования должен обладать обратной связью с самим собой. Особенное значение последняя задача приобретает, если

инструменты тестирования и сами тесты будут использовать повторно в будущих проектах. Для решения задачи выполняется следующее:

- эффективные тесты сохраняются в репозиторий тестов для дальнейшего использования;
- формируются новые Комплекты тестов;
- удаляются неэффективные и бесполезные средства и методы тестирования;
- обновляется набор Тестовых данных и среда тестирования;
- производится общее обслуживание средств автоматизации тестирования;
- документируется весь полученный опыт (включая негативный).

Выполнение рассмотренных выше задач и соответствующих видов деятельности позволит провести эффективное и успешное тестирование ИС или ПО.

### 6.5.3. Виды тестирования

В современной практике выделяется множество видов тестирования ИС как программного комплекса. Приведем краткое описание каждого вида тестирования.

- *Тестирование пользовательского интерфейса* ставит своей целью проверку интерфейсов, с которыми будут работать конечные пользователи. Проверяется их удобство, корректность работы, поддержка ввода требуемых данных, наличие необходимой функциональности и т. п.

- *Профилирование производительности* позволяет убедиться, что наиболее важные модули ИС имеют приоритет в получении аппаратных ресурсов при проведении операций. Действительно, поддержание наиболее важных модулей ИС имеет важное значение для качества и стабильности системы в целом.

- *Тестирование доступа* имеет принципиальное значение для безопасности. Пользователь должен иметь доступ только к тем функциям и только к той информации, которая соответствует его роли. Также тестируется возможность несанкционированного доступа извне, устойчивость ИС к взлому и хакерским атакам.

- *Нагрузочное тестирование* проверяет, как ИС работает с типичными и пиковыми нагрузками. Сюда включается тестирование при различной пользовательской нагрузке и тестирование при различном объеме данных.

- *Инсталляционное тестирование* необходимо для проверки возможности установить ИС или ПО на конечные операционные системы или аппаратное обеспечение. Проверяется корректность инсталляции, удобство, простота.

- *Тестирование функциональности* нужно для проверки соответствия системы функциональным требованиям.

- *Тестирование целостности данных* проверяет соответствие данных логике и структуре БД и хранилищ данных системы.
- *Тестирование отказоустойчивости* выявляет наиболее распространенные причины отказов системы. Как правило, причины критически важных отказов устраняются или, если это невозможно, подробно документируются.
- *Тестирование цикла работы* позволяет удостовериться в способности ИС выполнять свои задачи, условно говоря, «от А до Я». Проверяется, может ли система выполнить полный цикл каждой функции с самого начала и до завершения.
- *Тестирование на разных платформах* позволяет удостовериться в работоспособности системы с разными программными и аппаратными видами обеспечения.
- *Тестирование программного кода* необходимо для проверки кода на наличие ошибок и осуществляется на ранних этапах тестирования.

Каждый вид тестирования характеризуется собственными артефактами тестирования и собственной миссией. Каждый вид тестирования проводится обособленно, однако в рамках одного тестового цикла или одной итерации, как правило, производится несколько видов тестирования, осуществляемых параллельно.

Порядок проведения видов тестирования достаточно условный. Тем не менее некоторые правила все-таки существуют. Так, тестирование программного кода осуществляется еще в начале его написания. Инсталляционное тестирование проводится в конце и является частью теста на герметичность. Нагрузочное тестирование производится для каждого модуля после его создания.

**Альтернативные классификации видов тестирования.** Существуют и другие классификации видов тестирования. Так, существует классификация видов тестирования по степени изолированности компонентов, содержащая следующие виды:

- модульное тестирование (тестирование отдельных модулей);
- интеграционное тестирование (тестирование объединенных в группу модулей);
- системное тестирование (тестирование всей совокупности модулей и их взаимосвязей).

По степени подготовленности к тестированию выделяют:

- тестирование по документации (основные артефакты рассмотрены ранее);
- интуитивное тестирование (применимо только к очень маленьким проектам).

По признаку позитивности сценариев выделяют:

- позитивное тестирование (тестирование с типичной нагрузкой и допущением минимальности отклонений);

- негативное тестирование (тестирование с критической нагрузкой).

По степени автоматизации выделяют:

- ручное тестирование;
- автоматизированное тестирование;
- полуавтоматизированное тестирование (на практике встречается чаще всего).

По направленности:

- восходящее тестирование (начало с модулей, конец на уровне системы);
- нисходящее тестирование (тестирование ИС в целом и «спуск» на уровень компонентов).

Все рассмотренные виды тестирования так или иначе имеют место, но руководствоваться альтернативными классификациями не рекомендуется. Они могут служить дополнением, но не основой при проведении тестирования.

#### 6.5.4. Правила и рекомендации

Несмотря на множество формальных процедур, проработанных методологий и инструментов автоматизации, тестирование остается одним из наиболее зависящих от человеческого фактора этапов создания ИС. Действительно, проведение тестирования и восприятие его результатов напрямую зависит от конкретных лиц, задействованных в этом процессе. Соответственно, разумно было бы сформулировать некоторые правила, следование которым помогло бы увеличить эффективность тестирования.

Первое правило тестирования — это *необходимость описания входных и выходных данных еще до проведения тестирования*. В противном случае ошибочные результаты, которые выглядят правдоподобными, могут быть признаны в качестве верных. Таким образом, сценарий тестирования должен включать два типа данных: входные и выходные. Но, в зависимости от используемого средства автоматизации, правило может изменяться и даже нарушаться.

Следует помнить, что по возможности *автор не должен тестировать свою программу*. Разработчику будет трудно перестроиться на деструктивный образ мышления при тестировании собственной программы. Обнаружение недостатков в результатах собственного труда — сложная задача. Об этом не понаслышке знают люди творческих профессий, но это утверждение верно и для тестирования ИС/ПО. Кроме того, велика вероятность неправильного понимания механизмов работы программы самим разработчиком, что особенно значимо для функционального тестирования.

Соответственно, в идеале организация, которая занимается разработкой ПО, *не должна проводить тестирование самостоятельно*. Причина значимости этого правила близка к предыдущей. Приме-

нительно к своему продукту компания, как правило, руководствуется понятием «достаточно хорошо». Ф. Крачтен выделял ряд вариаций понятия «достаточно хорошо», которые способны негативно сказаться на итоговом качестве ИС.

- *Не слишком плохо.* В этом случае фирма стремится создать продукт, качество которого будет просто достаточным, чтобы остаться на рынке. Нередко при таком подходе руководствуются принципом «ИС должна быть качественной ровно настолько, чтобы нас не засудили».

- *Абсолютная непогрешимость.* Такая интерпретация «достаточно хорошего» встречается, когда руководство компании-разработчика старается не думать о неудачах по каким-то личным причинам или в надежде мотивировать таким образом сотрудников. Как правило, тестирование в данном случае сводится к демонстрации сильных сторон программы и подтверждению ее работоспособности.

- *Праведное изнеможение.* В данном случае организация-разработчик будет тестировать продукт до последнего. Тестирование может растянуться на множество итераций, а сроки и бюджет будут полностью нарушены. Как правило, в один момент начинают устраиваться мелкие дефекты, не критические для заказчика.

- *Заказчик всегда прав.* В этом случае качественным считается продукт, который нравится заказчику. Этот посыл лежит в философии гибких методологий разработки, но не настолько буквально. Следует помнить, что заказчик, как правило, не разбирается в технических вопросах и способен оценить прежде всего соответствие функциональным требованиям.

- *Установленный процесс.* В данном случае считается, что соблюдение установленного процесса само по себе является гарантом качества. Тестированию уделяются недостаточное внимание, либо же оно существует «для галочки», как один из этапов процесса.

- *Статические требования.* Это вариация утверждения «заказчик всегда прав» с акцентом на обязательное дословное выполнение требований, даже если они были сформулированы в неявном виде или содержат противоречивые указания.

- *Отчетность.* Как и следует из названия, качественной считается система, условия качества которой описаны в контракте. Тестирование используется для проверки соответствия этим условиям.

Очевидно, что все рассмотренные выше вариации «достаточно хорошей» ИС не соответствуют объективному качеству ИС. Наиболее адекватным вариантом были бы динамические компромиссы, в соответствии с которыми продукт считается достаточно хорошим, если он предоставляет заказчику существенные преимущества и не имеет критических проблем, а некритические проблемы при-

носят гораздо меньше вреда, чем генерирует полезности система. Оптимальным вариантом достижения такого результата является тестирование сторонней компанией.

Важно понимать, что результаты применения каждого теста должны быть досконально изучены. В противном случае тестирование вообще теряет смысл. К счастью, современные инструменты тестирования содержат мощные алгоритмы аналитики, и проблема существенно упрощается. Но в случае ручного тестирования, которое потребовалось по каким-либо причинам, проблема снова становится очень острой.

При тестировании входных данных следует обязательно проверять варианты взаимодействия ИС с *заведомо неправильными входными данными*. Очень часто такое тестирование приносит существенную пользу, поскольку позволяет выявить ошибки, которые в противном случае нельзя обнаружить.

Недостаточно проверять только выполнение программой своих задач. Нужно убедиться, что *программа не делает чего-то, что делать не должна*. К примеру, финансовая подсистема может верно рассчитывать зарплату сотрудников, но в то же время из-за существующих ошибок вносить изменения в файлы с личными данными.

И, разумеется, при планировании проекта ни в коем случае *нельзя исходить из допущения, что ошибки не будут обнаружены*. Также нередко на практике придерживаются правила: если в начале тестирования был найден ряд ошибок, то по мере проведения тестирования их число будет увеличиваться.

#### 6.5.5. Инструментальные средства тестирования

Сегодня на практике применяется множество автоматизированных средств тестирования. Наибольшую популярность получили продукты таких компаний, как IBM, HP, AutomatedQA и Borland. На их основных продуктах следует остановиться подробнее.

**Инструменты и средства тестирования IBM.** Пожалуй, IBM предлагает наиболее широкий спектр инструментов для тестирования на современном рынке. Первый инструмент, предназначенный для функционального тестирования, — это Rational Functional Tester. Инструмент поддерживает широкие возможности визуализации тестирования, допускает внесение изменений посредством естественного языка. Процесс тестирования автоматизируется с особым вниманием к частым изменениям интерфейса. Инструмент поддерживает работу со сценариями тестирования и в целом ориентирован на данные (совершение одинаковых действий с разными наборами данных).

Далее следует рассмотреть Rational Performance Tester. Данное решение предназначено для нефункционального тестирования. Оно используется для тестирования производительности и проверки

масштабируемости приложений. Решение поддерживает тестирование без программирования. Решение включает инструменты анализа первопричин, благодаря которым можно обнаружить причины низкой производительности. Тестирование проводится на основе сценариев, данных, пользовательского кода. Как правило, данное решение используется перед развертыванием ИС, чтобы убедиться в ее работоспособности.

Rational Service Tester необходим для функционального тестирования служб, которые не относятся к графическому пользовательскому интерфейсу. С помощью этого инструмента производится создание, изменение и выполнение тестов работоспособности и производительности. Возможно также проведение быстрого дополняющего тестирования в режиме реального времени.

Отдельного внимания заслуживает группа автоматизированных инструментальных средств для проведения анализа и повышения надежности и производительности приложений Rational PurifyPlus. Сюда включены три средства:

- Rational Purify, который обнаруживает утечки памяти и осуществляет поиск run-time ошибок;
- Rational Quality, который осуществляет сбор ключевой информации о тестируемом приложении;
- Rational PureCoverage, который осуществляет поиск упущеных при тестировании фрагментов в коде или работе программы.

Для функционального тестирования активно применяется средство Rational Robot. С его помощью производится автоматизированное тестирование клиент-серверных приложений.

И, наконец, среди продуктов IBM важное место занимает Rational Test RealTime. Средство тестирует созданные разработчиками компоненты, производит отладку программы, автоматически моделирует тестовые модули.

**Инструменты и средства тестирования HP.** Один из ведущих инструментов функционального тестирования выпускается компанией HP. Таковым инструментом является HP QuickTest Professional. Одна из важных причин популярности средства — это рекордер пользовательской активности, который позволяет преобразовать действия пользователей в скрипты. В результате существенно снижается время, необходимое для разработки сценария тестирования. Интересно, что информация обо всех объектах, с которыми взаимодействует пользователь, автоматически сохраняется в специальный репозиторий. Из-за этого на начальных этапах требуется настройка программы.

Компания-производитель рекомендует использовать данный продукт вместе с HP Quality Center, являющимся законченной интегрированной системой для контроля качества на всех этапах ЖЦ ИС или ПО.

Для тестирования производительности существует продукт HP LoadRunner. Инструмент позволяет генерировать сценарии тестирования, генерировать виртуальную нагрузку, разрабатывать и запускать сценарии, а также автоматически анализировать и визуализировать результаты нагрузочного тестирования.

**Инструменты и средства тестирования Borland.** С 2006 г. часто применяется инструмент под названием SilkTest. Инструмент предназначен для автоматизированного функционального тестирования ПО. Решение поддерживает два варианта тестирования: с автоматическим захватом объектов (простой вариант, доступный даже неподготовленным тестировщикам) и с написанием скриптов на .NET. При этом оба способа могут использоваться одновременно и дополнять друг друга.

Один из основных плюсов SilkTest — это точное моделирование методов работы конечного пользователя. Решение обращается к тестируемому приложению, как это делает пользователь (через графический интерфейс). Кроме того, решение считается достаточно дешевым, простым в использовании и легким в обслуживании.

Продукт поддерживает интеграцию с SilkTest Manager, необходимым для обеспечения управления тестированием. Компания Borland была приобретена Micro Focus International в 2009 г.

**Инструменты и средства тестирования AutomatedQA.** Инструмент TestComplete оптимален для применения, если тестирование осуществляется новичками в данной области или людьми, которые не слишком разбираются в программировании. Инструмент успешно применяется в случаях, когда предполагается имитация действий пользователей. TestComplete является очень популярным в России и на территории СНГ инструментом автоматизированного тестирования.

## Контрольные вопросы и задания

1. Зачем применяются методологии проектирования ИС?
2. Какие этапы были пройдены в развитии методологий разработки ИС?
3. Как выбрать подходящую методологию разработки ИС?
4. Как системный анализ применяется при разработке ИС?
5. Что такое информационный инжиниринг?
6. Назовите основные особенности структурного, процессного и объектно-ориентированного подходов.
7. Чем отличаются подходы к разработке «снизу вверх» и «сверху вниз»?
8. Какие существуют гибкие методологии разработки ИС?
9. Какими особенностями характеризуется методология Scrum?
10. Как можно использовать методологию Kanban при разработке ИС?
11. В каких условиях лучше всего применять методологию eXtreme Programming и почему?

12. В чем преимущества гибкой методологии разработки FDD?
13. Как и зачем используются CASE-средства при разработке ИС?

## **Рекомендуемая литература**

1. Амблер, С. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки : перевод с английского / С. Амблер. — Санкт-Петербург : Питер, 2005.
2. Бек, К. Экстремальное программирование : перевод с английского / К. Бек. — Санкт-Петербург : Питер, 2002.
3. Грекул, В. И. Проектирование информационных систем // Национальный открытый университет «ИНТУИТ», 2012. — URL: <http://publications.hse.ru/books/74833061> (дата обращения: 17.10.2020).
4. Канбан и «точно вовремя» на Toyota. Менеджмент начинается на рабочем месте / научный редактор Ю. Адлер ; перевод с английского Е. Пестеревой. — Москва : Альпина Паблишер, 2015.
5. The SCRUM Guide // SCRUMGUIDES.ORG. 2013. — URL: <http://www.scrumguides.org/scrum-guide.html> (дата обращения: 27.07.2020).

# **Тема 7**

## **ОСОБЕННОСТИ ПРОЕКТОВ В ОБЛАСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ НА ФАЗАХ ЖЦИС**

---

Эта тема описывает проектный подход, который применяется при управлении этапами жизненного цикла информационной системы, поскольку все фазы ЖЦИС (кроме эксплуатации) имеют проектный характер. Рассматриваются основные области проектного управления, включая управление заинтересованными сторонами, управление содержанием, управление сроками, управление стоимостью, управление рисками, управление качеством, управление командой, управление портфелем проектов. Рассмотрено понятие офиса управления проектами. В теме подробно рассмотрены корпоративные методологии основных вендоров, а также российские и международные стандарты.

После изучения данной темы студент будет:

**знатъ**

- основы управления заинтересованными сторонами;
- основы управления содержанием;
- основы управления сроками проекта;
- основы управления стоимостью проекта;
- основы управления рисками;
- основы управления качеством;
- основы управления командой проекта;
- основы управления портфелем проектов;
- понятие офиса управления проектами;
- основы корпоративных методологий Microsoft (MSF, On Target, MBS Partner Methodology);

• основы корпоративной методологии SAP (Accelerated SAP);  
• основы корпоративной методологии Oracle (OUM);  
• основные российские и международные стандарты (PMBOK, PRINCE2, ISO 21500:2012, ГОСТ Р 54869-2011);

**уметь**

- управлять заинтересованными сторонами, содержанием, сроками проекта, стоимостью проекта, рисками, качеством, командой проекта, портфелем проектов;
- производить обоснованный выбор наиболее эффективной в заданном контексте методологии проектного управления;

### **владеть навыками**

- управления проектами;
  - применения методологий и стандартов проектного управления;
  - поиска необходимой информации в методологиях и стандартах проектного управления.
- 

Управление проектами на всех фазах жизненного цикла информационной системы — это управление в условиях постоянно изменяющихся требований со стороны бизнеса, в условиях развивающейся функциональности системы, в условиях постоянного развития новых информационных технологий.

Работа КИС касается деятельности различных служб и должностных лиц предприятия. Однако отношения между ними могут быть более чем непростыми. Информатизация всегда вызывает слухи о сокращении штатов и повышении норм выработки. Более того, многие специалисты стремятся сохранить свою область деятельности от постороннего вмешательства, а потому возможны попытки прямого (или скрытого) саботажа нововведений на предприятии. В редких случаях может потребоваться привлечение психологов для работы с коллективом.

Невозможно не отметить, что создание КИС вызывает ряд сложных организационных проблем. Для создания КИС ИТ-службе выгоднее всего привлечь какую-либо стороннюю организацию. Однако ИТ-службе в любом случае придется отвечать за работу этой организации. Попытка обойтись своими силами чревата срывом сроков, а также острым дефицитом ресурсов при обилии текущих задач.

Зачастую профессиональные консультанты, нанятые «на стороне», могут беспристрастно и объективно оценить используемые на предприятии информационные технологии. Консультанты могут разграничить свою сферу деятельности и сферу деятельности ИТ-службы компании-нанимателя. В штате предприятия, тем не менее, должны находиться сотрудники, способные квалифицированно понять и проанализировать действия нанятых консультантов, поэтому предприятию не обойтись без собственных специалистов. ИТ-службы должны контролировать и координировать работы по созданию КИС начиная с самого первого этапа планирования и обследования, так как только они являются представителями интересов бизнес-заказчика на предприятии и могут обеспечить непредвзятость расчетов финансовых и временных ресурсов для проекта.

Процедура стратегического планирования должна стать основой для построения КИС на предприятии. Следом за этим необходимо определить основные бизнес-процессы предприятия и выделить информационные структуры, обеспечивающие их. Только в этом

случае созданная информационная система сможет стать базой для работы и развития предприятия.

Следует учитывать и изменяющиеся условия бизнеса, иначе КИС может устареть прежде, чем будет завершена. Безусловно, управлять глобальным проектом вроде создания КИС следует с точки зрения финансового менеджмента, однако традиционное планирование становится неадекватным в условиях современного динамичного рынка. Однако полный отказ от планирования также может привести к серьезным проблемам. Более того, создавать КИС по частям можно только при наличии единого системного проекта, иначе потери могут превысить ожидаемую пользу. Именно поэтому в настящее время все больше предприятий в сотрудничестве с консалтинговыми компаниями (такими, как Accenture, IBM и BearingPoint) организуют обновление или же создание с нуля стратегии развития ИС. Как правило, подобные проекты достаточно стандартны в плане фаз и включают в себя три основные стадии:

- 1) анализ текущего состояния ИТ на предприятии (анализ As-Is);
- 2) разработка целевого плана ИТ (формирование целевой модели To-Be);
- 3) разработка плана перехода от текущего состояния к целевому, еще называемого «дорожной картой» (*roadmap*).

---

**ИТ-стратегия** — план управления ИТ стратегического уровня, определяющий основные направления развития информационных технологий для оптимального их использования и достижения поставленных компанией бизнес-целей.

---

Причины, по которым компании принимают решение разрабатывать ИТ-стратегию, могут быть самыми разными.

- Изменение стратегии бизнеса требует пересмотра ИТ стратегии.
- Изменения организационной структуры (новое руководство ИТ-департамента, слияние нескольких компаний и их ИТ-функций).
- Предыдущая концепция развития ИТ требует пересмотра в силу своей неэффективности, превышения запланированных затрат, нарушения графика реализации.
- Необходимо сформировать механизм взаимодействия бизнеса и ИТ в соответствии со стратегическими задачами компании.
- Необходимо обосновать эффективность долгосрочных ИТ-инвестиций.

Соответственно, определенную роль играют и внутренние факторы (операционный менеджмент, оргструктура), и внешние (например, M&A). В зависимости от целей формирования стратегии определяется, в течение какого периода она будет использоваться (кратко-, средне-, долгосрочный).

В итоге проектов по формированию ИТ-стратегии могут создаваться следующие результирующие документы и артефакты:

- список целевых мероприятий и портфель проектов ИТ (в том числе планирование инвестиций в ИТ, паспорта проектов ИТ, балансировка портфеля ИТ, оценка бюджетов проектов ИТ);
- система управления проектной деятельностью (в том числе организационно-методологические принципы и стандарты управления проектами);
  - каталог услуг ИТ;
  - модель сорсинга ИТ;
  - описание организационной структуры ИТ;
  - модель ИТ-процессов.

Непосредственно в ходе разработки создаются ландшафты бизнес-процессов, приложений, данных и ИТ-инфраструктуры, которые могут использоваться при дальнейшем построении систем на предприятии.

Другим значимым типом проектов, способствующих эффективному построению ИС на предприятии, является ИТ-аудит.

---

**ИТ-аудит** — системный процесс получения, оценки и предоставления руководству компании объективных данных о текущем состоянии ИТ-инфраструктуры, информационных систем, бизнес-процессов организации, действиях и событиях, происходящих в сфере ИТ, устанавливающий уровень их соответствия определенным критериям и предоставляющий результаты заказчику.

---

Его результаты могут использоваться для планирования дальнейшей деятельности компании в сфере ИТ и ее оптимизации. Решения могут затем приниматься с учетом того, насколько эффективно ИТ-подразделения используют имеющиеся ресурсы для решения задач компании; насколько ИТ-деятельность соответствует предъявляемым требованиям как со стороны организации, так и со стороны внешних контрагентов и регуляторов; насколько надежно организованы работа ИТ-систем и обеспечение безопасности данных.

В ходе ИТ-аудита руководству компании предоставляются объективные данные о текущем состоянии ИТ — инфраструктуры, информационных систем, бизнес-процессов организации, действиях и событиях, происходящих в сфере ИТ. В частности, может проводиться аудит систем в процессе разработки, в ходе которого определяется соответствие процессов проектирования и разработки систем утвержденным регламентам и стандартам уровня организации, отрасли или мировых практик. Проводится также оценка эффективности управления проектом, анализа выгод и рисков, сбора требований, анализа запросов на изменение (в том числе

при использовании данных CMDB), процедур отката изменений и управления исходным кодом, процессов разработки системы контролей.

В процессе внедрения важен аудит инфраструктуры ИТ, оценивающий размещение, состав и состояние инфраструктурных активов организации, в том числе инженерных систем (бесперебойного электропитания, кондиционирования), сетевого обеспечения (LAN), вычислительных мощностей (программно-аппаратных платформ, хранения данных и резервного копирования).

Внедрение ИС на предприятии требует значительных (и не только финансовых) ресурсов. Эффективное внедрение КИС может стать только результатом командной работы, причем в команду должны входить представители разработчика и заказчика. Необходима сторонняя критика от людей аналитического и синтетического складов ума. Подробнее подход к формированию проектной команды будет рассмотрен в соответствующем пункте «Управление человеческими ресурсами».

Но не следует забывать, что далеко не последнюю (а скорее главную) роль в процессе внедрения информационной системы играет пользователь. Пусть он не участвует в действиях по внедрению на регулярной основе, но именно для него создается система, именно он будет ее использовать в своей деятельности. А значит, именно на пользователей должен быть направлен фокус в проекте, именно их требования необходимо учитывать для достижения максимального эффекта от внедрения системы и получения ожидаемых результатов при ее эксплуатации.

## **7.1. Управление фазами ЖЦИС в контексте проектной деятельности**

### **7.1.1. Управление заинтересованными сторонами**

Как уже неоднократно отмечалось, важным аспектом для внедрения и существования систем в частности и деятельности организации в целом является согласованность действий ИТ-департамента с интересами бизнес-пользователей и, конечно, руководства компаний. Формирование бизнес-архитектуры, сбор требований, процесс внедрения — любые действия, относящиеся к ИТ, так или иначе затрагивают интересы других сторон. Ценность ИТ-проекта должна быть так или иначе «донесена» до каждой из них для понимания смысла сотрудничества и, соответственно, выделения необходимых инвестиций.

Выделяют несколько основных заинтересованных сторон и основные вопросы, актуальные для них (табл. 7.1).

Таблица 7.1

## Заинтересованные стороны

Бизнес-заказчики	Представления о ценности ИТ и основные актуальные вопросы
Предприниматели	Как ИТ поможет мне создать новые конкурентные преимущества?
Инвесторы	Как и с какой прибылью вернутся мои инвестиции? Сравнение эффекта от ИТ с другими инвестиционными проектами
Топ-менеджеры	Как корпоративные ИТ помогают реализовать стратегические цели и планы бизнеса и снизить бизнес-риски?
Менеджеры среднего уровня	Ценность ИТ в сокращении затрат и уменьшении рисков операционной деятельности. Как обеспечить заданный уровень качества дешевле и с меньшими рисками?
Консультанты	Ценность ИТ не обсуждается — она очевидна. Ценность ИТ подтверждается практикой лидеров. Копируйте их — и ценность придет сама

Таким образом, важен не просто учет наличия возможного конфликта интереса, но и понимание основных приоритетов и движущих сил каждой из сторон, что в дальнейшем можно использовать с выгодной стороны. Как уже неоднократно подчеркивалось, любое внедрение системы сопряжено с внесениями изменений в процессы текущей деятельности, как минимум, задействованных в проекте подразделений (как максимум — предприятия и принципов взаимодействия с контрагентами). Соответственно, среди всех сотрудников обязательно будут как активные сторонники, так и противники подобных перемен, и крайне важна корректная оценка потенциального влияния проекта на всех субъектов, а также влияния субъектов непосредственно на проект (как положительного, так и отрицательного).

**Организационные условия осуществления проекта.** Невыполнение организационных условий непосредственно влияет на сроки, стоимость и даже принципиальную возможность достижения требуемого результата проекта. Этими условиями являются:

- сильная поддержка проекта со стороны Генерального директора (и других топ-менеджеров) предприятия заказчика;
- осознание руководством предприятия крайней необходимости внедрения системы;
- готовность руководства предприятия к четкой организации проекта обследования предприятия и реализации системы;

- готовность руководства предприятия к выделению квалифицированных сотрудников для совместной работы по проекту со специалистами подрядчика;
- готовность предприятия и его руководства к внедрению и проведению неизбежных изменений в управленических процессах, корпоративных стандартах учета и отчетности, созданию у сотрудников предприятия твердого убеждения неизбежности нововведений, поддержанию высокого статуса проекта и закреплению всех проектных распоряжений соответствующими приказами руководства;
- формирование совместной группы внедрения со стороны заказчика и подрядчика;
- создание органа корпоративного управления заказчика и проведение его заседаний по утвержденному регламенту, но не реже одного раза в месяц;
- назначение куратора проекта в ранге не ниже заместителя генерального директора компании заказчика, а также ответственных лиц по подразделениям предприятия для участия в проекте;
- создание для группы внедрения необходимых условий для работы на предприятии заказчика (помещение, доступ к ИТ-инфраструктуре), в том числе для работы в условиях вахтенного метода, т. е., возможно, без выходных дней и в нерабочее время.

**Управление реализацией и решение спорных вопросов.** Правильное и непрерывное управление проектом является залогом его успешного осуществления и завершения. Появление промежуточных результатов в ходе реализации проекта позволяет начать их использование еще до завершения проекта в целом. Например, завершение этапа автоматизации учета основных средств предприятия позволяет, не дожидаясь окончания проекта, осуществлять расчет амортизации.

Для решения возникающих в ходе проекта спорных вопросов и проблем применяется процедура управления решением спорных вопросов, призванная обеспечить качественное выполнение задач проекта в срок и снизить степень влияния связанных с этими вопросами факторов риска.

Решение каждого вопроса (проблемы) проходит четыре стадии:

- идентификация проблемы;
- принятие к дальнейшему рассмотрению либо отказ от такого;
- одобрение предложенного решения;
- закрытие проблемы.

Проблема может быть идентифицирована членами команды внедрения, пользователями или управленческим персоналом. В момент регистрации каждый спорный вопрос должен быть документирован с помощью Формы регистрации проблем или аналогичного доку-

мента. Для принятия решения о целесообразности документирования проблемы используется один из приведенных ниже критериев.

- Насколько существенно повлияет отказ от решения этой проблемы на проект (он может, например, вызвать задержку, изменение курса проекта, снижение качества или увеличение стоимости проекта)?
- Выходит ли проблема за рамки одной задачи или требует внимания более чем одного консультанта?
- Требует ли проблема решения со стороны куратора или руководителя проекта?

Проблемы должны оцениваться по степени своего влияния на проект по степени важности.

1. *Высокая*. Такие проблемы должны быть рассмотрены руководителем проекта и утверждены Координационным советом (например, изменение порядка лимитирования накладных расходов относительно порядка, описанного в проектном решении).

2. *Средняя*. Такие проблемы должны быть рассмотрены и решены руководителем проекта совместно с кем-либо из заместителей директоров предприятия заказчика. Примером может быть необходимость перераспределения должностных обязанностей между двумя или несколькими подразделениями, работающими в системе. Обо всех принятых таким образом решениях руководитель проекта отчитывается на очередном заседании Координационного совета.

3. *Низкая*. Рассмотрение и решение таких проблем осуществляются руководителем проекта самостоятельно. Обычно проблемы такого рода не имеют альтернативных решений. Суть такой проблемы обычно заключается в необходимости принять конкретное заранее известное решение (например, организовать своевременность ввода первичных документов на складе N).

Каждая проблема должна быть рассмотрена руководителем проекта со стороны исполнителя, в результате чего ей должен быть назначен один из следующих статусов.

- *Отклонена* — проблема отклоняется, если она не препятствует нормальному ходу выполнения проекта и не влияет на его коначный результат. Перед тем как отклонить проблему, необходимо обсудить это решение с лицом, идентифицировавшим эту проблему. Это позволит инициировавшему лицу еще раз сформулировать суть проблемы и, возможно, более удачно передать ее важные аспекты.

- *Отложена* — проблема откладывается, если решение по этой проблеме не может быть принято в момент рассмотрения. В этом случае обязательно должна быть указана дата, до которой откладывается решение проблемы.

- *Объединена* — проблемы, связанные тесно между собой, должны быть объединены. При объединении одна из проблем долж-

на быть закрыта, а описание другой проблемы должно быть изменено с учетом произошедшего объединения.

- *Принята* — для принятых проблем производится рассмотрение и оценка вариантов решения, выбор варианта и назначение исполнителя для его реализации.

Как только по какой-либо проблеме принято решение, оно должно быть преобразовано в последовательность шагов по его реализации, в числе которых должны быть определены:

- выполняемые задачи;
- подготавливаемые документы, отчеты и другие рабочие продукты;
- ответственные за выполнение задач и за техническую подготовку;
- критерии качества и завершения.

Как правило, списком проблем управления проектом занимается проектный менеджер. Затем этот список последовательно передается на рассмотрение куратору и Координационному совету (с занесением записей в проектную документацию — как правило, Протокол выполненных работ и Журнал регистрации проблем).

**Способы предотвращения конфликтных ситуаций.** Можно представить множество причин возникновения конфликтных ситуаций в процессе развития проекта. В данном разделе выделены основные источники и причины возникновения таких проблем, а также возможные пути их предотвращения.

1. Цель проекта в контексте решения критических вопросов предприятия должна быть поставлена перед группой внедрения достаточно четко. Высшее руководство предприятия должно заявить о полной поддержке людей, задействованных в проекте. В противном случае это приведет к снижению мотивации и личной заинтересованности и в конечном итоге существенно снизит вероятность успешного и эффективного завершения проекта.

2. Представители проектной команды должны регулярно встречаться друг с другом и с членами Координационного совета с целью обсуждения текущих вопросов, анализа состояния проекта и обмена мнениями. В противном случае может возникнуть ситуация, когда проект не будет расцениваться как самая приоритетная задача, а цели и задачи станут размытыми. Созыв Координационного совета также позволяет решать спорные вопросы и подводить промежуточные итоги по выполненному объему работ. Обсуждение способов решения конфликтных ситуаций должно проводиться в максимальном составе проектной команды. Каждый ее представитель представляет отдельное подразделение и, следовательно, имеет свое видение проблем и методов их решения. Различия в подходах должны быть учтены и, как следствие, должны быть выработаны единые порядок и нормы взаимодействия.

3. Другой потенциальный источник конфликтных ситуаций — это приоритеты управления и оценка производительности. Типичной является ситуация, когда член проектной команды напрямую отчитывается перед своим непосредственным руководителем и косвенно — перед руководителем команды внедрения. Требования, высказываемые обоими руководителями, вступают в противоречие, и рядовой член команды внедрения находится в состоянии растянутости. Ситуация, когда человек задействован на двух работах, потенциально может вылиться в две проблемы:

- члены проектной команды не поощряются в материальном плане за совмещение двух работ, и
- их вклад в общее дело остается недооцененным.

4. Исторически сложилось так, что оценка действий сотрудника в основном зависит от того, насколько успешно он работает на своем непосредственном рабочем месте. Построение системы учета вклада каждого члена проектной команды в проект гарантирует, что он будет лучше осознавать собственную значимость, а также предоставит ему дополнительные стимулы к повышению эффективности выполняемых работ.

Моральное и материальное поощрение членов проектной команды по результатам выполненных работ следует рассматривать, с одной стороны, как возможный источник мотивации, а с другой — как источник конфликтных ситуаций как в пределах группы внедрения, так и при взаимодействии с другими подразделениями предприятия. Существует множество систем поощрений, которые можно использовать в целях повышения мотивации. Любая из них должна быть тщательнозвешена с позиции ее влияния на работу команды.

### 7.1.2. Управление содержанием

**Проектная документация.** Важность документации очень просто недооценить, однако именно благодаря ей любой человек сможет уяснить как общую картину и суть проекта, так и узнать самые незначительные детали его реализации. Сквозное документирование проекта (включающее все результаты проведенных работ, замечаний, разногласий, предложений, проведенных встреч) должно осуществляться в обязательном порядке. Основным элементом управления проектом является регулярный отчет руководителя проекта о состоянии проекта, в котором отражаются результаты выполнения работ по внедрению на момент формирования отчета. На протяжении всего проекта в отчете отражаются работы, выполненные в настоящее время, находящиеся на стадии выполнения (с указанием степени завершенности) и предстоящие в ближайшем будущем. Типичные документы, создаваемые в ходе проекта, перечислены в табл. 7.2.

Таблица 7.2

## Типичные документы управления ИТ-проектами

Название документа	Автор/ ответственный	Периодичность	Адресат документа
<i>Регулярная отчетность по проекту</i>			
Индивидуальный отчет о проработанном времени	Консультант-программист от исполнителя	По указанию Руководителя проекта от исполнителя	Руководитель проекта от исполнителя
Отчет Руководителя проекта	Руководитель проекта от исполнителя (при его отсутствии — Руководитель проекта от Заказчика)	Ежемесячно	Куратор проекта, Координатор проекта
Протокол совещания Координационного совета	Секретарь совещания	При проведении совещания	Куратор проекта, Координатор проекта
<i>Отчеты Руководителя проекта</i>			
Регулярный отчет о состоянии проекта	Руководитель проекта от исполнителя	Ежемесячно	Куратор проекта, Координатор проекта
Отчет о результатах этапа	Руководитель проекта от исполнителя	При завершении этапа	
Протокол выполненных работ	Руководитель проекта от исполнителя	При выполнении работ	Руководитель проекта от исполнителя
<i>Процесс решения проблем</i>			
Форма регистрации проблемы	Консультант со стороны исполнителя		Руководитель проекта от исполнителя
Журнал регистрации проблем	Руководитель проекта от исполнителя		Куратор проекта, Координатор проекта
Изменения статусов проблем	Руководитель проекта от исполнителя		

Для эффективного управления проектом важными являются все без исключения составляющие. К примеру, даже такая незначительная на первый взгляд информация, как сведения о плане отсутствия специалистов (отпуска, командировки и пр.), является необходимой для планирования конкретных работ проекта и соответствующего использования специалистов. Именно из совокупности таких

мелких элементов складывается качественное выполнение проекта внедрения и обеспечивается гарантия успеха проекта в целом.

**Системная документация.** Пристальное внимание следует уделить формированию системной документации. Для формирования документации можно использовать диаграммную технику представления управленческих процессов. Примером такой техники могут служить IDEF-диаграммы. При помощи функций описываются основные процессы, которые преобразуют входящую информацию и формируют выходную. Количественные функции (длительность, стоимость и т. п.) позволяют моделировать управленческие процессы и анализировать варианты их улучшения.

Документация системы должна включать в себя следующие элементы: информационная архитектура, программная архитектура, программно-техническая реализация. Базисом документации по поддержке и дальнейшему развитию КИС должны стать технологические, организационные и методологические составляющие. Эти составляющие представляют собой, по сути, правила изменения системы и могут охватывать все стадии проекта.

Документация должна быть логически полной и удобно представленной. Предприятия, включая АСУ-подразделения, нередко страдают отсутствием или невыполнением регламентов, слабой дисциплиной в области эксплуатации и замедленным восприятием изменений системы. Язык спецификаций — важная деталь, позволяющая заказчику и подрядчику «говорить на одном языке». Создание любой составной части КИС должно сопровождаться созданием структурных схем и спецификаций, заранее согласованных с заказчиком. Логическая полнота спецификаций является обязательным условием, как и понимание их формулировок обеими сторонами.

Их понимание со стороны бизнеса важно для соблюдения интересов предприятия (и корректного планирования ресурсов и масштабов проекта) как во время заключения с подрядчиком контракта на системную интеграцию, так и в дальнейшем при прохождении всех этапов жизненного цикла этой информационной системы.

### 7.1.3. Управление сроками проекта

Управление сроками проекта — одно из ключевых направлений деятельности проектного управления. Управление сроками проекта осуществляется при помощи *Расписания проекта* (*Schedule*). Расписание проекта — это плановые даты исполнения плановых операций и наступления контрольных событий.

Соответственно, управление сроками проекта может быть представлено в виде управления расписанием проекта. Говоря об управлении расписанием, необходимо начать с *Плана управления расписанием*. Несмотря на кажущуюся избыточность, это очень важный документ. Большинство практиков рекомендуют либо создавать

план управления расписанием, либо представлять, что он реально существует.

План управления расписанием — это концепция представлений и размышлений об управлении расписанием, оформленная в виде документа. План управления расписанием содержит:

- базовое расписание проекта, по которому ведется контроль и мониторинг;
- шкала измерения отклонений от расписания проекта;
- план управления отклонениями в расписании проекта;
- процедуры контроля изменения расписания проекта.

Следующий важный элемент управления расписанием проекта — это так называемые *вехи*. Вехи — это важные для проекта события. Есть много способов выделить или задать вехи. Иногда вехи определяются еще спонсором проекта в момент формирования Устава проекта. Также в виде вех часто рассматривают события, которые обозначают завершение последовательности работ. Все вехи проекта собираются в *Список вех*. Данный документ входит в План управления проектом, Описание содержания работ и *Иерархическую структуру работ (ИСР)*.

Далее следует рассмотреть Иерархическую структуру работ. ИСР выглядит как схема (рис. 7.1), в нижней части которой находится декомпозиция верхнего уровня. ИСР позволяет разбить проект на мелкие и измеримые части. Работы и результаты, которые не вошли в ИСР, не являются частью проекта.



Рис. 7.1. Иерархическая структура работ (пример)

Самая малая часть ИСР — это **операция**. Разбиение пакета работ до уровня элементарных операций позволяет упростить оценку, мониторинг и контроль проекта. Каждой операции присваивается название или ID. Зачастую ИСР создается до уровня полной таблицы с перечнем операций, и это оправданное решение — операций может быть слишком много. Тем не менее после создания списка операций ИСР может быть еще раз пересмотрена.

Иногда можно встретиться с ситуацией, когда состав работ неизвестен, в результате чего ИСР не может быть сформирована полностью. В этом случае предварительную версию ИСР нужно составлять очень осторожно, а дальнейшую детализацию производить по мере начала работ. Такой подход получил название *метода набегающей волны*, но не стоит злоупотреблять им и применять в случае, когда детальную ИСР можно построить заранее.

**Сетевые графики.** После определения операций и составления списка операций определяется последовательность выполнения операций. Результатом этой деятельности являются сетевые диаграммы расписания PDM или ADM.

**Диаграммы PDM.** Сегодня чаще всего практиками используются диаграммы PDM.

На диаграмме PDM операции обозначены в виде узлов диаграммы (рис. 7.2). Дуги или стрелки отображают последовательность взаимодействия между операциями. Используются четыре типа зависимости: Старт-Старт, Финиш-Старт, Старт-Финиш и Финиш-Финиш.

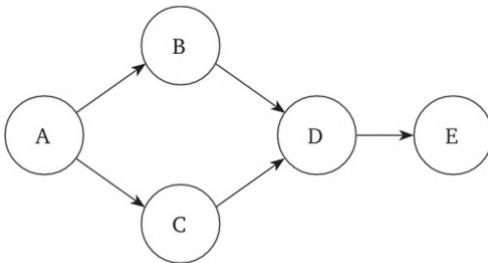


Рис. 7.2. Диаграмма PDM

**Диаграммы ADM.** Реже используются диаграммы ADM. Здесь операции отражаются на дугах или стрелках диаграммы, а узлы задают зависимости между операциями (рис. 7.3).

Диаграммы ADM допускают пустые (нулевые) операции. Пустые операции обозначаются пунктирными линиями. Они используются для увеличения наглядности. Длительность пустых операций равна нулю, и они не требуют никаких ресурсов.

Диаграммы ADM и PDM представляют собой сетевые графики. Сетевой график обычно предполагает движение слева направо.

Ни одна операция в сетевом графике не может быть начата, пока не завершена предыдущая, связанная с ней, операция. Стрелками обозначаются последовательности, причем стрелки могут пересекаться. Каждая операция обозначается своим номером, причем номер последующей операции должен быть выше, чем номер предшествующей. Образование петель считается недопустимым (т. е. не должно быть циклов).

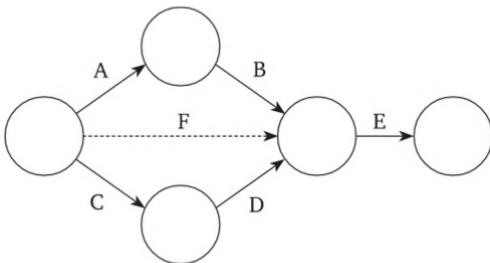


Рис. 7.3. Диаграмма ADM

Важно понимать, что сетевые графики никогда не содержат условий в стиле «Если операция А выполнена успешно, то переходите к операции В, а если нет, то переходите к операции С».

Отдельно следует отметить, что любая операция используется только один раз. Если операция присутствует в сетевом графике повторно, она должна иметь новый номер.

В сетевых графиках используют три типа зависимостей между операциями:

- директивные (последовательность работ жестко определена);
- дискретные (последовательность работ определяется менеджером проекта);
- внешние (последовательность работ определяется внешними условиями).

**Оценка длительности операций.** Когда созданы сетевые графики (т. е. когда определен порядок операций), можно переходить к созданию *иерархической структуры ресурсов*. В качестве ресурсов рассматриваются персонал, оборудование, материалы, и пр. Ресурсы должны быть спланированы таким образом, чтобы не было ни нехватки, ни излишков ресурсов. Все ресурсы должны кодироваться, причем ID ресурсов нужно организовать и сгруппировать удобным способом.

После этого можно переходить к оценке длительности операций, поскольку их длительность чаще всего зависит именно от доступности ресурса. Выделяют несколько методов оценки длительности.

- *Оценка One-Time*, которая назначается специалистами на основании экспертных оценок. Иногда такая оценка просто «угады-

вается». Очень часто при этом методе оценщики ошибаются. В базовое расписание, составленное по этому методу, обычно никто не верит. При этом часто возникает конфликт между специалистом по оценке и менеджером проекта.

- *Оценка по аналогии* используется, когда проект или операция аналогичны ранее выполненным.

- *Параметрическая оценка* базируется на математических методах, основанных на исторической или другой информации. Может использоваться время, требующееся для создания одной строки кода, время закупки оборудования и т. п. Часто используется кривая обучения (1000 строк кода могут быть написаны быстрее, чем 100).

- *Эвристическая оценка* использует ряд правил, к примеру, правило 80/20. Это правило означает, что часто 80 % проблем обусловлены 20 % источников проблем. Часто эвристическая оценка основывается на результатах параметрического анализа длительности.

- *Оценка по трем точкам* основана на методах статистики. Даются пессимистическая, оптимистическая и наиболее вероятная оценка длительности. На основе этого рассчитывается математическое ожидание, стандартное отклонение и дисперсия.

Оценки длительности операций ложатся в основу Метода критического пути (МКП). *Критический путь* — это самый длинный путь в сетевом графике (рис. 7.4). Чем длиннее этот путь, тем больше минимально возможное время завершения проекта. На следующем рисунке приведен упрощенный пример критического пути. Длительность каждой работы указана в скобках после условного обозначения работы.

Важно отметить, что практики рекомендуют рассматривать не только критический путь, но и путь, ближайший к критическому. Если произойдут изменения и критический путь сократится, то почти критический путь станет самым длинным, а потому и самым важным для завершения проекта.

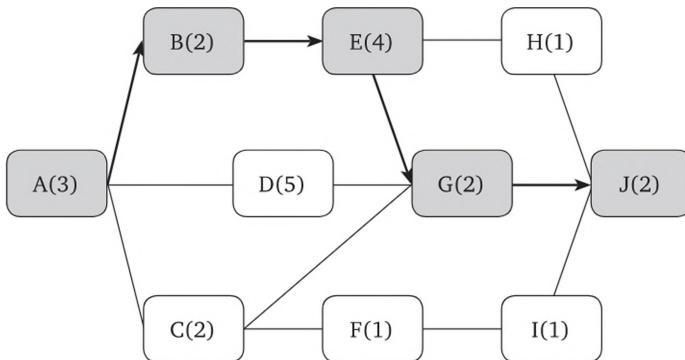


Рис. 7.4. Пример критического пути

**Временной резерв.** При оценке длительности операций важно уделять внимание временному резерву. В качестве временного резерва рассматривается время, на которое может быть задержано выполнение операции без нарушения сроков всего проекта. Есть три типа временных резервов:

- **свободный резерв** (время, на которое можно отложить одну операцию без задержки старта следующей операции). Свободный резерв = Ранняя дата старта последующей операции – Ранняя дата данной операции;

- **полный резерв** (время, на которое можно отложить одну операцию без нарушения сроков проекта или наступления контрольного события). Полный резерв = Поздняя дата окончания проекта – Ранняя дата окончания проекта.

- **резерв проекта** (время, на которое может быть отложено выполнение проекта без нарушения даты завершения, которая нужна объективно). В отличие от полного резерва, в данном случае используется более высокий уровень абстракции и рассматривается проект в целом, а не отдельные операции.

Важно понимать, что у операций критического пути нет временного резерва. Можно сказать, что любая задержка операций критического пути приводит к появлению отрицательного временного резерва.

**Разработка расписания.** Временная информация, полученная из определения критического пути и сетевой диаграммы в целом, вставляется в расписание проекта. Расписание основано на календарном плане и не сводится к простому описанию длительности.

Для разработки расписания нужно иметь:

- описание работ проекта;
- ИСР и список операций;
- последовательность операций;
- оценку ресурсов операций;
- оценку длительности каждой операции.

Разработка расписания начинается с выбора метода разработки. Чаще всего расписание основано на Методе критического пути или на Методе критической цепи. Обычно в организациях используется специальное ПО для разработки расписания. Как правило, инструментальные средства поддерживают полную функциональность, необходимую для работы с расписанием проекта.

**Модель расписания.** Обсуждение разработки расписания нужно начать с модели расписания. Модель расписания — это версия расписания, в которую вносятся данные в течение всего ЖЦ проекта. Эта модель нужна для быстрого анализа альтернатив, прогнозирования результатов и отклонений. Модель расписания используется также для анализа критического пути, профилей ресурсов, назна-

ченных задач и соответствия плану. С ее помощью прогнозируется время выполнения проекта.

Можно сказать, что модель расписания — это дорожная карта для менеджера проекта и проектной команды. С ее помощью менеджер проекта может сделать адекватные и разумные прогнозы по поводу длительности проекта. А после завершения проекта именно модель проекта является источником опыта и статистики для анализа.

Как правило, в рамках одного проекта может использоваться несколько разных версий модели расписания. Важно, чтобы они имели уникальную идентификацию, иначе будет сложно избежать путаницы. Это также важно для архивирования и аудита проекта.

**Календари и периоды выполнения работ.** Периоды выделяются планировщиком в зависимости от контекста или от требований. Они зависят также от специфики операций.

Календари включают в себя:

- количество рабочих дней в неделе;
- количество смен в рабочий день;
- количество часов в рабочем дне/смене;
- периоды сверхурочной работы;
- нерабочие периоды (праздники и т. п.).

Календарь проекта должен быть адекватным и соответствовать нормам трудового законодательства. Может использоваться как общий календарь, так и специальные календари для отдельных периодов или операций.

**Расписание проекта.** Не существует единого формата представления расписания проекта. Чаще всего расписание представляется в виде:

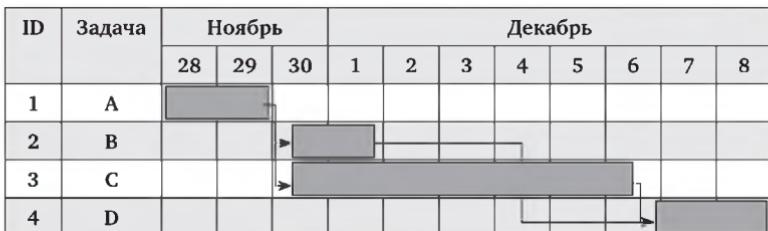
- таблицы с взаимосвязями между операциями;
- ленточной диаграммы (диаграмма Ганта) (рис. 7.5);
- сетевой диаграммы;
- диаграммы контрольных событий (рис. 7.6) и т. п.

Расписание должно основываться на длительности операций, доступности ресурсов, внешних ограничениях, сетевой логике и проверке допущений. Для подтверждения точности расписания требуется сделать прямой и обратный проходы между финишем и стартом.

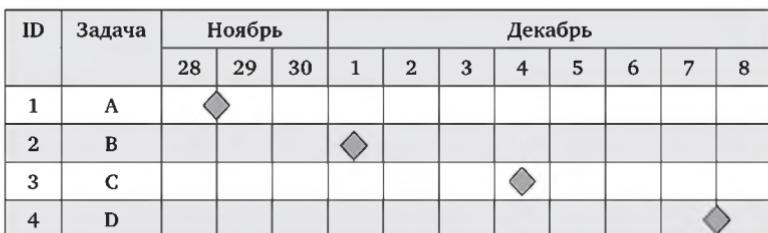
Расписание считается завершенным, когда оно согласовано, утверждено и задокументировано. Перед тем как начать процесс согласования, менеджер проекта должен:

- проанализировать приоритеты заинтересованных сторон;
- проанализировать альтернативные способы выполнения работ;
- рассмотреть взаимосвязь проекта с другими проектами;
- обеспечить выделение ресурсов на проект;
- пересмотреть календари;
- оценить реалистичность расписания;

- сжать расписание;
- пересмотреть все операции и иерархическую структуру работ;
- выровнять ресурсы.



*Рис. 7.5. Пример фрагмента расписания проекта в виде диаграммы Ганта*



*Рис. 7.6. Пример фрагмента расписания проекта в виде диаграммы контрольных событий*

На некоторых действиях остановимся подробнее. Так, выравнивание расписания по ресурсам очень важно, если проект столкнулся с жесткими ресурсными ограничениями. При помощи выравнивания расписания по ресурсам выявляются ситуации, когда ресурсы имеются только ограниченное время и (или) только в ограниченном количестве. Выравнивание расписания по ресурсам также используется в случае необходимости обеспечить выбранный уровень использования ресурсов. К примеру, до выравнивания возможна ситуация, когда согласно плану некоторый сотрудник работает одновременно над тремя задачами. После выравнивания нагрузка на сотрудника станет более реалистичной — задачи могут стать последовательными и т. п. При выравнивании расписания руководствуются принципом «Лучше удлинить проект, чем нанимать дополнительные ресурсы».

*Сжатие расписания* основано на следующих техниках:

- крашинг (привлечение дополнительных ресурсов для сокращения длительности операции);
- фаст-трекинг (распараллеливание операций на критическом пути).

**Управление расписанием.** Проект реализуется в соответствии с базовым расписанием проекта (окончательная утвержденная версия расписания). Задача менеджера проекта — измерение выполнения проекта и подготовка корректирующих действий в случае нарушения расписания. Все изменения сравниваются с базовым расписанием.

**Мониторинг и контроль расписания.** Проект может быть успешным, только когда изменения расписания отслеживаются и согласовываются. Для этого выполняются следующие действия:

- данные о статусе работ и объеме использованных ресурсов собираются и записываются;
- новые данные вводятся в модель расписания, после чего пересчитывается оставшийся объем работы, а также даты и время невыполненных работ;
- обновленная модель расписания сравнивается с базовым расписанием;
- если изменения согласованы, то расписание обновляется;
- в соответствии с планом коммуникации отправляются отчеты об обновлении расписания;
- если состав работ проекта изменился, то обновляется базовое расписание;
- создаются записи, объясняющие причины изменения расписания.

**Отчетность.** Как правило, используется несколько видов отчетности, связанной с расписанием проекта. Во-первых, это *Отчет о статусе*, который включает освоенный объем ресурсов и внесенные изменения. Во-вторых, *Отчет о прогрессе*, включающий контроль расписания и бюджета, а также ряд форм.

#### 7.1.4. Управление стоимостью проекта

**Совокупная стоимость владения (TCO).** Идея ТСО была предложена еще в 1987 г. компанией Gartner Group. С тех пор создано множество различных инструментов и методов определения значения стоимости владения системой.

---

**Совокупная стоимость владения** (англ. *Total Cost of Ownership, TCO*) — суммарная величина прямых и косвенных затрат владельца системы на протяжении всего ее жизненного цикла.

---

Однако единой методики или формулы расчета до сих пор не существует, так как структура осуществляемых затрат может быть разной в зависимости от специфики предприятия, проекта в целом и объектов расчета (оборудование, ПО, консалтинговые услуги и т. п.) в частности.

ТСО учитывает три основных типа затрат:

- прямые капитальные (CAPEX);
- прямые операционные (OPEX);
- косвенные затраты (потенциальные потери от различных плановых и внештатных ситуаций).

Отметим, что по сути косвенные затраты отражают эффективность управления информационными системами. Так, чем более эффективно ИТ-служба обеспечивает управление ИТ-инфраструктурой и системами, тем меньше косвенные затраты от простого систем, потерю времени конечных пользователей на самостоятельный поиск решения и ожидание помощи.

Однако один расчет совокупной стоимости владения без сравнения с прочими параметрами не дает полного представления о целесообразности эксплуатации системы. Логическая цепочка в данном случае выглядит следующим образом: чем большее число пользователей работает в единой системе и чем сложнее ее бизнес-процессы, тем выше как совокупная стоимость владения, так и результативность использования системы.

### Пример

При внедрении различных технологий могут учитываться следующие типы затрат:

- капитальные затраты:
  - сетевое программное и аппаратное обеспечение,
  - серверное программное и аппаратное обеспечение,
  - клиентское программное и аппаратное обеспечение,
  - стоимость лицензий (первоначальные инвестиции),
  - стоимость продления лицензий,
  - ...
- операционные затраты:
  - аренда серверов/ЦОД,
  - затраты на миграцию,
  - затраты на тестирование,
  - стоимость резервирования/копирования данных,
  - обучение,
  - внутренний и внешний аудит / консалтинговые услуги,
  - оплата труда персонала,
  - администрирование инфраструктуры;
  - ...
- косвенные затраты:
  - потенциальные потери от плановых и внеплановых сбоев программно-аппаратных платформ,
  - потери при нарушении конфиденциальности данных или при инцидентах информационной безопасности, а также затраты на восстановление деловой репутации,
  - потери времени сотрудников на помощь коллегам в решении вопросов поддержки информационных систем и пр.,

— снижение производительности (пользователи вынуждены ждать реакции системы, например, печать отчетов, что снижает эффективность их работы и приводит к денежным потерям организаций),

— ...

Возможно учитывать также затраты на модернизацию и утилизацию систем (например, масштабирование, замену программно-аппаратной платформы, увеличение производительности).

---

Выгода от реализации проекта, таким образом, может рассчитываться как разница между затратами до проекта и после его завершения.

### Пример

Для проекта *виртуализации ИТ-инфраструктуры* до и после предполагаемой его реализации могут быть рассчитаны значения таких параметров, как:

- администрирование серверов;
- восстановление серверов в случае чрезвычайной ситуации;
- затраты на незапланированный простой сервера;
- охлаждение и электропитание;
- подключение к сети;
- лицензии на VMware;
- проектирование и развертывание виртуальной ИТ-инфраструктуры;
- консалтинговые услуги по внедрению и обучению специалистов;
- ...

Некоторые затраты вырастут при реализации проекта (например, при необходимости закупки лицензий), но значительная часть снизится (стоимость аппаратного обеспечения, администрирования и пр.).

---

В целях количественного анализа используются и специальные программные средства, которые в основном реализуются на уровне электронных таблиц. При этом пользователи получают возможность сравнивать итоговые результаты с референтными значениями других компаний в отрасли и сходных масштабов проектов и систем для проведения анализа. Некоторые программные решения также позволяют осуществлять *What-If* анализ для моделирования динамики изменения ТСО при принятии того или иного решения в области ИТ (например, при консолидации серверов).

**Чистая приведенная стоимость проекта.** Смысл данного показателя состоит в вычислении текущей стоимости потоков денежных средств по периодам с последующим суммированием результатов. Риски проекта учитываются при использовании различных ставок дисконтирования (чем выше риск, тем выше будет выбранная для него ставка дисконтирования, которая может учитывать также инфляцию и прогнозную норму прибыли).

**Чистая приведенная стоимость проекта** (англ. *Net Present Value, NPV*) — текущая стоимость будущих денежных потоков инвестиционного проекта.

Всего для вычисления  $NPV$  применяются три основных шага.

1. Определение текущей стоимости затрат (объема инвестиций) через суммирование различных типов затрат.

2. Расчет текущей стоимости денежных потоков ( $PV$ ) к текущей дате:

$$PV = \sum_{t=0}^n \frac{CF_t}{(1+r)^t};$$

$$NPV = \sum_{t=0}^n \frac{CF_t}{(1+r)^t} - I,$$

где  $CF$  — денежный поток;  $r$  — ставка дисконтирования;  $I$  — инвестиции периода.

В случае осуществления инвестиций в несколько этапов применяется следующая формула:

$$NPV = \sum_{t=1}^n CF_t - \sum_{t=0}^n \frac{I_t}{(1+r)^t},$$

где  $CF$  — денежный поток;  $r$  — ставка дисконтирования;  $I_t$  — инвестиции периода  $t$ ;  $n$  — общее число периодов.

3. Сравнение текущей стоимости инвестиций в проект с текущей стоимостью доходов (табл. 7.3). Полученная разница представляет собой значение чистого дисконтированного дохода и определяется по следующей формуле:

$$NPV = PV - I_0.$$

Таблица 7.3

#### Сравнение текущей стоимости инвестиций и доходов

Значение	Эффективность
$NPV > 0$	Реализация предложенного проекта экономически эффективна, так как потенциально он принесет больше, чем требуемый процент возврата инвестиций
$NPV = 0$	Проект выгоден, так как принесет прибыль, эквивалентную требуемому проценту возврата инвестиций
$NPV < 0$	Реализация проекта экономически нецелесообразна

#### Пример

Рассчитаем чистую приведенную стоимость проекта по консолидации серверов.

Таблица 7.4

## Стоимость проекта по консолидации серверов (по годам)

Год <i>t</i>	Коэффициент дисконтиро- вания (при ставке 10%) = = $1/(1 + 0,1)^t$	Посту- пления денежных средств <i>PV</i>	Инве- стиции <i>I<sub>t</sub></i>	Чистый денежный поток <i>CF<sub>t</sub></i>	<i>NPV<sub>t</sub></i>
0	1,000	0	500 000	-500 000	-500 000
1	0,909	1 000 000	0	+1 000 000	909 091
2	0,826	750 000	0	750 000	619 835
3	0,751	500 000	0	500 000	375 657
<b>Итого</b>		<b>2 250 000</b>	<b>500 000</b>		<b>1 404 583</b>

Как можно видеть из приведенных в табл. 7.4 вычислений, несмотря на то что денежные поступления при реализации проекта будут превышать инвестиции на 1,75 млн, чистая приведенная стоимость будет меньше, порядка 1,4 млн с учетом текущей ставки дисконтирования, применяемой для каждого периода.

В MS Excel для расчета *NPV* используется функция: =ЧПС(). В английской версии: =NPV(). Ее синтаксис:

=NPV(rate; value1; [value2]...)

Здесь *rate* — ставка дисконтирования, выраженная в единицах за период (если представлены годовые потоки денежных средств, то ставка должна быть годовая); *value1*; [*value2*]... — потоки денежных средств по периодам фиксированной длины, которые могут быть представлены как диапазоны, конкретные значения через точку с запятой или несколько диапазонов.

**Возврат инвестиций (ROI).** Смысл показателя *ROI* в определении чистой прибыли от инвестиций, которая необходима для получения прибыли. Он также может говорить о том, насколько эффективно используются активы.

**Возврат инвестиций** (англ. *Return On Investment, ROI*) — отношение полученных (сэкономленных) денег от реализации проекта к вложенным в него средствам. *ROI* = 180% означает, что на каждый вложенный рубль вернулась (или была сэкономлена) сумма в 1 руб. 80 коп.

В MS Excel для расчета *ROI* используется функция =ВСД() (в английской версии =IRR()). Ее синтаксис:

=IRR(values; [guess])

Здесь *values* — диапазон с потоками денежных средств, *guess* — необязательное поле, используемое для определения ставки методом подбора. Предположение о величине ставки необходимо для начала отсчета именно от этого значения.

### Пример

Для приведенного ранее примера расчета *NPV* и значений денежного потока сумма *CF* по всем периодам составит 1,75 млн, а значит, при исходном значении инвестиций в 500 тыс. коэффициент возврата инвестиций будет равен 250 % (см. табл. 7.4).

**Период окупаемости инвестиций.** Как правило, инвесторы склонны более благоприятно относиться к проектам с более быстрым оборотом средств, нежели преследовать долгосрочные выгоды, ассоциированные с большими рисками.

**Период окупаемости инвестиций** (англ. *Payback Period, PP*) — продолжительность времени, в течение которого обеспечивается возврат первоначальных инвестиций.

Этот показатель определяется последовательным расчетом *NPV* для каждого периода проекта. Точка, в которой *NPV* станет положительным, будет являться точкой окупаемости.

### Пример

Таблица 7.5

Расчетные значения *NPV* (по годам)

Год <i>t</i>	<i>NPV</i>
0	-500 000
1	909 091
2	619 835
3	375 657
<b>Итого</b>	<b>1 404 583</b>

Так, для рассмотренного при расчете *NPV* примера точка окупаемости будет достигнута уже к первому году, когда *NPV* станет больше 0 (см. табл. 7.5).

#### 7.1.5. Управление рисками

**Выбор и реализация стратегии управления рисками.** Рассматривая важнейшие моменты управления проектами, невозможно обойти стороной риски, грамотное управление которыми способ-

но спасти проект, в то время как неверные действия по отношению к которым могут его «потопить».

---

**Риск** — неопределенное событие (условие), реализация которого влияет на ход проекта и достижение его целей.

---

В любом ИТ-проекте необходимо учитывать несколько крайне важных категорий рисков, ассоциированных как со внедрением, так и с поддержкой, технологическим совершенствованием и в целом с управлением системой:

- организационные: ассоциированные с поддержкой руководства компании и корректностью проведенного на первом этапе обследования бизнес-архитектуры компании;
- ресурсные: риски проектного управления в части «треугольника ограничений»: времени, ресурсов, качества;
- проектные: остальные риски проектного управления (в части контроля показателей проекта, управления коммуникациями, мотивацией, интеграцией);
- технологические: риски возникновения ошибок, вызванных недостаточно точным анализом и проектированием системы на уровне архитектуры приложений, данных и технологий.

Важно не только понимать, какие потенциальные ошибки можно совершить, но и когда именно следует проявлять к каждой из них особое внимание. Рассмотрим некоторые примеры в привязке к пяти этапам управления проектами по РМВоК (за основу берутся стадии проектного менеджмента, а не жизненного цикла ИС — в силу значительной зависимости рисков именно от ошибок управления).

**Инициирование.** При старте проекта крайне важна поддержка руководства компании и то, насколько высокий уровень приоритета проекта (в сравнении с другими реализуемыми в компании активностями) будет им определен. Именно здесь на первый план выходит возможный конфликт интересов бизнеса и ИТ, в случае появления которого может быть выделено недостаточно ресурсов, что повышает потенциальные «шансы» срыва сроков и (или) снижения качества проекта в дальнейшем. Также значительные риски предполагает этап обследования, на котором неверная постановка требований (в частности, из-за неэффективных коммуникаций) приведет к несоответствию итогового продукта ожиданиям и потребностям заказчика.

**Планирование.** На этапе планирования появляются новые риски, в частности нехватка необходимых компетенций у проектной команды для корректного проектирования системы и подготовки к ее реализации. Другой риск — неверная оценка сроков и объема проекта может привести к значительным задержкам проекта,

к примеру, по причине опоздания в согласовании контракта на закупку лицензий. Никогда не следует забывать также о рисках, ассоциированных со всеми контрагентами (поставщиками, подрядчиками), так как их банкротство, повышение тарифов, невыполнение договорных обязательств могут существенно сказаться на ходе реализации проекта.

**Исполнение.** Уже во время реализации системы вероятно обнаружение невозможности реинжиниринга бизнес-процессов в текущих условиях, так как их изменение потребует значительного увеличения сроков реализации либо бюджета проекта. Отсутствие мотивации у проектной команды, ошибки управления коммуникациями, рисками, интеграцией — все те риски, которые потенциально могут на этом этапе негативно сказаться на реализации проекта. Всегда существуют программные риски при приобретении ПО третьих фирм, однако еще более критичны здесь технологические риски. Ведь любые ошибки проектирования, настройки и реализации системы могут привести к потере данных, невыполнению контрактных обязательств, остановке системы (не говоря уже о последствиях ошибок для систем, выход из строя которых способен повлиять на безопасность жизни и здоровья человека).

**Мониторинг и управление.** На этапе мониторинга высокую степень важности имеет сравнение первоначальных целей/задач/ содержания проекта и полученного результата для максимально оперативного выявления и устранения замечаний. Именно поэтому столь необходимо участие грамотных специалистов в тестировании системы и обеспечение всех необходимых для процесса условий и ресурсов. Другим аспектом является вновь управление коммуникациями, только уже на этапе опытной эксплуатации и обучения пользователей, так как их участие и содействие крайне важно для последующего успеха проекта.

**Завершение проекта.** Основным риском по завершении проекта является изменение как внешних рыночных условий, так и внутренних процессов компаний. Из-за этого решение, создаваемое в ходе проекта, может потерять актуальность, а его изменение (в силу сложности и низкой гибкости архитектуры) может оказаться нерентабельным или просто невозможным.

**Сценарии действий и необходимые меры.** Перед началом проекта внедрения необходимо четко уяснить, что может помешать достижению результата, и разработать меры по предотвращению возможных негативных последствий. Например, риск заключается в следующем: на предприятии идет подготовка к внедрению корпоративной системы управления, однако многие сотрудники не обладают основами компьютерной грамотности, что может привести к существенному снижению эффективности процесса внедрения. В этом случае мерой по предотвращению риска является предвари-

тельное обучение сотрудников предприятия основам работы с базовыми компьютерными программами и проведение последующей аттестации. Иногда в ходе анализа рисков выявляются десятки факторов, которые могут повлиять на процесс внедрения, однако суть остается прежней — выявление рисков, разработка и выполнение мероприятий по их устранению, контроль над исполнением.

В первую очередь проводится *идентификация рисков*. Их список может быть сформирован руководителем проекта с командой внедрения и в дальнейшем согласован с куратором проекта. Этот список может быть представлен в форме таблицы и сгруппирован по этапам ЖЦ системы, по этапам управления проектом внедрения, по основным классам рисков и другим видам группировки. Часто карта рисков составляется на основе проведенного бизнес-моделирования (карты бизнес-процессов).

Следующим шагом проводится *оценка рисков*, которые возможно представить в виде таблицы или матрицы, с обязательным описанием риска, его возможных последствий в качественном и количественном выражении (либо денежный эквивалент ущерба от реализации, либо оценка по шкале от 1 до 5). Вычисление данного значения чаще всего проводится на основании предварительного анализа вероятных потерь (от временных и денежных ресурсов до ущерба репутации компании, потери рыночной доли и т. п.). На этой стадии возможно использование анализа чувствительности, сценарного анализа, имитационного моделирования Монте-Карло, «bow-tie» анализа причин, событий и последствий реализации рисков, и прочих специальных методов.

Затем указывается *вероятность реализации риска (%)*, а также возможные меры по снижению этой вероятности либо уменьшению потенциального ущерба. В самом простейшем варианте для получения итоговой оценки для каждого риска необходимо умножить значение Ущерба от реализации на Вероятность осуществления риска (переведя % в значение от 0 до 1). После ранжирования результатов и выделения рисков с наиболее высоким приоритетом назначаются ответственные за управление данным риском и принятие необходимых мер.

Существует несколько *стратегий минимизации* рисков. Сценарии действий по управлению рисками и примеры принимаемых по этим сценариям мер приведены в табл. 7.6.

И наконец, заканчивая разговор о рисках, упомянем столь важный аспект, как *«положительные» риски*. К примеру, в то время как «обычным» риском будет являться уход из компании ключевых сотрудников, «позитивным» возможным риском будет участие в проекте признанного в своей предметной области эксперта, у которого неожиданно освободится время. Рассмотрим ту же матрицу сценариев действий, только уже для позитивных рисков (табл. 7.7).

Таблица 7.6

## Сценарии действий и необходимые меры

Сценарий действий	Необходимые меры
Снижение риска	<p>Уменьшение как вероятности его наступления, так и возможных негативных последствий.</p> <p><b>Пример</b></p> <p>Так, для снижения риска недостоверности предоставленных на этапе обследования данных можно фиксировать каждый факт предоставления информации в формате аудио- и видеозаписей встреч, сохранения полученных и переданных сообщений, подписания протоколов совещаний со списком принятых решений и пр. Еще в данном случае возможно привлечение квалифицированных специалистов со значительным и релевантным (!) опытом реализации подобных проектов (в предметной области либо индустрии), а также подготовка шаблонов заполнения и загрузки информации для минимизации вероятности ошибки</p>
Перенос риска	<p>Перенос риска на другую сторону при заключении отдельных соглашений.</p> <p><b>Пример</b></p> <p>Заключение договоров страхования, аутсорсинга и прочие связанные с третьими сторонами активности. Так, потери от срыва работ в рамках контракта с подрядчиком/поставщиком могут быть (по крайней мере, частично) покрыты соответствующим пунктом договора об ответственности за неисполнение его условий. Убытки от пожара в серверной (или какая-то их часть) могут быть покрыты страховой компанией.</p> <p>Чаще всего исключением (риском, перенос которого почти невозможен) является ущерб деловой репутации компании и ее бренду</p>
Избежание риска	<p>В определенной степени наиболее «практичный» сценарий, предусматривающий реализацию необходимых проектных активностей (возможно, в чуть большем объеме или в более сжатые сроки) во избежание многих проектных, организационных и технологических рисков.</p> <p><b>Пример</b></p> <p>Ежедневная отчетность для устранения риска слабой организованности участников проекта, фиксация числа итераций и составление графика согласования и утверждения решений и результатов во избежание риска слишком длительного процесса оформления документов</p>
Принятие риска	<p>Занятие «позиции боевой готовности», если риск выявлен слишком поздно или уже реализован, определение способов покрытия ожидаемого ущерба и избежания подобных ситуаций в будущем.</p>

Сценарий действий	Необходимые меры
	<p><b>Пример</b></p> <p>Потеря данных при проблеме с data-центром в качестве реализовавшегося (возможно, не идентифицированного своевременно) риска. В этом случае в качестве реактивных мер можно активировать последние сделанные резервные копии, восстановить работоспособность системы, а в дальнейшем даже пересмотреть прежнюю политику компании в отношении достаточного уровня надежности используемого ЦОД, а также обновить (создать) стратегию обеспечения непрерывности бизнеса</p>

Таблица 7.7

## Сценарии действий и необходимые меры при позитивных рисках

Сценарий действий	Пример
Усиление (vs снижение) риска	Новость о возможном (при совпадении графиков) участии в качестве консультанта известного эксперта является позитивным риском, однако для усиления вероятности его реализации необходимо пересмотреть график проведения семинаров и встреч, чтобы поменять их расписание в соответствии со свободным временем эксперта
Перенос риска	Если компания — системный интегратор, работающая в России с программными продуктами Microsoft, создаст на их базе отраслевое решение для определенного локального рынка, которое будет пользоваться популярностью, Microsoft может предложить дальнейшее сотрудничество на льготных условиях
Использование (vs избежание) риска	При наличии возможности получения скидки вендора при закупке определенного объема лицензий (который ранее предполагалось приобрести двумя этапами) организация может изменить первоначальный план закупок и согласовать новый контракт на больший объем лицензий
Принятие риска	Положительным риском может быть расширение числа пользователей внедренной системы. Допустим, изначально планировалось использовать автоматизированную систему управления проектами только в головном офисе, но филиалы (дочерние компании), увидев все ее преимущества, могут также принять решение адаптировать данный программный продукт для своих нужд. Таким образом, сама организация будет иметь возможность реализовывать проекты в совместной работе нескольких филиалов, а компания — системный интегратор (либо внутреннее ИТ-подразделение) получит новый проект для выполнения

Риск всегда является неотъемлемой частью любого сложного и ответственного процесса. Более того, совершение рискованных действий необходимо для прогресса, а ошибки, как известно, являются основой приобретения опыта. Несмотря на то что некоторые риски в ходе реализации проекта внедрения системы неизбежны, это не означает, что попытки определить их и управлять ими вредят творческой работе.

**Обеспечение непрерывности бизнеса.** Чтобы ИТ-инфраструктура не просто соответствовала потребностям бизнеса, но и обеспечивала непрерывность доступа к системе и предоставляемым сервисам, необходимо также предусмотреть концепцию (стратегию) непрерывности. В этих целях можно использовать серверные технологии в различных конфигурациях (рис. 7.7).

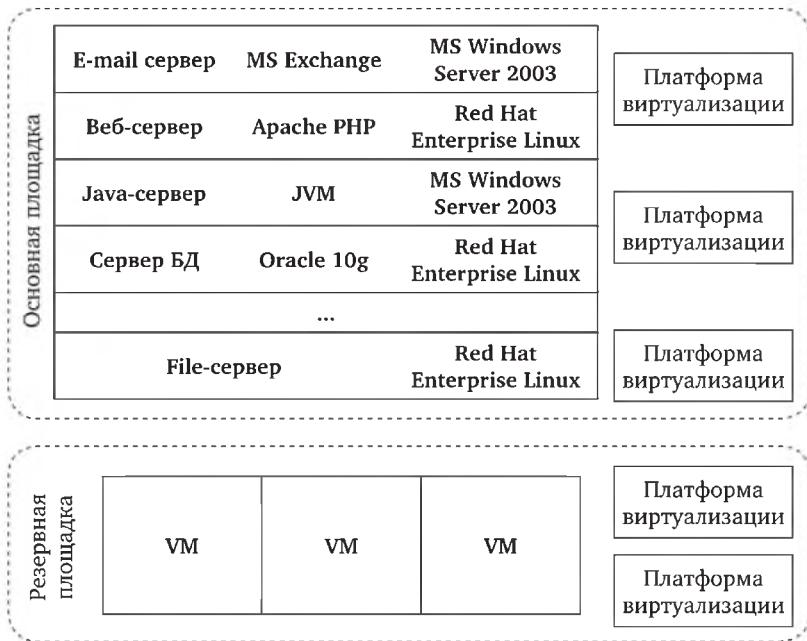


Рис. 7.7. Пример осуществления резервирования

В зависимости от ситуации технологические площадки:

- могут быть *территориально распределены* или локализированы;
- могут иметь *полный или частичный резерв оборудования* или не иметь его вообще:
  - в случае полного резерва при катастрофе на технологической площадке предоставление ИТ-услуги (приложения) обеспечивается за счет оборудования на второй площадке,

- в случае частичного резерва основное вычислительное оборудование установлено на одной площадке,
- если резервирование не осуществляется, то восстановление на новом оборудовании производится с резервных внешних носителей информации;
- могут иметь *кластерную конфигурацию* (распределение нагрузки между равнозначными активными серверами приложений).

Возможен и другой вариант: построение резервной площадки «в облаке» (основанной на SRM). Таким образом, на основной площадке обеспечивается локальная доступность, реализуются решения по защите данных, а на резервной площадке организуется аварийное восстановление.

### Пример

#### **Территориально распределенная конфигурация с подготовкой вычислительных ресурсов в виде «холодного» резерва**

Данная концепция предполагает подготовку резервной серверной площадки, включая всю необходимую серверную инфраструктуру, а также предварительную подготовку и размещение на ней набора оборудования, идентичного оборудованию основного ЦОД.

Основные и резервные вычислительные ресурсы работают в режиме взаимного резервирования, при котором часть оборудования на каждой из серверных площадок имеет запас вычислительной мощности. Резервные вычислительные мощности находятся в режиме простоя либо используются для решения второстепенных задач.

На резервной площадке устанавливаются дополнительные системы хранения данных и прочие технические средства резервного копирования данных с сессиями репликации данных с основным ЦОД периодичностью 1 раз в сутки.

При выборе конкретной стратегии непрерывности и восстановления принимаются во внимание следующие аспекты:

- уровень критичности приложения (системы), исходя из которого определяется допустимое время восстановления;
- режим восстановления (ручной/автоматический);
- стоимость реализации;
- сложность/время реализации.

Подход к реализации стратегии непрерывности бизнеса, как правило, закрепляется в плане обеспечения непрерывности (BCP).

---

**План обеспечения непрерывности бизнеса** — документ, предназначенный для выработки и соблюдения мер по минимизации рисков системных сбоев информационных систем компании и снижения возможных их последствий для бизнеса.

---

BCP направлен на обеспечение поддержки бизнес-процессов компании во время и после чрезвычайной ситуации и может охватывать либо один отдельный процесс, либо совокупность нескольких. ИТ-системы рассматриваются в BCP исключительно с точки зрения поддержки бизнес-процессов. BCP может даже не рассматривать вопрос восстановления процессов, фокусируясь только на временных требованиях непрерывности бизнеса. Основная цель подобного документа — своевременное определение вариантов действий в различных критических ситуациях. Это позволяет снизить объем финансового ущерба, обеспечивает более быстрое восстановление, а также удовлетворяет требованиям всех заинтересованных сторон (клиентов, акционеров, регуляторов, руководителей) к непрерывности работы компании.

Однако список задач по формированию плана обеспечения непрерывности бизнеса охватывает большее количество аспектов.

1. **Определение всех рисков для бизнеса** с формированием плана работ по каждому из данных рисков (в том числе ответственные сотрудники, меры предотвращения реализации рисков и нивелированию их негативного воздействия).

Так как план учитывает множество бизнес-факторов (например, репутационные риски), то как сам список рисков, так и список вариантов их снижения должны постоянно актуализироваться. Любые изменения в бизнес-процессах и бизнес-сервисах после их реализации должны быть отражены в BCP-плане.

2. **Определение критических сервисов и приложений**, параметров целевого времени восстановления (*recovery time objective, RTO*) и целевой точки восстановления (*recovery point objective, RPO*), стоимости каждого часа простоя.

### Пример

Как правило, подавляющее большинство приложений требует значения *RTO*, не превышающего 4 ч, и значения *RPO* менее 2 ч.

3. **Принципы переноса данных** между основной и резервной площадками, выбор оптимального решения (в том числе с рассмотрением удаленности резервной площадки, пропускной способности каналов связи).

4. **Формирование процесса/процедур восстановления сервиса** в случае наступления критической ситуации — разработка плана.

**План аварийного восстановления** (*Disaster recovery plan, DRP*) — план действий по восстановлению полной работоспособности информационной системы (в том числе на альтернативных площадках).

План аварийного восстановления, как правило, фокусируется на тех ситуациях, когда доступ к информационным системам отсутствует/ограничен в течение длительного периода. В отличие от BCP, план DRP существует на более операционном уровне, описывая порядок действий (организации технических и организационных решений) по построению системы защиты инфраструктуры ИТ. Так, среди наиболее популярных решений по защите от чрезвычайных ситуаций — построение резервной площадки или ЦОД (*disaster recovery site*).

В качестве основных методологических материалов разработки планов обеспечения непрерывности и аварийного восстановления используются следующие международные стандарты и руководства:

- ISO/IEC 27001:2005 (ISO/IEC 17799:2005);
- сборник лучших практик BCI (*The Business Continuity Institute*);
- руководство CobiT;
- Contingency Planning Guide for Information Technology Systems (NIST).

Для обеспечения непрерывности бизнеса, как правило, выполняется комплекс мероприятий:

- разработка политики непрерывности бизнеса — определение требований и основных принципов;
- анализ влияния на бизнес BIA (*business impact analysis*) — определение критических ИТ-ресурсов, приоритетов восстановления;
- разработка и внедрение превентивных мер;
- определение стратегии восстановления;
- создание плана обеспечения непрерывности бизнеса BCP;
- тестирование и обучение персонала;
- эксплуатация плана BCP, в том числе мониторинг реализации и обновление плана, взаимодействие с внешними контрагентами по вопросам непрерывности бизнеса.

**Обеспечение информационной безопасности.** Цель информационной безопасности — обеспечить бесперебойную работу организации и свести к минимуму ущерб от событий, таящих угрозу безопасности, посредством их предотвращения и сведения последствий к минимуму. С точки зрения безопасности все виды информации, включая бумажную документацию, базы данных, информационные системы, съемные носители, разговоры и другие способы, используемые для хранения и передачи знаний и идей, требуют надлежащей защиты. Информация и поддерживающие ее информационные системы и сети являются цennыми производственными ресурсами организации. Их доступность, целостность и конфиденциальность является приоритетными направлениями программы развития информационной безопасности.

В основе развития информационной безопасности на предприятиях часто находится риск-ориентированный подход, который под-

разумевает оптимальный набор средств защиты в соответствии с уровнем угрозы и возможными последствиями, определенными в ходе анализа рисков. Контроль и обеспечение защиты осуществляется на всех уровнях системы и предприятия в зависимости от потенциальных угроз и включает следующие механизмы.

**Сетевая безопасность.** На этом уровне реализуются задачи защиты внешнего периметра и локальной сети:

- потоковое сканирование входящего трафика;
- IPSEC/SSL VPN сервера, в том числе поддержка мобильных устройств, удаленного доступа, токенов, RSA-сертификатов;
- выявление и блокирование атак, вирусов, вредоносного ПО;
- предотвращение вторжений;
- веб-фильтрация по предустановленным категориям трафика между объектами ЛВС и глобальной сетью.

При выборе конкретных программных средств для защиты внешнего периметра в первую очередь важны критерии пропускной способности, высокой доступности сервиса и устойчивости к атакам на отказ в обслуживании.

Пользователи подсети более низкого уровня безопасности не должны иметь доступ в сеть более высокого уровня, пользователи сетей одинакового уровня безопасности не должны иметь доступ в сети друг друга без явного разрешения в политике безопасности, пользователи сетей более высокого уровня безопасности должны иметь возможность взаимодействия с остальными частями сети.

При построении беспроводных сетей должны использоваться протоколы с поддержкой механизмов аутентификации пользователей, взаимной аутентификации беспроводного устройства и точки доступа, шифрования передаваемых данных, а также обязательное физическое разделение беспроводных и локальных сегментов, с направлением трафика беспроводного сегмента на выделенный интерфейс межсетевого экрана. При организации гостевого сегмента его пользователям может быть разрешен только доступ в публичные сети.

При подключении в локальную сеть каждый пользователь должен проходить обязательную проверку на наличие работающего антивируса, актуальности базы антивируса, установки критичных обновлений операционной системы и т. д.

**Защита серверов и рабочих станций.** Защита от несанкционированного доступа и воздействия вредоносного ПО на пользовательские и системные файлы. Защита серверного оборудования, рабочих станций и мобильных устройств:

- антивирусная защита;
- межсетевое экранирование трафика на конечном устройстве;
- контроль целостности файлов.

**Контроль и управление доступом.** Создание централизованной системы идентификации и управления доступом пользователей к корпоративным информационным ресурсам:

- централизованное ведение учетных записей пользователей;
- поддержка ролевой модели предоставления доступа пользователям;
- поддержка механизмов согласования со средствами локального администрирования систем;
- поддержка Single sign-on, многофакторной аутентификации и пр.

**Обеспечение конфиденциальности данных.** Обеспечение конфиденциальности данных, обрабатываемой и хранящейся в информационно-телекоммуникационных системах (в частности, в области коммерческой тайны и персональных данных сотрудников, клиентов, контрагентов):

- использование меток данных;
- шифрование данных;
- управление сертификатами ЭЦП;
- мониторинг доступа пользователей к данным;
- контроль печати конфиденциальных документов и записи информации на внешние носители данных.

**Защита от утечки данных.** Функции контентной фильтрации трафика, URL-фильтрации, фильтрации e-mail, а также контроль съемных носителей и печати конфиденциальных документов.

**Построение централизованной системы управления и мониторинга средств информационной безопасности.** Внедрение совокупности решений в централизованную систему мониторинга для обеспечения контроля событий ИБ и оценки защищенности инфраструктурных компонентов:

- устройств сетевой безопасности;
- конечных устройств;
- системы централизованного управления доступом;
- dlp-компонентов;
- системы сканирования уязвимостей;
- прочие средства безопасности.

Проводятся также следующие действия:

- инвентаризация информационных активов;
- ведение и анализ журналов (логов) сетевого оборудования, межсетевых экранов, операционных систем, СУБД, приложений;
- аудит управления обновлениями;
- тестирование на проникновение.

При реализации инициатив необходимо не только провести интеграцию отдельных технических решений, но и принять ряд организационных мер:

- разработать соответствующее методологическое обоснование (детальную политику безопасности, роли и матрицу доступа пользователей всех приложений, классификацию инцидентов информационной безопасности и т. д.);
- подготовить организационно-распорядительные документы по управлению и взаимодействию в области информационной безопасности;
- зафиксировать зоны ответственности подразделений.

### 7.1.6. Управление качеством

На сегодняшний день существует бесчисленное множество определений понятия «качество», в частности, для различных отраслей бизнеса. Одно из самых употребляемых определений было сформулировано в 1994 г. в Международном стандарте ISO 8402:94 «Управление качеством и обеспечение качества».

---

**Качество** — совокупность характеристик объекта, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

---

Управление качеством системы является комплексной задачей, для успешной реализации которой можно, во-первых, фокусироваться на производстве качественного продукта изначально, а во-вторых, проводить своевременную оценку и корректировку уровня качества.

В первом случае мероприятия направлены на обеспечение соответствия проекта и ИС как таковой ранее определенным требованиям.

*Использование стандартов (управления проектами, разработкой программного обеспечения, управления ЖЦ и т. п.)* (рис. 7.8). Разумеется, опыт аналогичных проектов многих компаний и «лучшие мировые практики» дают общие руководства по управлению качеством и предлагают различные методики и инструменты, использование которых разумно и целесообразно. Однако необходимо понимать, что для их эффективного применения необходимо обладать соответствующим опытом и грамотно комбинировать разные стандарты во избежание их конфликтов между собой и несовместимости со спецификой предприятия или проекта.

*Обучение, повышение квалификации персонала, использование внешних консультантов.* Повышение уровня компетенций проектной команды является широко применяемым и даже обязательным условием получения качественных результатов.

*Четкие критерии качества и стабильная внешняя среда проекта.* Вероятность повышения качественного продукта повышается, если критерии оценки информационной системы заранее четко

сформулированы и прозрачны, а рамки проекта остаются неизменными, без уменьшения сроков или стоимости.

В случае же оценки и корректировки качества очень важны следующие меры.



Рис. 7.8. Использование стандартов на разных этапах цикла PDCA<sup>1</sup>

*Прототипирование и тестирование с пользователями.* Уже рассмотренное ранее в теме по ЖЦИС прототипирование существенно снижает риски получения некачественного или несоответствующего требованиям продукта, и своевременная проверка функциональности с пользователями при тестировании также способствует выявлению неверных архитектурных, функциональных или интерфейсных решений и их последующей корректировке.

*Регулярный контроль и анализ.* Контрольные карты, диаграмма Ишикавы для причинно-следственного анализа, анализ план/факт, планы совершенствования качества и процессов и другие методы управления проектами в зависимости от ситуации оказываются достаточно полезными в целях мониторинга качества.

<sup>1</sup> Цикл Деминга, цикл PDCA (от англ. *Plan-Do-Check-Act*) — циклически повторяющийся процесс принятия решений, использующийся в управлении качеством.

Рекомендации по управлению качеством систем описаны в ряде стандартов.

- ISO 9000:2005 «Системы менеджмента качества. Основные положения и словарь».
- ISO 9001:2008 «Системы менеджмента качества. Требования».
- ISO 9004:2009 «Менеджмент с целью достижения устойчивого успеха организации. Подход с позиции менеджмента качества».
- ISO 10001:2007 «Менеджмент качества. Удовлетворенность потребителя. Руководящие указания по кодексу поведения для организаций».
- ISO 10005:2005 «Системы менеджмента качества. Руководящие указания по планам качества».
  - ISO 10006:2005 «Системы менеджмента качества. Руководящие указания по менеджменту качества проектов».
  - ISO 10007:2003 «Системы менеджмента качества. Руководящие указания по менеджменту конфигурации».
  - ISO/TR 10013:2001 «Рекомендации по документированию систем менеджмента качества».
  - ISO/TR 10014:2006 «Менеджмент качества. Руководящие указания по реализации финансовых и экономических выгод».
  - ISO 10015:1999 «Управление качеством. Руководящие указания по обучению».
  - ISO 19011:2011 «Руководящие указания по аудиту систем менеджмента».

#### **ISO 10006:2003 (ГОСТ Р ИСО 10006—2005)**

**Международный стандарт:** ISO 10006:2003 «Управление качеством. Руководящие указания по менеджменту качества проектов».

**Российский аналог:** ГОСТ Р ИСО 10006—2005 «Системы менеджмента качества. Руководство по менеджменту качества при проектировании».

Стандарт акцентирует внимание своей аудитории на двух основных аспектах качества в управлении проектами: качестве процессов проекта и качестве продукции, так как недостаточное внимание или неверная реализация требований хотя бы к одному из них может оказывать негативное воздействие на ход проекта, его результаты, саму организацию.

Стандарт ISO 10006:2003 содержит собственные определения многих проектных терминов. Кроме того, ISO 10006:2003 описывает организационный аспект управления качеством в части ответственности руководства, затем переходя к управлению ресурсами, изготовлением продукта, и непосредственно к качеству (измерениям, анализу и улучшениям). Еще один раздел стандарта (глава «Системы менеджмента качества при проектировании», со ссылкой на ISO 9000) приводит восемь основных принципов: ориентацию

на потребителя, лидерство руководителя, вовлечение работников, процессный подход, системный подход к менеджменту, постоянное улучшение, принятие решений на основании фактов и взаимовыгодные отношения с поставщиками.

В табл. 7.8 перечислены основные процессы, выделяемые стандартом.

Таблица 7.8

**Процессы стандарта ISO 10006:2003**

Раздел	Процессы	Комментарии
Ответственность руководства	Стратегический процесс управления	Обзор этапов и принципов менеджмента качества
Управление ресурсами	Процессы, связанные с ресурсами	Принципы планирования ресурсов и их контроля (сравнение план/факт, примеры корректирующих мер)
	Процессы, связанные с персоналом	Установление организационной структуры проекта, выбор персонала с необходимыми компетенциями и его последующее развитие
Изготовление продукции	Взаимозависимые процессы	Работа с требованиями стейкхолдеров проекта, управление взаимодействием и получением обратной связи
	Процессы, связанные с областью применения	Определение концепции и результатов применения разрабатываемого продукта (системы), документирование и последующий контроль действий, необходимых для достижения целей проекта
	Процессы, связанные со временем	Определение зависимостей между проектными активностями и ограничений задач типа <i>Start no later than, Must finish on</i> , оценка их продолжительности и формирование графика проекта с контролем выполнения на основе полученных данных
	Процессы, связанные со стоимостью	Оценка стоимости проекта с составлением, согласованием и контролем бюджета
	Процессы, связанные с обменом информацией	Формирование системы информирования участников проекта

Окончание табл. 7.8

Раздел	Процессы	Комментарии
	Процессы, связанные с риском	Идентификация потенциальных рисков, оценка возможных последствий, разработка планов реагирования и контроль
	Процессы, связанные с закупкой	Составление плана закупаемой продукции, технических требований, формирование критериев выбора поставщиков, а также процессы контрактной работы
Измерение, анализ и улучшение	Процессы, связанные с улучшением	Определение «слабых мест» в части управления проектом и принятие корректирующих мер
	Измерение и анализ	Сбор и оценка корректности данных
	Постоянное улучшение	Распределение ответственности заказчика и подрядчика в части совершенствования процессов проекта

Непосредственно же текст стандарта содержит основные моменты, которым следует уделять внимание для успешной реализации проектов.

### Пример

#### Документирование требований к закупкам

В документах на закупку должны быть идентифицированы продукция, ее характеристики, соответствующие требования системы менеджмента качества и связанная с ними документация. Документы должны также предусматривать ответственность покупателя, определять стоимость и дату поставки продукции, требования аудита (при необходимости) и права доступа в помещения поставщика. Требования заказчика должны быть отражены в документах на закупку.

Документы (например, «Запрос расценок») должны быть построены так, чтобы упростить сопоставимые и полные ответы потенциальных поставщиков.

Документы на закупку должны быть рассмотрены до выполнения закупок, чтобы проверить, что все связанные с изделием требования и другие аспекты (такие как ответственность покупателя) полностью определены.

Рассмотрим еще один стандарт, концентрирующийся на одном из двух рассмотренных в ISO 10006:2003 аспектов: качестве процессов. Важным фактором обеспечения качества является повышение степени развития соответствующих процессов внутри самой организации-заказчика и ее подрядчиков. Чем раньше будут иден-

тифицированы пути повышения зрелости данных процессов и предприняты необходимые меры, тем больше вероятность реализации проектов с высоким уровнем качества результатов. В целях оценки уровня зрелости процессов управления компанией в целом и процессов управления разработкой программных решений в частности применяются описанные ниже стандарт ISO 15504 и модель CMMI.

### ISO 15504/SPICE

**Международный стандарт:** ISO 15504:2004 «Information Technology. Process Assessment» / SPICE (Software Process Improvement and Capability Determination).

**Российский аналог:** ГОСТ Р ИСО/МЭК 15504—2009 «Информационные технологии. Оценка процессов».

Стандарт, впервые представленный в 1990-х гг., отличает фокус на модели улучшения процессов, а именно: эффективности работы и возможностей процесса. Под эффективностью работы понимаются продуктивность и учет потребностей пользователей (бизнес-заказчика), а возможности процесса определяются, как правило, организацией — поставщиком услуг по разработке и сопровождению системы.

Стандарт содержит пять частей:

- Часть 1. Основные понятия и словарь;
- Часть 2. Проведение оценки;
- Часть 3. Руководство по проведению оценки;
- Часть 4. Руководство по применению для улучшения процесса и определения возможностей;
- Часть 5. Пример модели оценки процесса, основанной на ИСО/МЭК 12207, Дополнения 1 и 2.

Структура стандарта приведена в схематичном виде на рис. 7.9.

Раздел «Внешнее» содержит элементы, которые не входят в стандарт ИСО/МЭК 15504, однако связаны с его частями.

**Модели уровней зрелости СММ/CMMI.** Данный стандарт исходит из концепции, согласно которой существуют определенные критерии/признаки уровня развития процессов организации. В некотором роде он опирается на методологию зрелости систем управления СММ и ее модификацию для оценки процессов разработки ПО — СММI.

---

**CMM (Capability Maturity Model)** — методология, предложенная в конце 1980-х группой специалистов Университета Карнеги — Меллон и призванная сформировать систему критериев выбора организаций-подрядчиков для цели проектов разработки ПО. В связи с этим она называлась SE-CMM (Software Engineering Capability Maturity Model). А ее переработанная версия, посвященная целиком зрелости процессов разработки программного обеспечения, получила название **CMMI** (Integrated CMM).

---

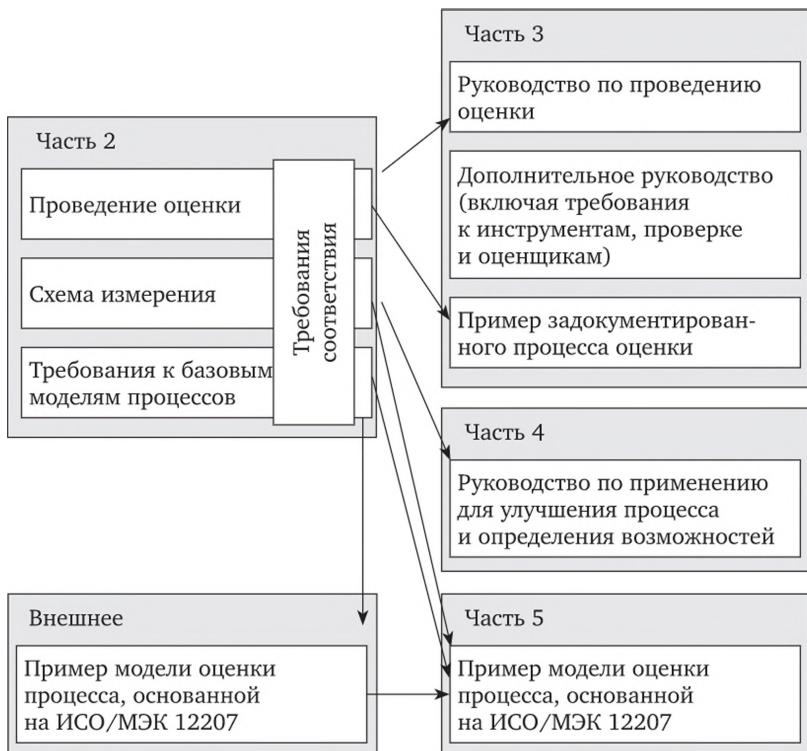


Рис. 7.9. Обзор взаимодействия между элементами серии стандартов ИСО/МЭК 15504

Модели СММ/CMMI базируются на трех основных предположениях, которые применимы также и к стандарту ISO 15504.

1. Существуют несколько уровней зрелости проектной организации. Эти уровни зависят от ряда критериев и применимы также к разрабатывающим и внедряющим системы организациям. Всего данных уровней выделяется пять.

2. Любая организация заинтересована в переходе на более высокий уровень зрелости.

Это важно не только для выживания и сохранения конкурентоспособности на рынке, но и для повышения внутренней эффективности и качества выполнения проектов.

3. Переход возможен только на следующий уровень.

Нельзя «перескакивать» через уровни, так как одновременное внедрение большого числа мер для повышения зрелости резко повысит риски и может привести к непредсказуемым последствиям в случае неготовности организации к переходу.

Рассмотрим описание уровней зрелости SE-CMM в их первоначальном виде.

**Уровень 1 — Начальный (*initial*).** Процесс разработки ПО представляет собой «черный ящик», не существует стандартов планирования и контроля проектов. Успех подобных проектов зависит целиком от компетентности, мотивации и героизма сотрудников, а не от применения отработанных процессов.

Схематично данный уровень зрелости представлен на рис. 7.10.

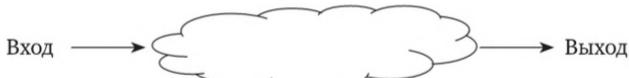


Рис. 7.10

**Уровень 2 — Повторяемый (*repeatable*).** Созданы базовые способы контроля проектов: планирование и мониторинг времени, стоимости, содержания и качества (рис. 7.11).

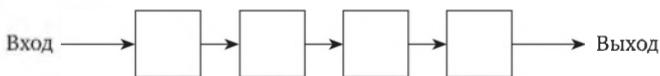


Рис. 7.11

**Уровень 3 — Определенный (*defined*).** В организации документированы и внедрены стандартные процессы. Внутренняя структура «черного ящика» становится видимой (рис. 7.12). Менеджеры и команда проектов хорошо понимают свои роли и обязанности в процессах разработки.

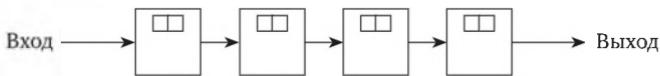


Рис. 7.12

**Уровень 4 — Управляемый (*managed*).** На данном уровне организацией определяются количественно измеряемые цели по достижению качества как программных продуктов, так и процессов. На основании получаемых при измерении результатов принимаются решения (рис. 7.13).

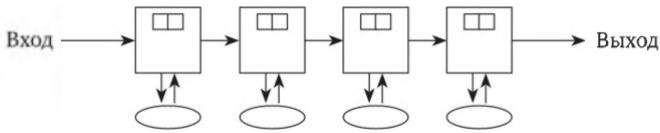


Рис. 7.13

**Уровень 5 — Оптимизирующий (*optimizing*).** Проактивное устранение существующих недостатков и «узких мест» процессов,

их постоянный поиск и реализация возможностей по совершенствованию этих процессов (рис. 7.14).

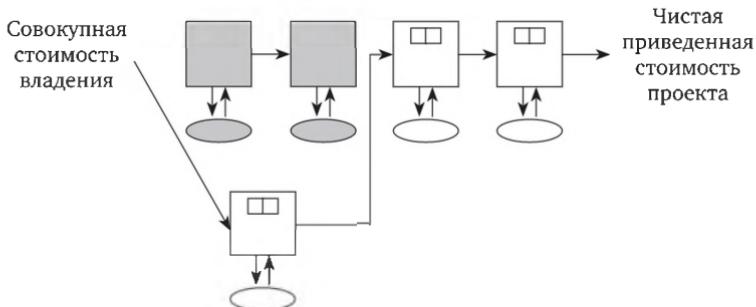


Рис. 7.14

Рассмотрев основные условия и особенности моделей зрелости SE-CMM, перейдем к конкретным описаниям уровней по CMMI (ISO 15504) в отношении процессов разработки. CMMI отличают более детализированные описания и использование не только терминологии пяти уровней зрелости, но и концепции «непрерывной модели» оценки процессов (дополнительные введенные на каждом уровне метрики: например, РА 2.1 «Управление производительностью»). Для большей наглядности и демонстрации применимости модели CMMI как части стандарта ISO 15504 к процессам создания и эксплуатации системы приведем по уровням соответствующие комментарии.

**Уровень 1 — Выполняемый (performed).** РА 1.1 Исполнение процесса. При разработке ПО используются типовые инструменты, основные процессы ЖЦ не регламентированы и исполняются в соответствии с квалификациями и опытом специалистов. Сроки неопределены в связи с сильной зависимостью от конкретной ситуации и отсутствием методик и регламентов в качестве базы.

**Уровень 2 — Управляемый (managed).** РА 2.1 Управление производительностью и РА 2.2 Управление продуктом. С ростом сложности проектов по разработке и внедрению систем, ростом требований к квалификации специалистов появляются процессы управления процессом и самим продуктом. Проводится документирование проводимых работ и предпринимаются первые шаги по регламентированию и формированию методик разработки, оценки и сопровождения систем.

**Уровень 3 — Устоявшийся (established).** РА 3.1 Определение процесса и РА 3.2 Становление процесса. Третий уровень зрелости процессов жизненного цикла программных средств предполагает их стандартизацию, возможность доработки с учетом текущих особенностей конкретного проекта. К этому моменту процессы уже

описаны, и для них сформированы четкие входные и выходные параметры, результаты, требования и условия. Определены рекомендуемые к использованию с учетом конкретной ситуации программные средства и инструменты.

**Уровень 4 — Предсказуемый (*predictable*).** РА 4.1 Измерение/оценка процесса и РА 4.2 Контроль/мониторинг процесса. На данном уровне зрелости процессов ЖЦ организаций уже способны реализовывать крупномасштабные проекты внедрения, даже с учетом жестких ограничений сроков, стоимости и заданного уровня качества. Соответственно, управление на четвертом уровне модели предполагает сформированные количественные показатели и характеристики процессов, для которых проводится мониторинг с последующим использованием полученной статистики и результатов для улучшения управления проектами и во избежание ошибок.

**Уровень 5 — Оптимизирующий (*optimizing*).** РА 5.1 Инновации процесса и РА 5.2 Непрерывная оптимизация. Наконец, на «Оптимизирующем» уровне зрелости возможно снижать совокупную стоимость создания и владения системой за счет своевременного планирования, проведения ретроспективного анализа и использования новых, более эффективных инфраструктурных ресурсов, методик организации управления проектом, выбора оптимальных с точки зрения качества, стоимости и соответствия требованиям программных решений.

Для каждого пункта вышеприведенного списка проводится оценка степени выполнения того или иного процесса. Итоговая шкала состоит всего из четырех значений:

- не реализован (0—15 %);
- частично реализован (>15—50 %);
- в значительной степени реализован (>50—85 %);
- полностью реализован (>85—100 %).

#### 7.1.7. Управление командой проекта

При привлечении подрядчика к внедрению проектная команда будет состоять из специалистов с обеих сторон: как заказчика, так и исполнителя. Пример состава участников от заказчика и исполнителя приведен ниже.

От заказчика:

- Координатор проекта;
- Менеджер (руководитель) проекта;
- Специалисты АСУ;
- Системный администратор;
- Системный архитектор;
- Консультанты-программисты.

От исполнителя:

- Куратор проекта;

- Менеджер (руководитель) проекта;
- Администратор проекта;
- Бизнес-аналитики;
- Консультанты по функциональности;
- Консультанты по постановке задач;
- Технические консультанты;
- Программисты-консультанты;
- Преподаватель курсов (проведение обучения по системе).

И кураторы, и менеджеры проекта, принимая самое непосредственное участие в принятии высокуюровневых решений по управлению проектом внедрения, как правило, являются членами Координационного совета или другого органа корпоративного управления ИТ со стороны Заказчика для определения стратегических, финансовых и организационных вопросов, а также согласования и утверждения результатов работы команды внедрения по проекту. Заседания подобного органа управления проводятся по мере необходимости, в зависимости от масштаба и сложности проекта (чаще всего на ежемесячной основе).

Опишем последовательно все основные проектные роли с обоих сторон.

*Со стороны исполнителя.* Куратор проекта — как правило, представитель топ-менеджмента компании, участвует в активностях по оформлению договора на выполнение проекта, осуществляя общее руководство проектом и определяет стратегические аспекты взаимодействия специалистов и компаний, занятых в нем.

Среди его основных задач в проекте:

- участие в работе Координационного совета;
- принятие оперативных административных решений, касающихся компании-исполнителя, контроль исполнения принятых решений;
- согласование с Координатором проекта изменений рамок проекта, требование при необходимости дополнительного финансирования (согласование изменения содержания, сроков и стоимости проекта);
- принятие результатов этапов проекта и проекта в целом;
- контроль выполнения в проекте принятых постановлений;
- назначение ответственных исполнителей со стороны компании-исполнителя;
- проведение в жизнь административных решений Координационного совета, касающихся компании-исполнителя.

### Пример

Необходимо еще в момент старта проекта решить, в каких случаях группа внедрения сама может принимать решение (например, по поводу конфликта требований разных пользователей), а в каких необходима

эскалация вопроса руководителем проекта для вынесения на заседание Координационного совета (к примеру, при определении прав доступа различных подразделений или внешних/дочерних предприятий).

---

*Руководитель проекта* участвует во всех этапах проекта по внедрению, осуществляет руководство работами в рамках утвержденных бюджетов, определяет операционные аспекты взаимодействия специалистов компаний, занятых в проекте. Как правило, его занятость в проекте составляет от 80 до 100 % (неполная занятость и совмещение нескольких ролей и (или) нескольких проектов одновременно существенно повышает риск неудачи проекта).

Среди его основных задач в проекте:

- управление проектом в рамках утвержденных планов и бюджетов;
- составление и исполнение бюджета проекта;
- участие на уровне Координационного совета в принятии стратегических решений по ходу выполнения проекта;
- представление Координационному совету результатов работы группы внедрения;
- мониторинг хода проекта, ведение отчетности и, в случае необходимости, вынесение на рассмотрение Куратора проекта (Координационный совет) решения об изменении объема, сроков и стоимости работ;
- разработка сценариев и планов проекта, процедур принятия решений;
- поддержание мотивации команды;
- контроль качества ведения проекта;
- обеспечение соответствия всех аспектов реализации проекта условиям заключенного с заказчиком договора;
- идентификация рисков и проблем процесса внедрения.

Руководитель проекта от исполнителя является руководителем группы внедрения, и на время выполнения работ по проекту все консультанты-программисты заказчика, входящие в группу внедрения, также работают под его функциональным руководством. Любые вопросы планирования работ, отчетности и представления к премированию решаются в первую очередь им. Однако данные полномочия могут быть делегированы им менеджеру проекта со стороны заказчика.

Перейдем к рассмотрению ролей проектной команды со стороны исполнителя (напомним, один человек может выполнять несколько функций: и консультанта по функциональности, и консультанта по постановке задач).

Основной задачей консультантов со стороны исполнителя является внедрение системы и обеспечение ее сопровождаемости (для участия в данном процессе специалистов со стороны заказчика).

Обладая знаниями в области функциональности системы и ее отдельных модулей, а также навыками проведения обучения пользователей, консультанты подрядчика участвуют во всех видах проектных активностей:

- собирают требования и проектируют систему;
- устанавливают и настраивают программное решение на тестовом сервере;
- консультируют и обучают сотрудников заказчика;
- тестируют и совершенствуют систему совместно с заказчиком при развертывании и проведении опытно-промышленной эксплуатации.

Бизнес-аналитик выполняет работы по проведению обследования предприятия, формирует в случае необходимости рекомендации по оптимизации и реинжинирингу бизнес-процессов заказчика и участвует в их исполнении, формирует рекомендации по информационному обеспечению бизнес-процессов и взаимодействию, а также принимает участие в формировании требований на создание системы (совместно с консультантом по постановке задач, если подобная роль отдельно выделяется).

Консультант по функциональности выполняет работы по настройке и доработке функциональных возможностей системы с учетом специфики бизнес-процессов предприятия-заказчика (на основе данных, предоставленных бизнес-аналитиком), а также принимает участие в процессе внедрения системы.

Технический консультант осуществляет техническую и технологическую поддержку проекта совместно с программистом-консультантом, ответственным за ПО.

Преподаватель курсов по функциональности осуществляет обучение специалистов заказчика, в дальнейшем играющих роль пользователей, описывая им интерфейс системы, ее ключевые возможности и настройки. Другой группой обучаемых сотрудников являются ИТ-специалисты заказчика, после внедрения осуществляющие текущую техническую и консультационную поддержку пользователей.

*Со стороны заказчика.* Среди руководящих ролей на стороне заказчика выделяется роль Координатора проекта. Координатор проекта, будучи, как и куратор от исполнителя, представителем топ-менеджмента компании (в ранге не ниже заместителя генерального директора предприятия-заказчика), участвует в принятии оперативных административных решений и контроле их выполнения.

Иногда эта роль называется «Куратор проекта от заказчика».

Координатор проекта:

- руководит работой Координационного совета;
- созывает заседания Координационного совета и определяет их повестку;

- утверждает изменение рамок проекта, обеспечивая при необходимости дополнительное финансирование;
- принимает результаты этапов проекта и проекта в целом;
- назначает ответственных исполнителей со стороны заказчика;
- отвечает за согласование действий различных подразделений и специалистов участвующих в проекте сторон;
- проводит в жизнь административные решения Координационного совета, касающиеся подразделений заказчика, принимает решение об их премировании.

Если система внедряется на предприятии, имеющем несколько филиалов, либо на нескольких предприятиях (одновременно или последовательно), рекомендуется для каждого предприятия или филиала создавать свою группу внедрения со своим собственным руководителем. При этом желательно, чтобы Координационный совет был общим. Именно координатор проекта будет в таком случае осуществлять руководство всем проектом внедрения.

*Руководитель проекта со стороны заказчика* выполняет те же функции, что и руководитель проекта от исполнителя, и фактически является его заместителем (во время отсутствия последнего на предприятии).

Оба руководителя проекта, как от заказчика, так и от исполнителя, имеют достаточно четко определенные зоны ответственности. Однако на протяжении всего процесса, вплоть до проведения тестовой эксплуатации, наиболее критична роль руководителя проекта от исполнителя, поскольку он занимается организацией основных активностей. С момента тестовой эксплуатации главная роль должна переходить к руководителю проекта от заказчика как лицу более заинтересованному в положительных результатах проекта. Тем не менее на практике координирует работы и управляет процессом создания и внедрения системы тот руководитель проекта, который более деятелен, активен и вовлечен в проект.

*Руководитель проекта со стороны заказчика:*

- распоряжается выделенными на проект фондами (в рамках бюджета проекта!);
- выносит возникшие проблемы на Координационный совет и контролирует выполнение принятых соответствующих постановлений.

Кроме того, руководитель проекта со стороны заказчика несет ответственность за:

- обеспечение обязательного присутствия требуемых специалистов предприятия-заказчика на рабочих местах во время реализации проекта;
- создание сплоченной, работоспособной группы внедрения и укрепление ее уверенности в конечном успехе проекта (привле-

чение всех членов группы к обсуждению текущих вопросов, периодическая замена людей, которые возглавляют обсуждение);

- устранение препятствий на пути к повышению эффективности функционирования группы (выделение помещения под обучение и обсуждение вопросов по проекту или привлечение дополнительных ресурсов для облегчения работы членов группы внедрения);
- своевременное представление проблем группы внедрения перед вышестоящим руководством и руководителями других подразделений;
- оказание помощи в идентификации требований и проблем процесса внедрения;
- обеспечение наличия и работоспособности компьютеров, принтеров, сети и другого необходимого на проекте оборудования;
- установку и настройку требуемого общесистемного и специального программного обеспечения на рабочих местах специалистов группы внедрения и пользователей заказчика.

В качестве руководителя проекта от заказчика должен выступать всесторонне развитый человек. Он должен обладать достаточными знаниями о методах ведения бизнеса на предприятии, способностью организовывать, обучать людей и преподносить им целостное видение функционирования предприятия.

В идеальной ситуации руководитель проекта от заказчика как функциональный руководитель всех работ должен 100 % своего рабочего времени посвящать проекту. Совмещение этой деятельности с выполнением других обязанностей в значительной степени повышает риск неудачи проекта.

Руководитель проекта со стороны заказчика должен обеспечить выделение команде исполнителя необходимого оборудования и инфраструктурных условий (включая установку требуемого общесистемного и специального ПО на рабочих местах группы внедрения). Как правило, ему в этом помогает *Администратор проекта*. Он осуществляет корректное своевременное документирование всех этапов проекта (включая написание протоколов встреч), согласование организационно-финансовых и договорных вопросов, а также организационное сопровождение проекта (заказ переговорных комнат, необходимого оборудования).

*Группа внедрения* со стороны заказчика включает как ИТ-специалистов, так и сотрудников, задействованных во внедрении и работе с системой бизнес-подразделений. ИТ-специалисты заказчика являются сотрудниками отдела автоматизации или ИТ-департамента (чаще всего программистами) и принимают активное участие во всех работах по внедрению системы (особенно на этапах создания проектного решения, доводки модулей системы в ходе тестирования и опытно-промышленной эксплуатации, формирования системной документации).

При планировании проекта необходимо оценить усилия, требующиеся для того, чтобы все члены группы внедрения имели достаточно времени на работу в рамках проекта, а также определить сферы ответственности. Это может потребовать выделения дополнительных полномочий, ресурсов, временной перемены сферы ответственности.

Функции ИТ-специалистов (консультантов-программистов) заказчика:

- содействие консультантам исполнителя по вопросам внедрения и сопровождения системы;
- сотрудничество с командой исполнителя в части предоставления информации о системах заказчика и специфике предприятия (сбора недостающей информации);
- обучение специфике работы с системой и ее настройке с дальнейшей передачей необходимых знаний ключевым сотрудникам предприятия и консультированием пользователей;
- формирование инструкций пользователей и администратора системы (баз данных) с последующим их обновлением по мере необходимости;
- обеспечение наполнения системы необходимыми выверенными данными (заполнение справочников, подготовка и ввод первичной информации, начальных остатков и т. п.) через личное участие (обучение) и через осуществление контроля над наполнением системы силами других ответственных специалистов заказчика.

Консультантам-программистам заказчика крайне важно перенимать навыки и знания о системе у консультантов исполнителя, так как после окончания контрактов на внедрение и сопровождение именно они будут заниматься поддержкой и развитием данной системы.

**Требования к компетенции проектных команд ICB IPMA.** Основные компетенции и рекомендации по формированию проектных команд описываются в руководстве, опубликованном некоммерческой профессиональной ассоциацией по управлению проектами IPMA. Созданная еще в 1965 г., сейчас IPMA объединяет 50 национальных ассоциаций по управлению проектами. В России ее представляет национальная ассоциация СОБНЕТ. СОБНЕТ разработала соответствующий стандарт для сертификации российских специалистов — «Основы профессиональных знаний и Национальные требования к компетентности специалистов по управлению проектами» (НТК).

Ключевым стандартом IPMA по управлению проектами является ICB, *IPMA Competence Baseline*, руководство, описывающее требования к компетенциям, необходимым менеджерам проектов и членам проектных команд для управления проектами, программами или

портфелем проектов. Система оценки компетенций состоит из четырех уровней сертификации IPMA:

- 1) уровень А — сертифицированный директор проектов;
- 2) уровень В — сертифицированный старший менеджер проектов;
- 3) уровень С — сертифицированный менеджер проектов;
- 4) уровень D — сертифицированный специалист по управлению проектами.

Для каждого из уровней определяются:

- входные требования;
- основная компетенция;
- дополнительные требования.

### Пример

Уровень В — Сертифицированный старший менеджер проектов.

• Входные требования:

— минимум пять лет опыта управления проектами, из которых три года — в статусе руководителя сложных комплексных проектов.

• Основная компетенция:

— способность руководить комплексными проектами.

• Дополнительные требования:

— ответственность за все элементы управления комплексным проектом;

— использование релевантных процессов, методов, техник и инструментов управления проектами;

— статус руководителя крупной проектной команды.

Основной отличительной особенностью ICB является выделение 46 различных систематизированных «элементов компетентности» со взаимодействиями между ними (табл. 7.9). Для каждого из элементов определяются: название, описание содержания, основные шаги и критерии оценки опыта для сертификации.

Таблица 7.9

#### Элементы компетентности ICB IPMA

Технические компетенции	Поведенческие компетенции	Контекстуальные компетенции
Успешность управления проектом. Заинтересованные стороны. Требования и цели проекта. Риски и возможности. Качество. Организационная структура проекта.	Лидерство. Вовлечение и мотивация. Самоконтроль. Уверенность в себе. Разрядка. Открытость. Творчество. Ориентация на результат.	Ориентация на проект. Ориентация на программу. Ориентация на портфель проектов. Осуществление проектов, программ и портфелей проектов. Родительская компания.

Технические компетенции	Поведенческие компетенции	Контекстуальные компетенции
Работа команды. Разрешение проблем. Структура проекта.	Продуктивность. Согласование. Переговоры. Конфликты и кризисы.	Бизнес и предпринимательская деятельность. Системы, продукты и технологии.
Замысел и итоговый продукт проекта. Время и фазы жизненного цикла проекта. Ресурсы. Стоимость и финансы. Закупки и контракты. Изменения. Контроль и отчетность. Информация и документация. Коммуникации. Инициация проекта. Закрытие проекта	Надежность. Понимание ценностей. Этика	Управление персоналом. Здоровье, безопасность, охрана труда и окружающей среды. Финансы. Юридические аспекты проекта

### 7.1.8. Управление портфелем проектов

Когда в организации реализуется несколько взаимосвязанных проектов, либо объединенных временными и ресурсными ограничениями, либо имеющих перед собой одну цель, следует говорить о *программе* или *портфеле проектов*. Эти понятия различаются по своей сути, но часто используются в качестве синонимов, и поэтому важно в самом начале прояснить существующие различия.

Выделим все различия в табличной форме (на основании Стандарта по управлению программами, разработанному в 2006 г. составителем PMBoK, институтом PMI, и Руководства ICB IPMA, табл. 7.10).

Таблица 7.10

#### Проект, программа и портфель проектов

Характеристика	Проект	Программа	Портфель проектов
Основная цель	Проекты уникальны по сути и имеют четкие результаты	При реализации программы изменяются многие условия и элементы для соответствия ожидаемым выгодам	Портфели проектов концентрируются на бизнес-активностях, меняющихся с изменением стратегических целей компании
Видение и стратегия	Определяются в ТЭО проекта	Реализуются в рамках программы	Проводится мониторинг портфеля

Продолжение табл. 7.10

Характеристика	Проект	Программа	Портфель проектов
			на соответствие стратегии
Бизнес-выгоды	Не рассматриваются в рамках проекта	Являются основным элементом программы	Часто не рассматриваются
Организационные изменения	Часто не рассматриваются	Рассматриваются	Не рассматриваются
Ограничения по времени и стоимости	Определяются в начале проекта, и постоянно осуществляется их мониторинг	Относятся к конкретным проектам внутри программы	Базируются на приоритетах и стратегических целях конкретного портфеля проектов
Управление изменениями	Менеджер проектов старается минимизировать необходимость изменений	Менеджеры программ принимают происходящие изменения	Менеджеры портфеля проектов проводят регулярный мониторинг изменений
Измерение успеха	Успех измеряется соответствием плану по бюджету, срокам, содержанию и качеству работ	Успех измеряется в достижении выгод, реализации стратегических преимуществ, положительном ROI	Успех измеряется в терминах интегрированного успеха компонентов портфеля проектов
Стиль лидерства	Стиль лидерства основывается на выполнении задач для достижения целевых показателей	Стиль лидерства основывается на управлении взаимоотношениями сторон и разрешении конфликтов при соблюдении интересов стейкхолдеров	Стиль лидерства основывается на добавочной ценности, привносимой в процесс принятия решений
Принцип руководства	Менеджеры проекта руководят специалистами в команде	Менеджеры программ руководят менеджерами проектов	Менеджеры портфеля проектов координируют своих подчиненных (в офисе управления проектами)
Принципы планирования	Детальное планирование для обеспечения выполнения работ	Высокоуровневое планирование всех проектов	Акцент не на планировании, а на создании и поддержке

Характеристика	Проект	Программа	Портфель проектов
			необходимых процессов и коммуникаций в портфеле проектов
Принципы мониторинга	Менеджеры проектов проводят мониторинг задач и работ для выполнения проекта	Менеджеры программ организуют мониторинг проектов и текущую работу через структуры корпоративного управления	Менеджеры портфеля проектов контролируют интегрированные показатели эффективности проектов

Далее мы будем рассматривать пример управления программой проектов как более сложной структуры, фокусирующейся не только на формальных ограничениях и методах балансировки проектов, но и на стратегическом аспекте их выбора. **Программой проектов** в данном контексте и терминологии будет являться объединенный общей целью и условиями выполнения набор проектов, реализация которого призвана обеспечить соответствие ожидаемым выгодам. В таком случае важен именно совокупный эффект от выполнения всей программы, а не отдельные результаты каждого из проектов (рис. 7.15).

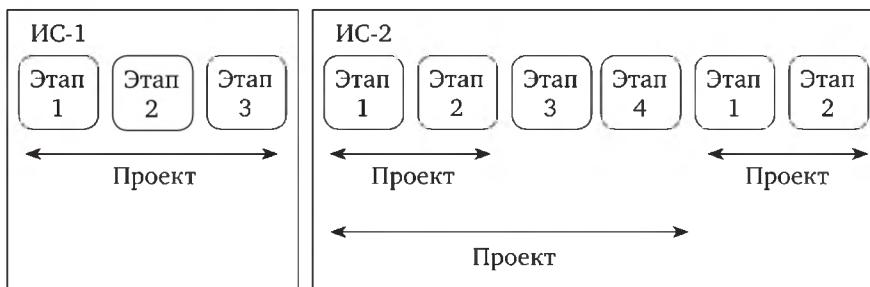


Рис. 7.15. Проекты как составные части программы проектов

### Пример

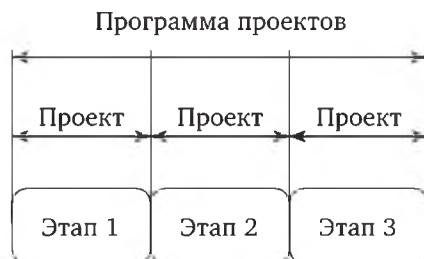
Рассмотрим программу проектов, цель которой — «Организация обеспечения информационной поддержкой управления сетью распределенных мультифункциональных придорожных комплексов». При этом среди предоставляемых комплексом услуг: терминалы оплаты, вендинговые автоматы, кафе с wifi и, разумеется, сама заправка (газом, нефтепродуктами, стеклоомывающей жидкостью и т. д.). Соответственно, среди систем автоматизации будут, как минимум:

- система управления автозаправочным комплексом;
- система управления гостиничным комплексом;
- система управления розничной торговлей в региональной сети АЗК;
- мультисервисная корпоративная сеть;
- ИТ-инфраструктура в центральном офисе;
- системы безопасности и охраны:
  - система видеонаблюдения внутренней территории комплекса,
  - автоматическая паркинг-система,
  - система регистрации автомобильных номеров (въезды и выезды на территории комплекса),
  - система видеонаблюдения касс,
  - другие системы.

Автоматизация такого набора бизнес-процессов и построение комплекса (в значительной части с инфраструктурной точки зрения) не могут происходить одновременно, и важно корректно расставить приоритеты и определить те системы, создание которых является необходимым условием для реализации следующих.

Так, в построении систем безопасности и охраны с большой долей вероятности будут задействованы другие специалисты, нежели в автоматизации процессов розничной торговли. Кроме того, масштаб и длительность второго проекта превысят аналогичные показатели первого. Это необходимо также учитывать при планировании персонала на проекты и заключении контрактов на их реализацию.

Для успешного прохождения всего жизненного цикла ИС крайне важен учет тесной взаимосвязи его этапов как отдельных проектов. К тому же неизбежно и их взаимодействие с прочими реализующимися компанией проектами, возможно, в рамках борьбы за совместно используемые ресурсы. Соответственно, удобнее и целесообразнее рассматривать ЖЦИС до этапа эксплуатации как *единую программу проектов*, планирование которой, как правило, осуществляется «сверху вниз» в силу необходимости распределения временных, денежных и прочих ресурсов. Критическим фактором успеха является логическая связь этапов/проектов (определенная выбранный моделью жизненного цикла). На рис. 7.16 приведен пример для каскадной модели.



*Рис. 7.16. Пример для каскадной модели*

## Пример

---

Если, как уже было сказано, жизненный цикл информационной системы может рассматриваться как целая программа проектов, разберем отдельно проект по модернизации на примере системы бухгалтерского учета, проведенный российской компанией — системным интегратором.

Допустим, компания (головной офис которой находится в одной из европейских стран) во всех российских подразделениях использовала Microsoft Dynamics Nav версии 3.01. При анализе и сборе требований, предшествовавших модернизации, было определено, что с ростом масштабов бизнеса, числа операций, количества пользователей возникла необходимость расширения функциональных возможностей системы и поддержки вдвое увеличившегося числа одновременно работающих в системе сотрудников.

В результате было принято решение модернизировать ИС для обеспечения полноценного учета финансовой деятельности (по двум стандартам: РСБУ и US GAAP), получения информации из систем других филиалов компании (для чего понадобится разработка интеграционного решения и интерфейсов), увеличения скорости обработки данных.

При подготовке технического задания на модернизацию был проведен анализ всех финансовых операций, сформирована схема работы сотрудников в системе (с проверкой вводимых данных, оптимизацией многих процедур), а также выбрана программная платформа для модернизации (Microsoft Dynamics Nav, только уже версии 4.0).

По итогам разработки, настройки и тестирования получена модернизованная информационная система с расширенной функциональностью (например, выгрузка данных по контрактам во вновь интегрированную систему оперативного учета), увеличенным быстродействием и более эффективным использованием аппаратных ресурсов (за счет обновленной версии ПО).

А значит, действительно, этап модернизации (как и остальные стадии, за исключением эксплуатации) имеет признаки проектной деятельности, и каждым из этапов можно управлять через определение конечного результата проекта, а также временных и стоимостных ограничений.

---

### 7.1.9. Офис управления проектами

Офис управления проектами (*Project Management Office, PMO*) представляет собой подразделение или отдельную организацию, которая осуществляет централизацию и координацию управления проектами, входящими в его сферу деятельности. При этом сфера ответственности может быть очень вариативной, начиная с поддержки в проектном управлении и заканчивая непосредственным управлением.

Среди функций офиса управления проектами выделяют:

- управление общими ресурсами, относящимися к проектам PMO;

- определение и разработку методологии, стандартов, лучших практик управления проектами;
- обучение, надзор, коучинг, наставничество;
- мониторинг соответствия стандартам, процедурам, шаблонам управления проектами;
- проведение аудита;
- управление общей документацией проекта, включая принципы, процедуры, шаблоны и др.;
- обеспечение коммуникации между проектами и между членами проектных команд.

На первый взгляд может показаться, что РМО полностью идентичен роли менеджера проекта с той лишь разницей, что РМО — это еще и организационная структура. Это не так. Роль проектного менеджера принципиально отличается от деятельности РМО. Так, если проектный менеджер должен выполнить конкретные цели проекта, то РМО обычно рассматривает возможности достижения более хорошего результата. Разница присутствует и в использовании ресурсов. Менеджер проекта ограничен проектными ресурсами, а РМО балансирует общие ресурсы организации между всеми проектами. И, наконец, менеджер проекта в ходе своей работы может управлять только проектными ограничениями, а РМО — стандартами, общими рисками и взаимными зависимостями проектов на уровне целой организации.

Как правило, создание РМО происходит по ряду объективных причин. Выделяют следующие причины для создания офиса управления проектами:

- проектов настолько много (или они настолько крупные), что ими становится тяжело управлять;
- прозрачность управления проектами минимальна;
- в разных проектах используется разная отчетность;
- вся информация о состоянии проектов находится исключительно в головах проектных менеджеров;
- уход менеджера из проекта слишком критичен и чреват последствиями вплоть до закрытия проекта.

Если организация столкнулась с одной или несколькими из перечисленных проблем (а зачастую они встречаются группой), то создание РМО — логичный выход из сложившейся ситуации.

РМО — это альтернатива классическому управлению проектами, которому присущи изложенные выше проблемы. Офис управления проектами предполагает формирование корпоративной культуры управления проектами, которая ориентируется на накопление знаний и их последующее использование. В результате организация сохраняет полученные уроки, а также создает важный в целом реестр рисков. В какой-то степени РМО позволяет сделать полезными даже

ошибки, потому что извлеченный из них опыт станет достоянием организации.

Структура, функции и способы работы РМО варьируются в зависимости от типа организации. В качестве двух «полюсов» принято выделять проектные и непроектные организации. Различия между офисами управления проектами для каждой из них приведены в табл. 7.11.

Таблица 7.11

**РМО в непроектной и проектной организации**

Непроектная организация	Проектная организация
РМО существует в виде группы проектных менеджеров, подчиненных одному топ-менеджеру или напрямую генеральному директору	Проектный офис отвечает за производственные проекты и за проекты развития
Менеджеры проекта занимаются планированием и контролем работы каждого участника проекта	Зоны ответственности за проекты принадлежат разным подразделениям (не подчиненным друг другу)
Методология управления проектами обычно «отстает» от практики	Часто создание центрального РМО сталкивается с конфликтом полномочий
Основная нагрузка ложится на менеджеров проектов	Разрешить конфликт полномочий удается только через понимание того, что проектная организация сама по себе является проектным офисом
Редкое применение специализированного ПО	

Теперь следует рассмотреть различные виды РМО. Одна из самых распространенных классификаций была предложена компанией Gartner. Выделяют следующие типы РМО:

- *репозитарный (Repository Model)* — проектный офис, главная функция которого — предоставлять стандарты управления проектами, а также стандартный инструментарий и базу для проектного управления;
- *обучающий (Coach Model)* — офис управления проектами обеспечивает работу проектного менеджмента за счет обучения и координации;
- *управляющий (Management Model)* — офис управления проектами осуществляет централизованное управление.

Ролевая структура РМО достаточно проста. В качестве ключевых сотрудников рассматриваются:

- руководитель РМО;
- менеджеры проектов;
- администраторы проектного офиса (или проекта);

- бизнес-аналитики;
- специалисты технической поддержки;
- функциональные специалисты (1—2 на участок).

**Внедрение РМО.** Для быстрого и эффективного внедрения РМО на предприятии требуется соблюдение нескольких условий. Во-первых, необходимо наладить устойчивую и эффективную коммуникацию с Заказчиком РМО, которым часто является генеральный директор. Далее требуется определить серьезную проблему, которую нужно решить как можно скорее ввиду возможности наступления негативных последствий. После этого необходимо предложить быстрореализуемое решение, которое создает ценный для бизнеса актив, базирующийся на ключевых ресурсах РМО (знаниях, компетенциях, процедурах, технологиях и т. п.). Если актив удастся защитить путем его постоянного совершенствования, то фундамент РМО заложен.

Важно понимать, что РМО может быть утвержден только «наверху», поэтому перед началом необходимо подготовить какое-то основание. Для этого можно заниматься «пропагандой» РМО как среди рядовых сотрудников, так и среди управленцев различного звена. А когда первые камни заложены в основание РМО, нужно как можно чаще демонстрировать реальную пользу нового актива. Параллельно с этим требуется повышать профессионализм проектных менеджеров, поскольку проекты по-прежнему сильно зависят именно от них.

В самом общем виде выделяют три этапа развития офиса управления проектами:

- этап управления проектами (фокус на обучение и тренинг специалистов);
- этап управления программами (фокус на высокоуровневых программах управления);
- этап управления портфелем (фокус на управлении знаниями и реализацией проектов).

В целом же существует три общепринятых метода внедрения РМО. Каждый из них будет подробно рассмотрен далее.

**Метод пилотного проекта.** Метод применяется, когда имеется проект с не очень большим сроком выполнения. При этом проект должен быть важен для компании, поскольку только в этом случае топ-менеджмент будет рассматривать его в качестве представительного. Характеристики пилотного проекта должны пересекаться с другими проектами.

Важно понимать, что в качестве пилотного ни в коем случае нельзя выбирать проект, который заведомо имеет неразрешимые проблемы. Большинство неразрешимых проблем не будет решено при помощи РМО, однако идея создания офиса управления проектами будет дискредитирована в компании на какое-то время.

В целом метод пилотного проекта рекомендуется применять, когда топ-менеджмент компании заинтересован в РМО, но не осознает его эффективности или сомневается в ней.

*Метод четырех ступеней.* В данном случае нужно последовательно выполнить четыре шага — пройти четыре ступени на пути к полноценному РМО.

1. *Закладка фундамента.* Здесь устанавливают границы РМО, определяют цели и задачи. Производится оценка текущих возможностей РМО.

2. *Запуск краткосрочных мероприятий.* Для демонстрации полезности РМО производится запуск краткосрочных мероприятий и поддержка имеющихся проектов.

3. *Разворачивание долгосрочных мероприятий.* Здесь нужно наладить обучение персонала, оптимизировать процессы управления проектами, создать стабильную систему поддержки.

4. *Поддержка и развитие.* Достигнув этой ступени, РМО полноценно функционирует и поддерживает деятельность по проектам.

На первых ступенях часто можно столкнуться с недостатком формальных полномочий у руководителя РМО. Текущие проектные команды не будут соблюдать новые правила, а функциональные и проектные руководители могут быть настроены очень скептически. При этом полезность РМО очень трудно продемонстрировать, что осложняется проблемами получения информации от руководителей проектов.

*Долгосрочный метод.* Этот метод включает десять шагов. На первом шаге производится *Инвентаризация*. Собирается информация обо всех проектах компании. После этого создают реестр проектов, который очень часто оказывается важным элементом в наведении порядка в управлении проектами.

На втором шаге производится *Календарное планирование*. Как правило, календарный план не очень подробный. Он не должен отнимать много времени на поддержание в актуальном состоянии. С другой стороны, план не должен быть излишне общим — он должен помочь понять реальное положение дел и принять обоснованные решения. Однако главная цель второго шага другая. Требуется разработать правила календарного планирования, опираясь на опыт проектного управления компании. В дальнейшем технология планирования распространяется на все проекты.

Третий шаг — это *Отслеживание*. Создать технологию планирования сложно, но еще сложнее сделать так, чтобы ее стали применять. Любой план становится бесполезным, поскольку после его написания могут произойти изменения, которые сделают его неактуальным. Соответственно, нужны правила внесения изменений. На этом шаге разрабатывается схема актуализации плана. Формируются правила перепланирования.

Название следующего шага — *Взаимодействие*. Закончив с планированием проекта, необходимо переходить к его реализации. Если правила работы и взаимодействия не изменить, то новая технология планирования будет отторгнута компанией по естественным причинам. Таким образом, на данном этапе важно встроить РМО в структуру компании. Для этого определяют права, обязанности и полномочия членов проекта и функциональных подразделений, а также формализуют правила взаимодействия между ними.

На шаге *Коммуникаций* производится формализация информационных потоков путем создания схемы документооборота и шаблонов для основных документов. Также производится хранение созданных документов и извлечение из них информации и (или) знаний.

Следующий шаг называется *Регламенты, процедуры и шаблоны*. Как явствует из названия, здесь требуется создать формальную основу управления проектами, которая минимизирует «творческие» издержки. Действительно, многие проектные задачи практически однотипны, однако их решение требует от участников применения творческих подходов. Этого требуется избежать для оптимизации проектных работ. Важно отметить, что именно РМО отвечает за корректное закрытие проектов.

После этого можно переходить к *Обучению и созданию корпоративной системы управления проектами (КСУП)*. Процедуры, инструкции, методики и методологии создаются и распространяются офисом управления проектами, и для этого требуется ИС или хотя бы регламенты распространения. Вместе с этим производится обучение менеджеров проектов, экспертов по функциональным областям и других участников РМО.

Далее происходит создание *Базы знаний и ИСУП*. С этой целью внедряются индивидуальные программные средства проектного управления, будь то MS Project, Primavera, сайт РМО/проекта и т. п. Эффективное использование программных инструментов также обеспечивается офисом проектного управления.

*Администрирование и управление ресурсами* предполагает организацию труда проектных команд и отдельных элементов структуры компании в целом. Оборудуются рабочие места, разрабатываются алгоритмы карьерного продвижения, формируются методы «зabora» сотрудников из отдела на проект и их возвращения.

Когда управление отдельными проектами становится зрелым и эффективным, начинается этап *Управления портфелем проектов*. Разнообразные проекты группируются, систематизируются. Программы и портфели делятся на управляемые части, которые связаны по срокам и результатам.

## 7.2. Корпоративные методологии

### 7.2.1. Методологии компании Microsoft

**Microsoft Solution Framework.** Компания Microsoft подготовила и применяет несколько методик для покрытия не только ЖЦС, но и технологической инфраструктуры, их поддерживающей.

В контексте рассмотрения ЖЦПО нас интересует именно методология разработки: как являющийся одним из основных аспектов управления и взаимодействия участников процесса, так и другие области знаний (управление рисками, планирование). В целом охватываемые MSF дисциплины описаны в пяти частях (так называемых белых книгах), однако интересно, что командами консультантов Microsoft применяется на практике не этот ресурс, а методика MSF for Agile Software Development, которая является прикладным вариантом MSF и отражает общий методологический подход к итеративной разработке.

Если обратиться непосредственно к процессу разработки и внедрения, то его характеризуют:

- итеративность;
- формирование в качестве результата ИТ-решения.

---

**ИТ-решение** — сконцентрированная поставка набора элементов (таких как программно-технические средства, документация, обучение, сопровождение и внешние коммуникации), необходимых для удовлетворения некоторой бизнес потребности конкретного заказчика.

---

Именно ИТ-решения (а не просто программные продукты, поставляемые в виде дистрибутивов) являются основным продуктом и результатом процесса разработки и внедрения по MSF (несмотря на то что аналогичная философия лежит в основе многих методологий, все равно термин «программный продукт» является очень распространенным).

Основной состав MSF — это две модели и три дисциплины, которые подробно рассматриваются в пяти белых книгах. В состав MSF входят:

- модели:
  - модель группы,
  - модель процессов;
- дисциплины:
  - дисциплина «Управление проектами»,
  - дисциплина «Управление рисками»,
  - дисциплина «Управление готовностью».

**Модель процессов.** Модель процессов — это «основа основ» методологии MSF. Модель процессов MSF основана на сочетании во-

допадной и спиральной моделей жизненного цикла ИС. Таким образом, в методологии MSF проект реализуется поэтапно, все этапы могут повторяться «по спирали», а между этапами существуют заранее определенные контрольные точки (рис. 7.17).



Рис. 7.17. Модель процессов MSF<sup>1</sup>

Модель совмещает предсказуемость и широкие возможности планирования из водопадной модели с вариативным подходом к решению задач, который присущ спиральным моделям. В результате удается добиться высокой степени прогнозируемости, учитывать текущие условия и окружение, а также проводить работу на уровне всего предприятия.

**Создание общей картины ИТ-решения.** Первый этап модели MSF — это Создание общей картины ИТ-решения. Задачи этапа таковы:

- определить проектную команду;
- определить структуру проекта;
- определить бизнес-цели проекта;
- оценить текущую ситуацию;
- сформировать документ с описанием общей картины и областью действия проекта;
- определить требования пользователей;

<sup>1</sup> Introduction to the Microsoft Solution Framework // TechNet : сайт. URL: <https://technet.microsoft.com/en-us/library/bb497060.aspx> (дата обращения: 15.06.2020).

- разработать концепции для ИТ-решения;
- оценить риски;
- закрыть этап.

Этап содержит в себе две контрольные точки: «Костяк команды сформирован» и «Общая картина ИТ-решения создана».

Первая контрольная точка не сводится к получению полного пофамильного списка участников проекта. Для достижения этой контрольной точки требуется определить роли и обязанности членов команды, а также определить иерархии ответственности и ответности. Также данная контрольная точка предполагает, что определена общая структура команды и наложены каналы взаимодействия с заказчиком.

Контрольная точка «Общая картина ИТ-решения создана» предполагает разработку концепции ИТ-решения, которым команда будет руководствоваться в дальнейшем для достижения бизнес-целей, декларированных в проекте. Контрольная точка характеризуется наличием описания того, что входит в проект, а что не входит. При этом документ не является итоговым: к данной точке создается предварительная рецензируемая версия.

Завершение этапа происходит, когда достигнута третья контрольная точка — «Документ общей картины и области действия проекта утвержден».

**Планирование.** На этапе Планирования от команды требуется понять, каким будет продукт и его реализация. На этом этапе происходит формирование функциональной спецификации, производится детализация плана работ, осуществляется оценка бюджета и сроков, требующихся для реализации проекта.

Анализ требований также выполняется на этапе Планирования. В методологии MSF принято разделять требования на четыре типа: бизнес-требования, пользовательские требования, функциональные требования и системные требования. Вслед за этим команда приступает к разработке проекта решения и планированию профилей пользователей. На основе выделенных профилей разрабатываются сценарии применения, которые будут выполняться пользователями одного типа. Кроме того, рассматриваются альтернативные варианты использования системы.

Задачи Планирования могут быть сформулированы следующим образом:

- разработать архитектуру ИТ-решения;
- сформировать функциональную спецификацию;
- разработать проектные планы;
- сформировать календарный график работ;
- создать среду разработки, тестирования и тестовой эксплуатации;
- закрыть этап.

Планирование — достаточно сложный и обширный этап, который насчитывает пять контрольных точек:

- функциональная спецификация создана;
- план управления рисками создан;
- среда разработки и тестирования определена;
- план и календарный график проекта созданы;
- проектные планы утверждены.

Все результаты данного этапа в дальнейшем используются для принятия компромиссных решений.

*Разработка.* Этап Разработки подразумевает непосредственное создание ИТ-решения, т. е. написание и документацию программного кода. Убедившись, что задачи предыдущих этапов выполнены, проектная команда приступает к реализации задач, характерных для этапа разработки:

- создать прототип ИТ-решения;
- разработать программные компоненты ИТ-решения;
- создать готовое ИТ-решение (рассматривается как последовательность промежуточных выпусков);
- закрыть разработку (реализовать все функции в ИТ-решении, поставить код и документацию заказчику).

В методологии MSF выделяют следующие результаты разработки:

- исходный программный код и исполняемые файлы;
- сценарии установки и конфигурации для развертывания;
- итоговая функциональная спецификация;
- элементы поддержки ИТ-решения;
- спецификации и сценарии тестирования.

Этап разработки также содержит и контрольные точки, основная из которых — «Итоговое утверждение области действия проекта». Эта контрольная точка считается достигнутой, когда все функции продукта реализованы и прошли предварительное тестирование. Считается, что после этого ИТ-решение пригодно к внешнему тестированию и началу стабилизации.

*Стабилизация.* Этап Стабилизации нужен, чтобы довести продукт до требуемого уровня качества. Стабилизация предполагает проведение всеобъемлющего тестирования с целью обнаружения и устранения дефектов. Кроме того, в рамках этапа Стабилизации проверяются сценарии развертывания и осуществляется пробная эксплуатация ИТ-решения.

Тестирование включает следующие виды активности:

- тестирование компонентов;
- тестирование БД;
- тестирование ИТ-инфраструктуры;
- тестирование безопасности;
- тестирование интеграции;
- тестирование эргономичности;

- нагрузочное тестирование;
- регрессивное тестирование;
- ведение отчетности по тестированию.

Следующая задача — пробная эксплуатация. Для этого ИТ-решение развертывается на рабочих местах для эксплуатации группой тестировщиков со стороны заказчика, которые в дальнейшем будут непосредственно работать с ИТ-решением. Важно проверить способность ИТ-решения функционировать в реальных рабочих сценариях.

Важнейшая контрольная точка — это «Появление версии ИТ-решения, в которой не найдено критических ошибок». После этого выпускается несколько финальных версий, одна из которых, признанная наиболее удачной, развертывается для пробной эксплуатации. Финальная контрольная точка для этого этапа — «Подтверждение готовности продукта к развертыванию в промышленной среде».

**Развертывание.** Этап Развёртывания включает установку ИТ-решения, промышленную стабилизацию и окончательную передачу заказчику и группе сопровождения. Основные контрольные точки этапа:

- основные компоненты развернуты;
- решение развернуто;
- решение стабилизировано;
- решение передано в эксплуатацию заказчику.

Важно понимать, что очень трудно определить формальное завершение этапа Развёртывания. Причина кроется в том, что неполадки могут выявляться и в ходе промышленной эксплуатации. Таким образом, требуется четкое определение условий достижения финальной контрольной точки в каждом конкретном проекте.

**Модель группы.** Управление проектной командой — важная часть MSF. Методология включает детально проработанную Модель группы. Модель группы возникла как ответ на потребность в четком понимании ролей, обязанностей и задач каждого члена проектной команды (табл. 7.12). Считается, что такая модель способствует мотивации сотрудников, упрощает работу и повышает эффективность их деятельности.

Таблица 7.12

Роли, цели и функциональные области MSF

Роль	Цели	Функциональные области
Управление продуктами	Обеспечить бизнес-ценность решения. Определить решение в рамках ограничений проекта. Обеспечить выполнение требований	Коммуникации. Аналитика. Планирование

Окончание табл. 7.12

Роль	Цели	Функциональные области
Управление программой	Реализовать проект в рамках ограничений. Реализовать средства, с помощью которых выполняются требования	Управление проектом. Управление программой. Управление ресурсами. Обеспечение выполнения. Управление качеством. Операции проекта
Разработка	Построить ИТ-решение в соответствии со спецификациями	Разработка ИТ-решения. Технологическое консультирование
Тестирование	Проверить соответствие ИТ-решения заданным условиям качества. Утвердить выпуск ИТ-решения	Все виды тестирования
Выпуск и эксплуатация	Развернуть ИТ-решение и обеспечить переход к эксплуатации. Обеспечить выполнение потребностей и ожиданий бизнес-подразделений заказчика	Управление выпусками. Инфраструктура. Операции. Управление сборками. Управление инструментами
Взаимодействие с пользователем	Оптимизировать удобство использования ИТ-решения. Обеспечить готовность пользователей к работе с ИТ-решением. Обеспечить выполнение требований и ожиданий пользователей	Специальные возможности. Взаимодействие со службой поддержки. Обучение. Удобство использования. Проектирование пользовательского интерфейса

Для крупных проектов с большими командами внедрения могут дополнительно создаваться группы направлений и функциональные группы, причем MSF даже предлагает таблицу совместимости ролей, показывающую, какие из них допустимо, нежелательно или совсем нельзя совмещать (табл. 7.13).

Важно понимать, что некоторые роли не должны выполняться одним человеком. К примеру, возложение ответственности по Управлению продуктом и Управлению программой на одного человека может привести к конфликту интересов, так как менеджер проекта должен контролировать основные процессы в области стоимости, сроков, взаимодействия команды, рисков и т. д., а не определять приоритеты бизнеса. Есть и роль, которую вообще не рекомендуется совмещать с другими, — роль Разработчика, что сделано исходя из предположения, что его активности являются наиболее критичными в проекте и любое наделение разработчиков дополнительными обязанностями приведет к срыву план-графика проекта.

Таблица 7.13

## Варианты совмещения ролей в MSF

Роль	Управление продуктами	Управление программой	Разработка	Тестирование	Выпуск и эксплуатация	Взаимодействие с пользователем
Управление продуктами		-	-	-	±	±
Управление программой	-		-	±	+	±
Разработка	-	-		-	-	-
Тестирование	+	±	-		+	+
Выпуск и эксплуатация	±	+	-	+		±
Взаимодействие с пользователем	+	±	-	+	±	

Таким образом, в отличие от многих других корпоративных методологий, определенные в MSF этапы/вехи, состав проектной группы, ролевая модель и другие элементы подходят не только для решений Microsoft. А значит, MSF представляет собой более гибкий и универсальный подход для внедрения других систем или программных продуктов.

**On Target.** Методология внедрения решений *On Target* была разработана компанией Navision для внедрения своих программных продуктов. После приобретения Navision корпорацией Microsoft было принято решение доработать *On Target*, к тому моменту содержащую шаблоны описаний бизнес-процессов, документации, организационных структур ИТ и квалификационных требований к специалистам (табл. 7.14).

Таблица 7.14

Этапы в методологии *On Target*

Этап	Цели	Действия
Подготовка проекта	Разработать основные документы. Сформировать команду	Предварительное планирование. Разработка базовых процедур. Формирование рабочей группы. Разработка и утверждение Устава проекта
Анализ	Подготовить команду. Сформировать функциональные требования к ИС	Обучение представителей заказчика. Анализ компании заказчика. Формирование и утверждение функциональных требований. Формирование плана и бюджета. Формирование технического задания

Этап	Цели	Действия
Проектирование	Сформировать нефункциональные требования к ИС. Сформировать принципы реализации требований	Проектирование реализации функциональных требований в ИС. Описание интерфейсов и модификаций. Уточнение плана и бюджета
Разработка и тестирование	Разработать ИС. Протестировать ИС	Разработка и тестирование функциональности. Разработка и тестирование интерфейсов. Разработка модификаций и дополнений
Развертывание	Развернуть систему на предприятии заказчика	Развертывание на рабочие места. Настройка прав и уровней доступа. Верификация начальных данных и операций. Перенос данных. Обучение и подготовка инструкций на рабочие места
Эксплуатация	Запустить ИС в эксплуатацию. Осуществить сдачу-приемку ИС	Запуск ИС в эксплуатацию. Опытная эксплуатация ИС. Сдача-приемка ИС

В силу того что к моменту приобретения Navision у Microsoft уже применялись свои проверенные корпоративные методологии MSF и MOF, в дальнейшем On Target была дополнена и к моменту выведения на рынок Microsoft Dynamics превратилась в результат доработок в MS Dynamics Sure Step/Microsoft Business Solutions Partner Methodology.

**Microsoft Business Solutions Partner Methodology. MBS Partner Methodology** — это дальнейшее развитие On Target. Эта методология ставит своей целью не просто создание ИС, но также максимальное удовлетворение потребностей заказчика.

Как можно видеть на рис. 7.18, этапы не сильно отличаются от аналогичных в On Target. На этапе Диагностики производится анализ и описание бизнес-процессов компании заказчика, а также выявляются ключевые бизнес-потребности. Производится первоначальное определение бюджета, сроков, границ и результатов проекта. Создается рабочая группа, причем сотрудники заказчика, вошедшие в нее, содействуют в проведении диагностики предприятия. В конце стадии формируются отчеты о проведенной диагностике, фиксируются ограничения проекта, документируются предложения по разработке и внедрению ИС.



Рис. 7.18. Этапы проекта в Microsoft Business Solutions Partner Methodology

Роли в проекте приведены на рис. 7.19.

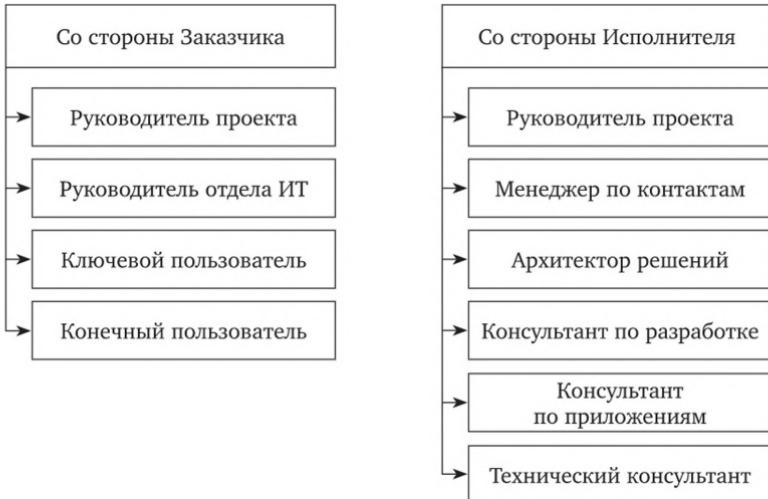


Рис. 7.19. Роли в проекте по методологии MBS Partner Methodology

На этапе Анализа окончательно формируются Управляющий комитет и проектная группа. Формируются такие документы, как План проекта, Устав проекта, Порядок отчетности, а также определяются условия сдачи-приемки и принципы управления изменениями и рисками. Сотрудники клиента знакомятся с базовой функциональностью продукта за счет проведения тренингов. Требования к ИС уточняются и детализируются, в результате чего формируется Спецификация функциональных требований. Формируются первые варианты интерфейсов, а предприятие начинает готовиться к изменению бизнес-процессов.

На этапе Проектирования на основе Технического задания формируется Программный дизайн, который описывает функциональность ИС, интерфейсы с внешними ИС, порядок тестирования, разработки и приемки. На данном этапе производится также планирование контроля качества, уточняются сроки и ресурсы проекта.

Вслед за этим начинается этап *Разработки и тестирования*. В самом начале этапа производится настройка среды для разработки, среды для тестирования и среды для интеграции. Производится первоначальная реализация функций и интерфейсов, которые сразу же тестируются. Вслед за этим результаты разработки передаются заказчику, который также производит тестирование. Производится исправление ошибок, а заказчик корректирует требования. Далее разработка и тестирование повторяются. Наконец, наступает комплексное тестирование ИС у заказчика. Производится финальное обнаружение ошибок и изменение требований. Все результаты разработки после этого объединяются в единую рабочую среду. Система настраивается, в нее переносятся данные. Конец этапа — это финальные испытания и начало подготовки к сдаче-приемке.

*Развертывание* начинается с официальной сдачи проекта заказчику. Заказчик оценивает достижение целей проекта на основе определенных ранее критерии успеха. Далее начинается подготовка системы к запуску в промышленную эксплуатацию. Производится обучение конечных пользователей. Осуществляется первоначальная поддержка специалистами исполнителя. В конце этапа происходит официальное завершение проекта, который оценивается заказчиком.

И, наконец, наступает этап *Начального сопровождения*. Выполняется регулярная (и даже ежедневная) поддержка работы заказчика с ИС. Система периодически обновляется, причем в новых версиях устраняют ошибки. Производится мониторинг изменения требований заказчика. Может начаться планирование новых проектов.

### 7.2.2. Oracle Unified Method

Oracle Unified Method (OUM) — комплекс методов, который охватывает большую часть стадий ЖЦИС. Каждый проект в методологии OUM состоит из пяти фаз:

- начальная фаза (*Inception*);
- проектирование (*Elaboration*);
- разработка (*Construction*);
- переход к эксплуатации (*Transition*);
- эксплуатация (*Production*).

Каждая фаза может включать до 15 процессов. Сами процессы и их интенсивность на каждой из фаз отражены на рис. 7.20<sup>1</sup>.

Несмотря на видимую схожесть с RUP, данный подход не является итерационным, он пошаговый.

---

<sup>1</sup> Девятов С. Методология Oracle Unified Method // Блог С. Девятова. 2011. 20 нояб. URL: <http://stan1slav.blogspot.ru/2011/11/oracle-unified-method.html> (дата обращения: 07.10.2020).

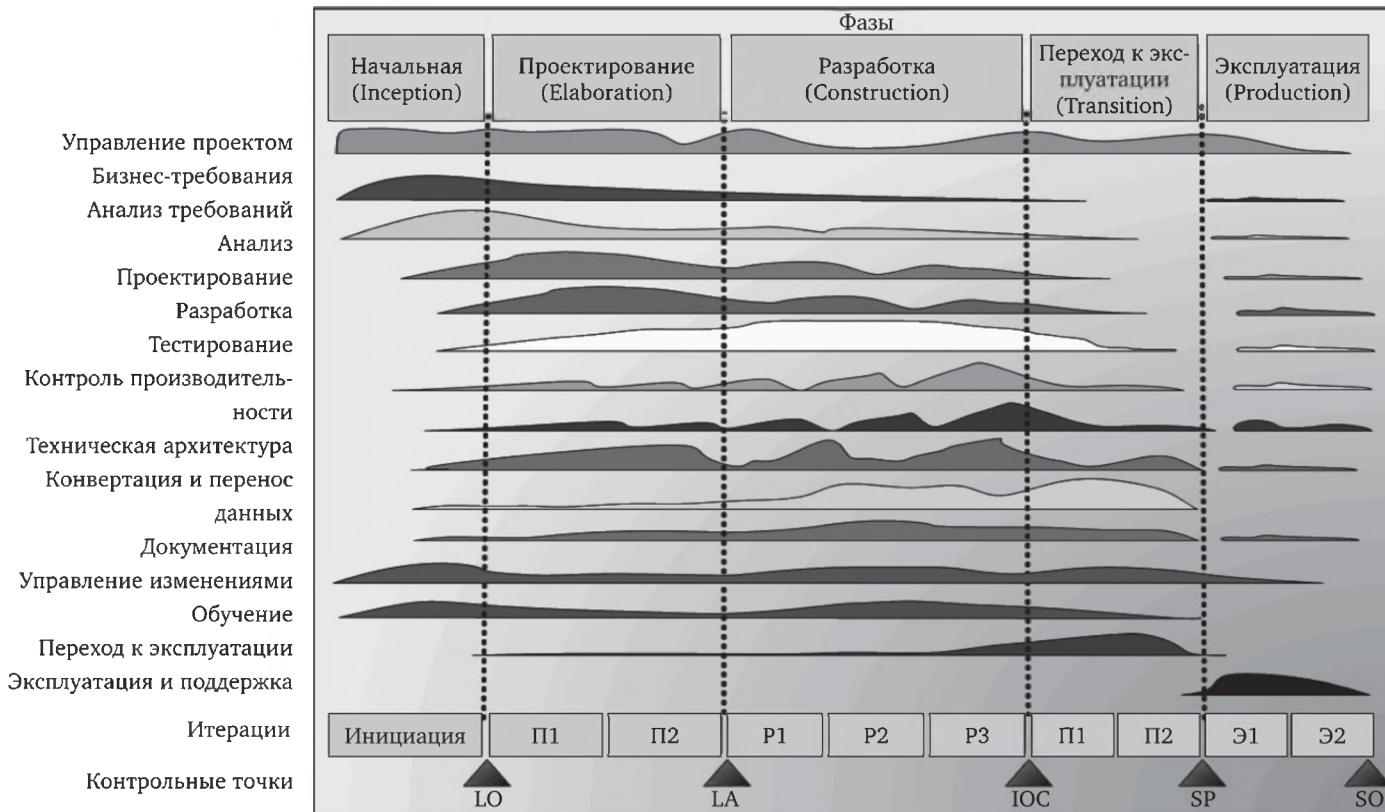


Рис. 7.20. Фазы и процессы Oracle Unified Method

**Начальная фаза. Цели.** Каждая фаза ОУМ преследует достижение каких-либо целей. Для начальной фазы выделяют следующие цели:

- понять, какого результата нужно достичь по завершении проекта и какой продукт для этого потребуется;
- определить масштабы проекта и выделить его составные части;
- определить и ранжировать риски, вместе с этим разработать стратегию управления рисками;
- определить лицо (со стороны заказчика), ответственное за проект, а также определить вовлеченных в проект пользователей заказчика;
- разработать план проекта;
- обучить проектную команду ОУМ и определить роли;
- сформировать требования заказчика в формализованном виде;
- разработать концептуальный прототип, отражающий функциональные требования;
- определить будущих тестеров и разработать требования к тестам производительности.

**Процессы.** Процессы для начальной фазы ОУМ приведены в табл. 7.15.

Таблица 7.15

**Процессы начальной фазы ОУМ<sup>1</sup>**

Процесс	Описание
Бизнес-требования	Определение функциональных требований заказчика (на основе MoSCoW <sup>2</sup> ). Определение нефункциональных требований заказчика (на основе MoSCoW)
Анализ требований	Создание модели прецедентов, которая отражает основные бизнес-требования заказчика в виде прецедентов
Внедрение	Моделирование концептуального прототипа системы (иллюстрирует интерфейс, функциональные возможности ИС). Проработка ключевых технических вопросов для проверки выполнимости требований заказчика
Тестирование	Определение начальных требований к тестам
Контроль производительности	Определение, разработка и выполнение шагов по контролю производительности. Определение производительности критических бизнес-транзакций. Определение системы показателей контроля производительности

<sup>1</sup> MoSCoW (от англ. *Must have, Should have, Could have, Won't have*) — метод приоритизации требований, в котором каждому требованию присваивается один из четырех уровней приоритета.

Процесс	Описание
Техническая архитектура	Ознакомление с принципами работы эталонного варианта архитектуры предприятия заказчика. Сбор информации о требованиях к разрабатываемой архитектуре и ее компонентам
Конвертация и перенос данных	Определение компонентов и объектов с данными для переноса. Определение требований по конвертации данных. Определение методов загрузки данных в систему. Выбор технологии для переноса
Документация	Определение набора требований к документации. Разработка стратегии документирования
Обучение	Разработка плана обучения проектной группы

**Проектирование. Цели.** Фаза проектирования преследует достижение следующих целей:

- расставить приоритеты в прецедентах и сформировать дополнительные требования;
- детализировать бизнес-требования и проанализировать их;
- сформировать бизнес-требования в виде прецедентов;
- согласовать внешний вид интерфейсов;
- согласовать участие в проекте представителей заказчика;
- разработать базовую техническую модель (включая компоненты);
- разработать план устранения рисков;
- убедиться, что возможен переход к следующей фазе.

**Процессы.** Процессы для фазы проектирования ОУМ приведены в табл. 7.16.

Таблица 7.16

#### Процессы фазы проектирования ОУМ

Процесс	Описание для фазы
Бизнес-требования	Разработка спецификации программных требований (в соответствии с методом MoSCoW)
Анализ требований	Выявление путей снижения рисков. Анализ того, какие прецеденты не могут быть автоматизированы на основе стандартной функциональности. Анализ потребности в дополнительной разработке
Анализ	Анализ прецедентов и структуризация с помощью схематической объектной модели (модель анализа)
Разработка	Разработка класса архитектуры, программных компонентов и их интерфейсов. Проектирование баз данных

Окончание табл. 7.16

Процесс	Описание для фазы
Внедрение	Разработка функционального прототипа (на основе самых важных прецедентов). Разработка архитектурного прототипа (на основе архитектурно значимых прецедентов). Разработка прототипа пользовательского интерфейса (определяется стандартами пользовательских интерфейсов в приложениях)
Тестирование	Разработка стратегии тестирования. Планирование тестов. Разработка сценариев тестирования
Контроль производительности	Определение требований к контролю производительности (и стратегии контроля производительности). Определение стратегии тестирования производительности. Определение моделей и сценариев тестов производительности. Разработка программ проведения тестов производительности. Определение данных, которые будут использоваться при тестировании производительности
Техническая архитектура	Определение требований к архитектуре. Определение стратегии доступа к информации. Определение стратегии аварийного восстановления. Определение стратегии создания резервных копий. Определение стратегии управления ИС. Определение стратегии интеграции с другими ИС. Разработка стратегии безопасности и контроля
Конвертация и перенос данных	Разработка стратегии конвертации и переноса данных. Сортировка и упорядочивание данных (в методологии ОУМ выполняется заказчиком)
Документация	Разработка стандартов и процедур документирования
Обучение	Обучение проектной команды. Подготовка организации к внедрению ИС
Переход к эксплуатации	Разработка стратегии перехода по одному из методов. Последовательный метод (отключение элементов старой ИС и ввод новых). Параллельный метод (работа новой и старой ИС одновременно). Метод замещения (полная остановка старой ИС и запуск новой)

**Разработка. Цели.** Фаза разработки преследует достижение следующих целей:

- описать оставшиеся бизнес-процессы;
- подготовиться к стадии перехода к эксплуатации;

- протестировать ИС и инфраструктуру;
- сформировать документацию;
- подготовить процесс обучения конечных пользователей ИС.

**Процессы.** Процессы для фазы разработки ОУМ приведены в табл. 7.17.

Таблица 7.17

**Процессы фазы разработки ОУМ**

Процесс	Описание
Бизнес-требования	Могут быть выявлены новые требования, которые нужно учесть
Анализ требований	Хотя компоненты спроектированы и разработаны, могут быть определены новые бизнес-процессы. Тогда модель бизнес-процессов изменяется
Анализ	Совершенствуются результаты, полученные на фазе проектирования
Разработка	Настройка ИС для соответствия функциональным и нефункциональным требованиям. Разработка оставшихся прецедентов
Внедрение	Внедрение и тестирование программных компонентов. Конечные пользователи оценивают соответствие требованиям и указывают на недостатки. Система внедряется на основе исходных кодов, скриптов, исполняемых файлов
Тестирование	Тестирование компонентов ИС и всей ИС. Тестирование нефункциональных требований
Контроль производительности	Разработка компонентов, необходимых для проведения тестирования производительности
Техническая архитектура	Создание руководства по управлению системой. Тестирование систем резервного копирования и восстановления. Определение заключительной платформы и сетевой архитектуры
Конвертация и перенос данных	При необходимости производится повтор действий из фазы проектирования
Документация	Публикуется документация ИС и интерактивный справочник
Обучение	Подготовка к проведению тренингов по обучению конечных пользователей ИС. Проведение учебных тренингов
Переход к эксплуатации	Усовершенствование стратегии перехода. Разработка плана установки новой ИС

**Переход к эксплуатации. Цели.** Фаза перехода к эксплуатации преследует достижение следующих целей:

- получить одобрение проекта со стороны заказчика;
- провести приемо-сдаточные испытания и решить возникшие проблемы;
- убедиться, что ИС соответствует требованиям заказчика;
- подготовить промышленную среду;
- завершить конвертацию данных.

**Процессы.** Процессы для фазы перехода к эксплуатации ОУМ приведены в табл. 7.18.

Таблица 7.18

**Процессы фазы перехода к эксплуатации ОУМ**

Процесс	Описание для фазы
Тестирование	Проведение приемо-сдаточных испытаний (с привлечением дополнительных пользователей со стороны заказчика). Рекомендуется использовать конвертированные данные
Контроль производительности	Проведение настройки и конфигурирования ИС для достижения соответствия требованиям
Конвертация и перенос данных	Проверка правильности конвертации и переноса данных
Документация	Документация обновляется в случае изменения ИС
Обучение	Продолжение обучения конечных пользователей. Проведение оценки эффективности ИС
Переход к эксплуатации	Подготовка промышленной среды. Ввод системы в эксплуатацию

**Эксплуатация. Цели.** Фаза эксплуатации преследует достижение следующих целей:

- выполнить обязательства гарантийного периода;
- мониторить производительность ИС и устранять недостатки;
- создать журнал регистрации ошибок и вносить исправления;
- оценить эффективность внедренного решения;
- разработать план расширения ИС.

**Процессы.** Процессы для фазы эксплуатации ОУМ приведены в табл. 7.19.

Таблица 7.19

**Процессы фазы эксплуатации ОУМ**

Процесс	Описание
Контроль производительности	Проведение промышленного контроля производительности

Процесс	Описание
Эксплуатация и поддержка	Разработка списка расширений ИС (на основе метода MoSCoW). Оценка соответствия ИС требованиям к производительности. Контроль внесения изменений в ИС

## 7.3. Российские и международные стандарты

### 7.3.1. РМВоК

Вряд ли есть менеджеры, не слышавшие о РМВоК. Свод знаний по управлению проектами (Project Management Body of Knowledge) интегрирует в себе множество сведений, связанных с проектным управлением, основываясь на лучших мировых практиках.

Различные стандарты и методологии предлагают разные определения проекта:

---

**Проект** — предпринимаемое усилие, организующее человеческие, материальные и финансовые ресурсы в неизвестный путь в рамках уникального предмета работы, заданной спецификации, с ограничениями на затраты и время, с тем, чтобы следование стандартному жизненному циклу проекта приводило к осуществлению успешных изменений, определенных посредством количественных и качественных целей и задач.

---

или

---

**Проект** — уникальный процесс, состоящий из набора взаимоувязанных и контролируемых работ с датами начала и окончания и предпринятый, чтобы достичь цели соответствия конкретным требованиям, включая ограничения по времени, затратам и ресурсам.

---

Именно данные идеи и реализует РМВоК, рассматривая аспекты времени, затрат и состава работ особенно детально. Важно, что данные ограничения взаимозависимы, а значит, сжатые сроки приводят к увеличению стоимости проекта, увеличение состава работ — также к увеличению стоимости и (или) времени выполнения проекта. Грамотное управление этими тремя рычагами дает возможность менеджеру предпринимать упреждающие действия.

Начинается РМВоК с описания общего жизненного цикла управления проектом и его основных групп процессов:

- инициация;
- планирование;

- исполнение;
- мониторинг и управление;
- завершение проекта.

Эти группы являются достаточно универсальными и при дальнейшей детализации могут служить основой любых отраслевых стандартов.

### Пример

---

PMBoK рассматривает следующие процессы исполнения:

- руководство и управление исполнением процесса;
- подтверждение качества;
- набор команды проекта;
- развитие команды проекта;
- управление командой проекта;
- распространение информации;
- управление ожиданиями заинтересованных сторон;
- осуществление закупок.

---

Более интересны выделяемые PMBoK области знаний:

- управление интеграцией;
- управление содержанием;
- управление сроками;
- управление стоимостью;
- управление качеством;
- управление человеческими ресурсами;
- управление коммуникациями;
- управление рисками проекта;
- управление поставками;
- управление стейххолдерами (область знаний, появившаяся только в 5-й версии PMBoK, опубликованной в 2013 г.).

Этим областям знаний посвящены соответствующие разделы, и для каждой из них предлагается ряд активностей, методик, инструментов и других видов знаний, зарекомендовавших себя на практике.

Рассмотрим структуру и содержание PMBoK на примере области знаний «Управление содержанием» (рис. 7.21). По структуре остальные главы PMBoK аналогичны.

### Пример

---

1. В первую очередь PMBoK дает *определение области знаний*. «“Управление содержанием проекта” включает в себя процессы, обеспечивающие включение в проект тех и только тех работ, которые необходимы для успешного завершения проекта. “Управление содержанием проекта” непосредственно связано с определением и контролем того, что включено и что не включено в проект» (PMBoK v4).



Рис. 7.21. Группы активностей по PMBoK

2. Приводится общая схема области знаний.

На ней указываются все основные процессы, а также относящиеся к каждому из них «Входы», «Инструменты и методы», а также «Выходы».

«Входы» обозначают необходимую для исполнения процесса информацию (например, «Реестр заинтересованных сторон», на основании которого определяются основные лица, для которых организуется сбор требований).

«Инструменты и методы» — основные активности и методы их осуществления (семинары, анкеты, групповые творческие методы, при помощи которых определяются содержание и иерархическая структура работ).

«Выходы» — ключевые результаты процесса (например, «документация по требованиям» и «матрица отслеживания требований», сформированная на основании осуществленных в ходе процесса активностей).

Всего в данной области знаний выделяется пять групп активностей: «Сбор требований», «Определение содержания», «Содержание иерархической структуры работ», «Подтверждение содержания», «Управление содержанием».

3. Рассмотрим один из процессов выбранной области знаний подробнее (рис. 7.22).



Рис. 7.22

На этапе сбора требований в числе прочих предлагаются групповые творческие методы (пп. 4): мозговой штурм, метод номинальных групп (мозговой штурм голосование), метод Дельфи (ответы на анкету группы экспертов в течение нескольких раундов), интеллект-карты (выявление сходств и различий в понимании среди идей мозгового штурма), диаграммы сходства (группировка идей). Для каждого из этих методов приводятся формулы и инструкции по применению.

PMBoK приводит также подробные описания процессов и формируемых в процессе работы с ними документов и результатов (обобщенный пример такого описания показан на рис. 7.23).



Рис. 7.23. Образец организованной по фазам иерархической структуры работ

Каждая вновь издаваемая версия PMBoK учитывает наиболее важные тенденции опыта управления проектами. Пятая версия (изданная в 2013 г.) также содержит несколько нововведений: например, к ним относятся четыре новых процесса планирования управления (содержанием, расписанием, стоимостью и заинтересованными сторонами), которые развиваются концепцию формирования плана управления проектом из нескольких вспомогательных планов. Также в ней произведены изменения содержания некоторых групп процессов и даже добавлена одна новая область знаний («управление заинтересованными лицами»), по сути частично основанная на управлении коммуникациями (концентрирующемся теперь исключительно на взаимодействии внутри проектной команды).

Однако, несмотря на всю свою распространенность, РМВоК не является единственной доступной методологией, и для предметной области ИТ-проектов применяются также PRINCE2, MSF, RUP и многие другие. Они могут оперировать такими параметрами, как число стадий и областей знаний, роли и процедуры управления в проектах, варьируя их в различной степени.

### Пример

В РМВоК критериями успешности можно считать соответствие треугольнику ограничений — составу работ, срокам, стоимости, а также качеству. PRINCE2 предлагает для оценки проекта фокусироваться на аспектах финансовой успешности, положительном мнении конечного пользователя и косвенной пользе для исполнителя (три перспективы: заказчик, исполнитель и пользователь). В то же время Scrum, будучи еще более ориентированным на заказчика, рассматривает единственный фактор: удовлетворенность клиента полученными результатами (проект успешен, если заказчик готов продолжать сотрудничество с исполнителем и (или) рекомендовать его другим).

Переходя от эталонных рекомендаций РМВоК к более практической стороне управления проектами внедрения, отметим ряд особенностей и принципов.

**Этапность.** Масштабные проекты разбиваются на этапы (подэтапы), продолжительность выполнения которых не должна превышать двух-трех месяцев и суть которых не обязательно должна однозначно соответствовать fazam жизненного цикла ИС. Более важно, что каждый этап работ должен иметь законченный результат, полезный для предприятия, независимо от хода и результата выполнения всего проекта. На каждом этапе проводятся работы, необходимые в том числе для реализации последующих этапов, однако некоторые из них должны выполняться строго последовательно, в то время как другие можно реализовывать параллельно. Процесс планирования можно осуществлять с помощью встроенных средств MS Project либо любых других инструментов автоматизированного управления проектами и потоками работ. Это важно для поддержки таких активностей, как оперативный контроль хода выполнения проекта, оценка потребных ресурсов (человеческих, материальных, временных и т. д.).

**Результативность.** Каждый этап проекта характеризуется наличием ясно определяемых и оцениваемых результатов. Критерии оценки результатов определяются при планировании проекта. Если общий процесс внедрения занимает более полутора лет, проект планируется таким образом, чтобы первые результаты были получены уже через пять-семь месяцев после его начала. Это дает дополнительную мотивацию сотрудникам предприятия, которые видят конкретные результаты проекта. Внедрение выполняется помодульно

и начинается с тех модулей, функционирование которых необходимо для внедрения других модулей (либо с тех, которые способны достаточно быстро принести реальную отдачу).

**Постоянный контроль.** Необходимо постоянно держать под контролем как сам процесс выполнения работ, так и достижение целей их выполнения. По окончании каждого этапа или подэтапа осуществляется анализ степени изменения внешних и внутренних факторов проекта и определяется необходимость корректировки дальнейших планов. Контроль проводится посредством составления и анализа графика выполнения работ по внедрению системы, а также периодических отчетов должностных лиц проекта на каждом его этапе.

**Сквозное документирование проекта.** Все результаты выполняемых работ, а также возникающие в процессе работы дополнения, замечания, разногласия, предложения в обязательном порядке оформляются в виде соответствующих документов, причем помимо их согласования и утверждения даже получение пакета документов должно фиксироваться подписью ответственного лица.

**Активное вовлечение заказчика в проект внедрения.** Как уже обсуждалось выше, группа внедрения от заказчика должна формироваться из руководителей среднего звена и ключевых специалистов предприятия. Функции специалистов группы внедрения от заказчика и исполнителя определяются при планировании проекта и в дальнейшем могут корректироваться в начале работ по каждому этапу проекта.

Без активного участия специалистов заказчика в процессе внедрения успешная реализация проекта внедрения практически невозможна. Работа консультантов подрядчика и сотрудников заказчика в одной команде значительно повышает эффективность проекта внедрения системы. Для успеха проекта в целом группе внедрения необходимы общее видение проблемы, нацеленность на один и тот же положительный результат, заинтересованность каждого и группы в целом в достижении этого результата.

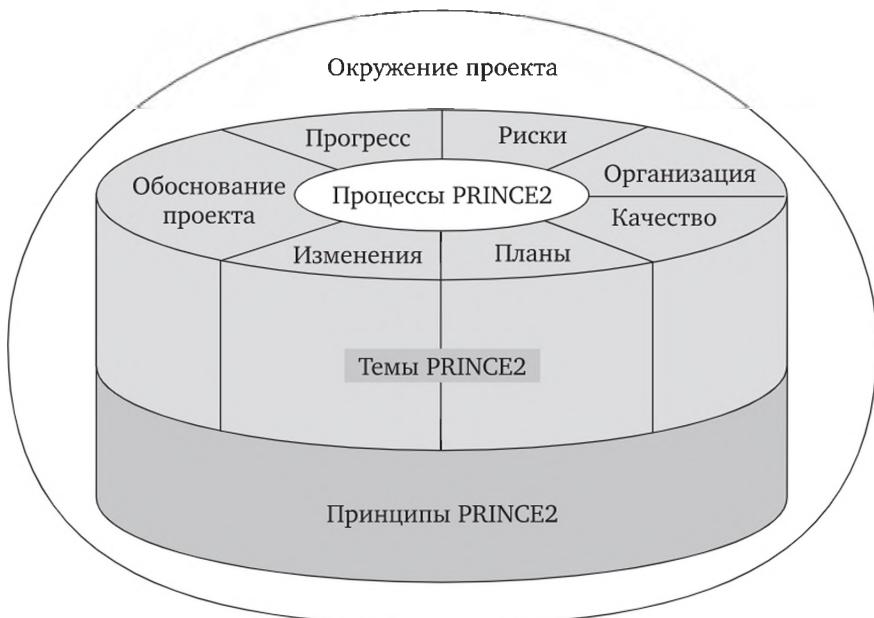
### 7.3.2. PRINCE2

---

**Методология управления программами/проектами в организации PRINCE2 (PRojects IN Controlled Environments)** — методология, концентрирующаяся на организации, менеджменте и мониторинге хода ИТ-проектов с точки зрения продукта.

---

Методология PRINCE2 включает семь принципов, семь тем (далее — элементов) и семь процессов (для каждого из которых определяются входные и выходные объекты, цели и ключевые активности) (рис. 7.24).



**Рис. 7.24. Структура методологии PRINCE2<sup>1</sup>**

**Принципы.** PRINCE2, будучи процессно-ориентированной методологией проектного управления, базируется на семи основных принципах (табл. 7.20).

Роль проектной команды в данном случае может быть описана пятью ключевыми активностями:

- адаптация семи элементов PRINCE2 к условиям проекта;
- применение и адаптация специфической терминологии;
- адаптация описаний продукта к условиям проекта;
- адаптация описаний ролей в соответствии с PRINCE2;
- адаптация процессов для соответствия вышеописанным принципам.

**Таблица 7.20**

**Принципы PRINCE2**

Принцип	Описание
Постоянное бизнес-обоснование	Каждый проект должен быть целесообразен и основан с точки зрения соотношения затрат и выгод от его реализации
Обучение на собственном опыте	Проектные команды должны непрерывно обучаться, анализировать опыт предыдущих проектов (основные проблемы, риски, преимущества) и делиться выводами

<sup>1</sup> По материалам сайта: <http://www.isaudit.ru>.

Принцип	Описание
Определение ролей и ответственности	Организационная структура должна учитывать закрепление ответственности за членами проектной команды в части учета интересов всех сторон: бизнес-заказчиков, пользователей, поставщиков и потребителей
Управление по стадиям	Планирование, мониторинг и контроль проектов на каждом этапе
Управление исключениями	Для каждого проекта определены конкретные критические значения показателей, уровни и механизмы эскалации возникающих проблем
Фокус на продуктах	Определение требований к качеству результатов проекта и их характеристик
Адаптация к условиям конкретного проекта	Адаптация методологии в зависимости от масштабов, сложности, важности и окружения конкретного проекта

Все эти активности и их особенности подробно описываются в тексте методологии.

**Элементы. Бизнес-кейс (обоснование проекта)** дает ответ на вопрос «Почему?» (рис. 7.25). Основная цель создания подобного бизнес-кейса состоит в создании механизмов оценки целесообразности (и возможности) реализации проекта.

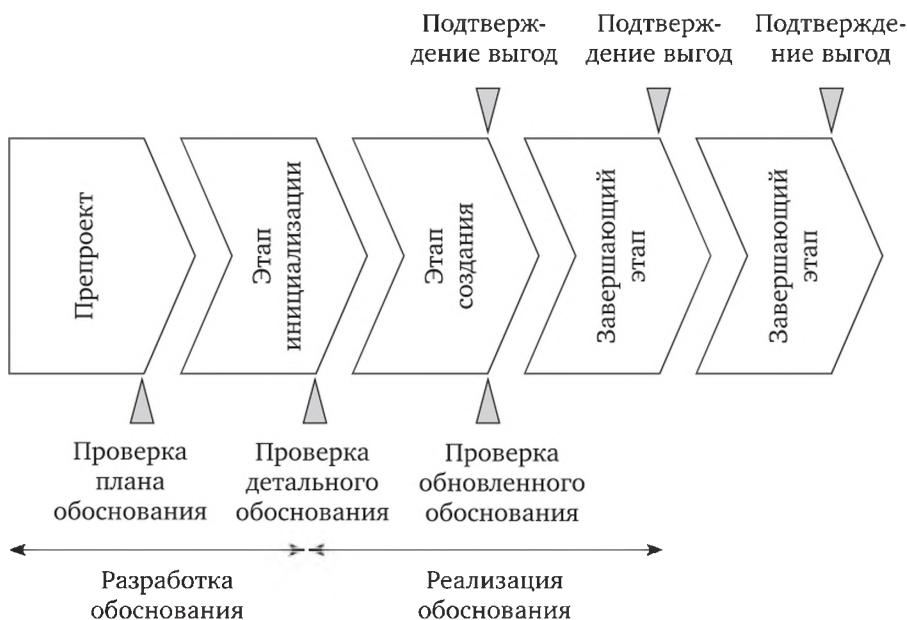


Рис. 7.25. Бизнес-кейс (обоснование проекта)

- Организация** = «Кто?». Определение и создание проектной структуры ответственности.
- Качество** = «Что?». Определение методов создания качественного продукта (включая аспект контроля качества).
- Риск** = «Что, если?». Определение, оценка и контроль всех рисков.
- Планы** = «Как? Сколько? Когда?». Определение способов создания и распространения продуктов, прогнозирование динамики спроса.
- Изменения** = «Какое влияние?». Определение эффекта от внедрения одобренных изменений проекта.
- Прогресс** = «Где мы сейчас? Куда направляемся?». Создание механизмов мониторинга и оценка реальных достижений для планирования целей проекта, в том числе для ответа на вопрос: стоит ли продолжать двигаться в выбранном направлении?

**Процессы.** И наконец, PRINCE2 предлагает собственную карту процессов управления проектом<sup>1</sup> (рис. 7.26). Эти процессы описаны в табл. 7.21.



Рис. 7.26. Карта процессов управления проектом в методологии PRINCE2

Таким образом, PRINCE2 является достаточно гибкой методологией проектного управления, которая обеспечивает четкую структуру, предоставляет общие методы и терминологию проектного управления. Методология PRINCE2 может быть адаптирована для

<sup>1</sup> По материалам сайта: <http://www.isaudit.ru>.

любого типа проектов и отраслей. Возможность дополнительно получить сертификацию своих знаний по PRINCE2 делает ее популярной среди консультантов и компаний-подрядчиков.

Таблица 7.21

**Процессы управления проектом в методологии PRINCE2**

Процесс	Описание
Запуск проекта	Обоснование основных причин осуществления проекта, его задач и ожидаемого результата, при этом запуск проекта должен осуществляться в очень короткие сроки
Руководство проектом	Координация проекта, определение ролей команды проекта, создание плана этапов
Инициирование проекта	Обоснование необходимости проекта (например, технико-экономическое обоснование), планирование ресурсов
Управление границами этапа	Мониторинг выполнения планов, контроль ключевых показателей проекта, предоставление информации спонсорам и заказчикам проекта (управление коммуникациями)
Контроль этапа	Мониторинг и контроль, осуществляющийся лидером проекта (определение подлежащих решению задач, сбора информации, предоставление отчетов, принятие необходимых корректирующих действий)
Управление созданием продукта	Обеспечение создания требуемых в ходе проекта продуктов
Завершение проекта	Обеспечение контролируемости стадии завершения проекта, контроль соответствия результатов документированным положениям при инициации проекта, подготовка финальных отчетов, обеспечение последующей поддержки и обучения

### 7.3.3. ISO 21500:2012

**Международный стандарт:** ISO 21500 Guidance on project management.

**Российский аналог:** ГОСТ ИСО 21500—2014 «Руководство по проектному менеджменту».

ISO 21500:2012 изначально создавался как стандарт, регламентирующий проектное управление на международном уровне. Разработка стандарта велась силами трех рабочих групп, ответственных за различные направления:

- разработка терминологии;
- регламентация процессов управления проектами;
- определение и описание в приложениях ключевых концепций проектного менеджмента.

Один из самых «молодых» стандартов, он во многом схож с РМВоК. В первые же годы использования этот стандарт стал достаточно популярен.

В ISO 21500:2012 приведены описания понятий и процессов, формирующих управление проектами, однако детального руководства не приводится. Сам документ предназначен в первую очередь для топ-менеджеров и спонсоров проекта, а также для менеджера проекта и проектной команды. Благодаря ISO 21500 они получают знания о принципах и практиках управления проектами, а также некое эталонное представление об актуальных практиках управления.

Структура ISO 21500 приведена ниже.

#### 1. Общие положения.

2. Термины и определения (в том числе «операция», «запрос на изменения», «критический путь», «лаг», «лид», «кривая обучения», «словарь ИСР» и пр.).

#### 3. Концепция управления проектами.

Например, общий обзор, среда проекта (в том числе программы и портфели проектов), внешнее руководство проекта, заинтересованные стороны (рис. 7.27), компетенции участников проекта, ограничения проекта.



Рис. 7.27. Схема заинтересованных сторон процесса

#### 4. Процессы управления проектами.

Процессы управления проектами рассматриваются с двух ракурсов: с точки зрения управления проектом как групп процессов (применимых к конкретным фазам или проекту в целом), а также с точки зрения группировки процессов по предметным областям.

Предметные области состоят из процессов, которые могут относиться к любой фазе проекта и не зависят от отрасли или специфики.

Взаимодействие групп процессов, их входы и выходы приведены в Приложении 8.

Процессы управления проектами по предметным группам приведены в Приложении 9.

5. Приложения. Блок-схемы взаимодействия отдельных процессов в каждой группе процессов.

### Пример

Рассмотрим описание одного из процессов на примере «Управления содержанием». Одним из его подпроцессов является 4.3.13 «Определение работ». Приводится цель («Выявление, определение и документирование всех операций, которые должны быть запланированы и осуществлены в целях достижения целей проекта»), а также таблица входов/выходов (табл. 7.22).

Таблица 7.22

Таблица входов и выходов по стандарту ISO 21500

Основные входы	Основные выходы
Иерархическая структура работ. Словарь иерархической структуры работ. Планы проекта. Утвержденные изменения	Перечень работ

### 7.3.4. ГОСТ Р 54869—2011

Национальный стандарт ГОСТ Р 54869—2011 «Проектный менеджмент. Требования к управлению проектом».

Стандарт ГОСТ Р 54869—2011 принадлежит к серии стандартов, в которую также входят:

- ГОСТ Р 54871—2011 «Требования к управлению программой»;
- ГОСТ Р 54870—2011 «Требования к управлению портфелем проектов».

Стандартом определяются следующие процессы управления проектом:

- процесс инициации проекта;
- процесс планирования управления изменениями в проекте;
- процесс планирования содержания проекта;

### Пример

Цель процесса: определение требований проекта и состава работ проекта.

Выходы процесса:

- а) требования к проекту со стороны заказчика, других заинтересованных сторон проекта, а также законодательства и нормативных актов

определенны, проанализированы на предмет возможности их выполнения, согласованы с заказчиком проекта и документированы;

б) определены, согласованы с заказчиком и документированы ключевые данные по продукту проекта, а именно:

1) назначение, свойства и характеристики продукта,

2) критерии и методы приемки продукта проекта и его составных частей,

3) допущения и исключения, касающиеся продукта проекта;

в) определены, согласованы с заказчиком и документированы работы проекта, а также допущения и исключения, касающиеся работ проекта.

---

- процесс планирования расписания проекта;
- процесс планирования бюджета проекта;
- процесс планирования персонала проекта;
- процесс планирования закупок в проекте;
- процесс планирования реагирования на риски;
- процесс планирования обмена информацией в проекте;
- процесс организации исполнения проекта;
- процесс контроля исполнения;
- процесс завершения проекта.

В стандарте также приводится взаимосвязь основных элементов проектного управления.

## **Контрольные вопросы и задания**

1. Назовите методологии управления проектами от Microsoft, которые используются в проектах по разработке ИС и их особенности.
2. Что такое Oracle Unified Method?
3. Как SWEBOK применяется в проектном управлении?
4. Какие группы процессов, области знаний и принципы выделяются в РМВоК?
5. Из каких принципов, тем и процессов состоит методология PRINCE2?
6. Какие стандарты ISO и ГОСТ используются в проектном управлении?
7. Как осуществляется управление человеческими ресурсами со стороны заказчика и со стороны исполнителя?
8. На каких правилах и стандартах основывается управление качеством?
9. Какая документация должна быть создана в рамках управления содержанием?
10. Как происходит выбор и реализация стратегии управления рисками?
11. Когда возникает потребность в управлении программой или портфелем проектов?

## **Рекомендуемая литература**

1. Бэнко, К. Управление портфелями проектов: соответствие проектов стратегическим целям компании : перевод с английского / К. Бэнко, Ф. У. Мак-Фарлан. — Москва : Вильямс, 2007.

2. Грекул, В. И. Методические основы управления ИТ-проектами : учебник для вузов / В. И. Грекул, Н. Л. Коровкина, Ю. В. Куприянов. — Москва : Интuit, 2010.
3. Грекул, В. И. Учебный курс «Управление внедрением информационных систем» // Национальный открытый университет «ИНТУИТ». URL: <http://www.intuit.ru/studies/courses/2196/267/info> (дата обращения: 27.07.2020).
4. Грей, К. Ф. Управление проектами: практическое руководство : перевод с английского / К. Ф. Грей, Э. У. Ларсон. — Москва : Дело и Сервис, 2003.
5. Руководство PMBoK. — 5-е изд. — Project Management Institute, 2014.
6. Hinde, D. PRINCE2 Study Guide / D. Hindle. — N. Y. : SYBEX, 2012.

# **Тема 8**

## **ПРИМЕР БИЗНЕС-КЕЙСА**

### **И ПРАКТИЧЕСКОЕ ЗАДАНИЕ**

---

В этой теме содержится описание компании «Экспресс-доставка». Студентам предстоит ознакомиться с описанием и реализовать на практике основные виды деятельности, которые относятся к различным этапам жизненного цикла информационной системы. Помимо самого описания компании тема содержит контрольное задание, примеры его выполнения и соответствующие пояснения. Также тема включает дополнительные задания по бизнес-кейсу компании «Экспресс-доставка».

После изучения данной темы студент будет:

**знать**

- особенности текстовой формы представления информации о компании;
- особенности практической реализации проектов, относящихся к управлению жизненным циклом ИС;

**уметь**

- извлекать аналитическую информацию из предоставленного описания компаний;
- применять программные средства, использующиеся в профессиональной деятельности;
- применять основные компетенции, связанные с управлением жизненным циклом ИС;

**владеть навыками**

- формирования Устава проекта;
- формирования Реестра рисков проекта и Формы регистрации риска;
- использования программного средства MS Project;
- использования программного средства Bizagi Modeler для моделирования бизнес-процессов в нотации BPMN;
- использования программного средства ARIS Express для построения карты процессов и моделирования бизнес-процессов в нотации EPC;
- использования программного средства MS Visio для построения диаграммы «сущность-связь»;
- формирования требований к ИС на основе FURPS+;
- применения метода приоритезации MoSCoW;
- выбора готового программного решения на основе критериев;
- использования программного средства StarUML 5.0 для моделирования основных диаграмм UML;
- формирования спецификаций прецедентов;
- формирования плана тестирования.

## **8.1. Описание компании**

Данный кейс посвящен автоматизации предприятия экспресс-доставки, которое работает на российском рынке. Организация, деятельность которой будет изучаться в рамках этого кейса, ООО «Экспресс-доставка» занимается приемом, обработкой, хранением и доставкой стандартной корреспонденции и грузов.

ООО «Экспресс-доставка» было образовано в 1996 г. и начало свою деятельность с доставки обычных грузов по Москве. С течением времени компании удалось расширить спектр услуг и выйти на рынок междугородней доставки.

Организация обслуживает как физических, так и юридических лиц.

ООО «Экспресс-доставка» имеет центральный офис в Москве (осуществляющий общее руководство) и семь региональных филиалов в различных федеральных округах Российской Федерации. Организация имеет широкую сеть автомобильных маршрутов (более 500 по территории РФ).

Стратегические цели компании предполагают:

- удержание позиций на рынке доставки отправлений по России;
- улучшение сервиса и уровня удовлетворенности клиентов.

Для достижения этой цели руководство компании приняло решение выполнить программу проектов по радикальному обновлению бизнеса путем внедрения новых технологий и реинжиниринга бизнес-процессов.

Говоря об основных бизнес-процессах, в компании можно выделить направления:

- основная производственная деятельность;
- продвижение услуг компании на рынке (работа с клиентами).

Основная производственная деятельность заключается в перевозке отправлений от отправителя до получателя и обеспечении сохранности отправления на всем пути его следования (при приеме, обработке, перевозке и вручении отправлений). Всего организация работает с двумя видами отправлений:

- почтовая корреспонденция;
- посылки и упаковки.

Организационная структура компании представлена на рис. 8.1—8.2.

Рассмотрим подробнее функции некоторых подразделений ООО «Экспресс-доставка».

Управление информационных технологий и телекоммуникаций. Этот отдел занимается информационной и технической поддержкой процессов, протекающих в ООО «Экспресс-доставка». К ним от-

носятся: обслуживание аппаратной части (компьютеров, сети, серверов), поддержка программных продуктов, в том числе доработка и устранение инцидентов.

**Отдел обработки.** Отдел обработки занимается приемом отправлений в офисах и сортировкой этих отправлений. С отдела обработки начинается движение отправления по маршруту (они вручают груз курьерам при отправке на маршрут), а также оканчивается — при приеме отправлений от курьеров при их возвращении с маршрутов. Переадресация и возврат отправлений также осуществляются данным отделом, как и вручение отправлений в офисах компании.

**Управление логистики.** Одной из основных задач управления логистики является оптимизация и поддержка существующей сети маршрутов, принятие решений об объединении, изменении, отмене или вводе новых маршрутов. Анализируются и отслеживаются затраты на содержание маршрутов, оценивается рентабельность и востребованность маршрутов. Постоянно производится корректировка единой электронной базы маршрутов.



Рис. 8.1. Организационная структура главного офиса

На управление логистики возложена функция организации и обеспечения работы центра сбора и обработки информации ООО «Экспресс-доставка».

Управление логистики разрабатывает стандарты работы и отдельных операций, рекомендации и предложения по рационализации производственной деятельности ООО «Экспресс-доставка» в целях минимизации затрат предприятия при доведении материальных и информационных потоков от структурных подразделений ООО «Экспресс-доставка» до клиентов.

Управление логистики участвует в разработке и внедрении новых технологий обработки упаковок, тары, форм сопроводительных документов, а также выполняет подбор современных технических средств и оборудования автоматической идентификации объектов, осуществляя согласование с заинтересованными сторонами (контрагентами, отдельными корпоративными клиентами и др.).



Рис. 8.2. Организационная структура регионального филиала

**Отдел организации перевозок.** Занимается поддержкой маршрутов: распределением загрузки, контролем сроков отправки, составлением планов маршрутов, информированием кладовых о выдаваемых на маршрут отправлениях и т. д.

**Отдел обслуживания.** Занимается приемом и вручением отправлений у отправителей, а также вручением отправлений в офисах компании.

**Отдел перевозок.** Занимается перевозкой отправлений по плановым маршрутам. Его сотрудники ответственны за вручение отправлений курьерам перед маршрутом, вручение отправлений в офисах компаний, а также прием отправлений от маршрутных курьеров по возвращении с маршрутов.

**Автобаза.** Это выделенный отдел, основная функция которого — транспортное обеспечение подразделений ООО «Экспресс-доставка». Отдел:

- занимается приемом и выдачей автомобилей по заявкам;
- осуществляет учет и контроль автомобилей;
- проводит своевременное техническое обслуживание;
- осуществляет сбор и обработку данных по каждому автомобилю: пробег, учет заправок, время нахождения в пути и т. д.;
- по данным электронного мониторинга контролирует правильность учетных данных по эксплуатации транспортных средств;
- выявляет и проводит разбор нарушений: правил эксплуатации автомобилей, правил дорожного движения и установленных маршрутов.

### 8.1.1. Описание бизнес-процессов

Основные бизнес-процессы и участвующие в них подразделения компании перечислены в табл. 8.1.

Таблица 8.1  
Бизнес-процессы и занятые в них подразделения

Бизнес-процесс по направлениям	Подразделение
Прием отправлений в офисах	Отдел обработки
Прием отправлений у отправителей	Отдел обслуживания
Сортировка отправлений	Отдел обработки
Переадресация и возврат отправлений	Отдел обработки
Подготовка маршрутов	Отдел организации перевозок
Вручение отправлений курьерам при отправке на маршруты	Отдел обработки, отдел перевозок
Перевозка отправлений	Отдел перевозок
Прием отправлений от курьеров после возвращения	Отдел перевозок, отдел обработки
Вручение отправлений курьерами	Отдел обслуживания, отдел обслуживания, отдел перевозок

### Прием, обработка и доставка посылок и отправлений

**Прием отправлений в офисах.** Если прием отправлений производится в офисах, сотрудник получает от клиента все отправления

и сопроводительные документы на них (реестры в 2 экз.) и, в обязательном порядке, в присутствии клиента:

- проверяет правильность оформления сопроводительного документа;
- просчитывает количество отправлений и сличает его с итогом, указанным в сопроводительном документе;
- тщательно просматривает каждое отправление на целостность и правильность оформления;
- производит поименную проверку соответствия записей в реестре с данными, указанными на отправлениях;
- производит замер габаритов, взвешивает, в обоих экземплярах проставляет вес каждого отправления, сумму сбора, приемные номера;
- оформляет счет и другие финансовые документы;
- прикрепляет приемный номер.

После этого сотрудник расписывается на втором экземпляре реестра с указанием количества, даты и времени приема, заверяет его подписью с расшифровкой и оттиском штампа и отдает отправителю. Первый экземпляр реестра остается в подразделении.

В случае недостачи или неправильного оформления эти отправления вычеркиваются из обоих экземпляров, итог исправляется.

Далее отправления и документы подготавливаются для передачи курьерам для их последующей перевозки в отдел сортировки.

**Прием отправлений у отправителей.** Если прием отправлений производится в помещениях отправителя, курьер получает от клиента все отправления и сопроводительные документы на них (реестр в 2 экз.) и, в обязательном порядке, в присутствии клиента:

- проверяет правильность оформления сопроводительного документа;
- просчитывает количество отправлений и сличает его с итогом, указанным в сопроводительном документе;
- тщательно просматривает каждое отправление на целостность и правильность оформления;
- производит поименную проверку соответствия записей в реестре с данными, указанными на отправлениях;
- производит замер габаритов, взвешивает, в обоих экземплярах проставляет вес каждого отправления, сумму сбора, приемные номера;
- оформляет счет и другие финансовые документы.

В случае недостачи или неправильного оформления эти отправления вычеркиваются из обоих экземпляров реестра, итог исправляется.

После приема отправлений курьер проверяет правильность заполнения соответствующих граф контрольного листа.

Во второй копии реестра курьер указывает прописью количество принятых отправлений, дату и время, подписывается и ставит штамп. Эта копия вручается отправителю.

**Сортировка отправлений.** Сортировка происходит в два этапа: общая сортировка отправлений по маршрутам и детальная сортировка. Передача рассортированных пакетов для детальной сортировки осуществляется под роспись в передаточной ведомости с указанием количества прописью. На всех документах ставится оттиск штампа заполняющего их сотрудника.

После передачи отправлений на детальную обработку они сортируются по пунктам назначения. Сотрудник проверяет наличие отправления в базе по номеру. Они приписываются к реестрам в двух экземплярах: один направляется вместе с отправлением, второй — подшивается в производственные документы офиса, где формируется постпакет. При наличии трех и более пакетов, направляемых в один офис, они формируются в постпакет. Упаковка производится двумя работниками (на всех документах ставится по две подписи). Одновременно на одном рабочем месте упаковывается не более одного постпакета. Каждый пакет имеет свой номер, указанный на нем. На рассортированные отправления составляются реестры в двух экземплярах.

Транзитные пакеты и постпакеты также проходят сортировку. При необходимости они вскрываются, сверяются по вложенным документам. Данные об отправлениях вносятся в базу.

При отправке в один офис трех и более постпакетов или упаковок их приписывают к отдельной накладной, которая приписывается к общей накладной на данный маршрут с указанием номера, даты ее составления и количества приписанных к ней отправлений.

В случаях повреждения оболочки пакета проводится переоформление: заклеивание бумажной лентой с оттиском мастичной печати (без доступа к вложению) или переупаковка в дополнительную оболочку (в случае доступа к вложению). Составляется акт.

**Переадресация и возврат отправлений.** Переадресация или возврат отправлений осуществляется в случаях, когда отправление имеет неточный адрес или искаженное название. Каждое такое отправление сопровождается актом с объяснением причины невозможности доставки отправления. После возвращения отправления с актом в отдел обработки направляется запрос в офис, принялший отправление. Возврат или переадресация отправлений допускается только с согласия этого подразделения или по истечении 10-дневного срока. На адресной стороне делаются отметки «Досыпается в...» или «Возвращается в ...». На обратной стороне указывается причина. Отметки подчеркиваются цветным карандашом, подписываются начальником отделения и заверяются оттиском печати.

Письменные уведомления о переадресовке и возврате отправлений подшиваются с расходными документами. Такие отправления приписываются в конце реестра и упаковываются в общие постпакеты. Если постпакет не составляется, то возвращаемые или досылаемые пакеты, независимо от их количества, упаковываются в отдельные постпакеты. Возвращаемые пакеты и посылки приписываются непосредственно к накладной, в которой делается отметка «Возвращается» или «Досылается».

**Подготовка маршрутов.** Подготовкой маршрутов занимается отдел организации перевозок. К этому процессу относится следующая деятельность:

- планирование загрузки маршрутов;
- составление планов-заданий на каждый маршрут;
- подготовка сопроводительных документов.

Из центральной БД выводятся данные по всем отправлениям, полученным, обработанным и ожидающим отправления на маршрут. Из всего этого списка производится выборка по направлениям согласно планируемым маршрутам. Уже из этого списка сотрудниками отдела выполняется повторная выборка отправлений по различным кладовым согласно типу маршрута — формируется план-задание на маршрут. В процессе формирования плана-задания учитывается срочность отправления, контроль сроков доставки, возможности транспортного средства и другие факторы. План-задание содержит перечень отправлений, хранящихся в разных кладовых, которые курьер должен получить при отправке на маршрут. До тех пор пока отправления не выданы на маршрут, план-задание можно изменить в связи с изменившимися условиями.

Подобные задания, только в рамках своих отправлений, получает каждая кладовая, чтобы к моменту прибытия курьера подготовить отправления к маршруту.

В определенное время курьер отдела перевозок приходит в отдел организации перевозок, получает под роспись свое план-задание и выдвигается к кладовым для получения отправлений. Каждая кладовая, выдав отправления на маршрут, делает соответствующую отметку в плане-задании.

После выдачи отправлений на маршрут офисы, в которые направлены отправления, уведомляются о маршруте. Если произошли какие-то изменения, связанные с маршрутом (например, задержка), происходит повторное уведомление.

В Excel ведется учет фактических перевозок по факту получения документов от перевозчика.

**Вручение отправлений курьерам при отправке на маршрут.** Отправления доставляются курьерами. Все отправления заблаговременно подготавливаются (по возможности, упаковываются в пост-

пакеты) согласно плану направления. Перед отправкой на маршрут курьер получает инструктаж о порядке доставки отправлений (получает маршрутный лист), после чего направляется в отдел обработки для получения отправлений. Экспедитор проверяет все документы курьеров (предписания, служебные удостоверения, сопроводительный лист) и выдает им накладные и поименно приписанные к ним отправления.

Курьеры, принимая отправления:

- проверяют правильность оформления накладных;
- проверяют поименно полученные отправления в порядке их записи в накладных;
- осматривают внешнее состояние, целостность и правильность упаковки;
  - просчитывают количество принятых отправлений. Отправления в поврежденной упаковке, с неясными оттисками печатей, пломбами, поврежденными бумажными наклейками и перевязями принимать запрещается;
  - расписываются в копии накладной за принятые отправления, проставляют дату, время и заверяют оттиском печати (если более одного курьера, то в приемке расписываются все);
  - укладывают принятые отправления в мешки (контейнеры) с обязательным просчетом;
  - опечатывают их, заполняют сопроводительный лист о количестве мест, подлежащих погрузке в транспортное средство, и докладывают ответственному лицу о готовности следовать маршруту;
  - укладывают мешки в транспортное средство с просчетом количества мест. Пересчет отправлений обязателен при каждой перегрузке отправлений.

Если курьер по не зависящим от него причинам не был отправлен на маршрут, он докладывает об этом ответственному лицу и сдает принятые отправления обратно.

Экспедитор принимает отправления, расписывается на подлиннике накладной, указывая количества принятых отправлений (прописью), а на копии накладной в присутствии курьера гасит его расписку путем перечеркивания и делает отметку с проставлением даты, времени, росписи и оттиска штампа.

Для оказания помощи и охраны отправлений при погрузке в транспортное средство в подразделениях выделяется посадочная бригада из числа дежурных курьеров и, при необходимости, специальные средства. Старшему посадочной бригады передается заполненный курьерами сопроводительный лист. По окончании погрузки старший курьер делает отметку в сопроводительном листе, подтверждающую прием всех мест в транспортное средство, и возвращает его посадочной бригаде. Сопроводительные листы хранятся в сброшюрованном виде установленным порядком в архиве.

**Перевозка отправлений.** Перевозка отправлений осуществляется по автомобильным маршрутам, которые включены в единую сеть маршрутов.

Перед началом работы курьер получает в отделе обработки отправления для доставки и все необходимые документы, а также контрольный лист с указанием организаций, в которых необходимо осуществить сбор отправлений. После этого курьер приступает к выполнению маршрута по утвержденному маршрутному листу.

По прибытии в организацию курьер, в зависимости от поставленных задач, производит сбор и (или) выдачу отправлений.

После выполнения маршрута курьер:

- проверяет по реестрам правильность вручения пакетов адресатам;
- составляет справки на неврученные отправления;
- докладывает о выполнении задания;
- представляет документы должностному лицу на проверку;
- после проверки сдает экспедитору реестры и неврученные пакеты со справками.

Экспедитор, приняв реестры и неврученные отправления:

- проверяет, все ли листы реестров сданы;
- исключает из реестров возвращенные пакеты, заверяет подписью и оттиском штампа;
- передает неврученные пакеты на сортировку по передаточной ведомости;
- делает отметку в маршрутном листе о количестве возвращенных листов реестра (цифрами) и отправлений (прописью).

Маршрутные листы брошюруются в отдельную папку и хранятся в течение шести месяцев.

Работа курьеров по обслуживанию маршрута длительностью более 8 ч организуется посменно. При этом назначается дежурный курьер, который обязан находиться на рабочем месте, быть одетым по форме. Передача отправлений между дежурными курьерами производится по передаточной ведомости.

Отправления в поврежденной оболочке или с неясными, поврежденными печатями, пломбами и перевязями от курьера-обменщика не принимаются и возвращаются обратно. Маршрутный курьер принимает оставшиеся отправления в установленном порядке, а в копии накладной (под копировку на подлинник) делает отметку о причинах возврата отправлений, подписывает и заверяет штампом. Непринятые отправления вычеркиваются курьером. Данные отправления курьеры-обменщики сдают обратно в офис с составлением справки под расписку в копии накладной.

Маршрутные курьеры в пути следования могут вскрывать постпакеты при получении об этом запроса от офиса их отправки. После вскрытия постпакета и проверки вложения курьеры, при необходимости

мости, изымают из него указанное в запросе отправление. Производится запись в реестре. Запись заверяется подписями курьеров и штампом. Зачеркивать запись и производить исправление итога в реестре запрещается.

После вскрытия постпакета и проверки вложения маршрутные курьеры вновь упаковывают его в постпакет, опечатывают своей печатью (пломбой) и составляют акт в двух экземплярах, из которых первый прикладывают к постпакету, а второй, как приходный документ, к накладной, полученной от офиса, сдавшего этот постпакет. Изъятое отправление маршрутные курьеры отправляют по назначению с припиской к соответствующей накладной.

Офису, от которого поступил запрос с просьбой о проверке вложения постпакета, курьеры сообщают о результатах проверки. Копия запроса подшивается с актом.

Перед прибытием в конечный пункт маршрута курьеры:

- совместно со встречающей бригадой производят выгрузку отправлений;
- перед выходом из транспортного средства осматривают рабочее место, хранилище, кладовую, салон и проверяют, не остались ли там отправления;
- сдают в офисе все отправления по накладным;
- сдают под расписку на хранение дежурному по подразделению снаряжение, печати, проездные и другие документы, кроме удостоверений личности и предписаний;
- в предписании делают необходимые отметки;
- докладывают руководителю подразделения, а в нерабочее время — дежурному по подразделению о выполнении маршрута, а также обо всех происшествиях в пути (необменах, нарушениях, допущенных курьерами-обменщиками и т. п.), сдают для проверки производственные документы и получают указание о времени явки на работу.

После встречи маршрута, доставки и сдачи отправлений старший посадочной бригады докладывает дежурному о результатах и передает ему подписанный сопроводительный лист.

Перед отправкой из конечного пункта маршрутные курьеры:

- в установленное время прибывают в офис для инструктажа, получения снаряжения, проездных и производственных документов;
- принимают отправления и доставляют их в транспортное средство.

По возвращении в начальный пункт сдают отправления, документы и докладывают о выполнении маршрута.

**Прием отправлений от курьеров после возвращения.** Прием отправлений от маршрутных курьеров производится в офисах. После выполнения маршрута и сбора всех отправлений, по прибытии в офис, курьер сдает принятые отправления вместе с первым эк-

записью реестра под расписку в контрольном листе экспедитору. Контрольный лист после сдачи отправлений и получения расписки в их приеме возвращается под расписку в журнале выдавшему его работнику для проверки и последующей передачи для подшивки к производственным документам.

При этом обязательно:

- проверяется правильность оформления накладной;
- просчитывается и сверяется с документами количество принимаемых отправлений;
- проверяется правильность оформления;
- проверяется правильность адресования;
- указывается в накладной прописью количество принятых отправлений, дата, время приема, ставится подпись, которая заверяется штампом;
- в случае предъявления постпакетов с нарушенной оболочкой они вскрываются в присутствии сдающих курьеров, составляется акт.

Затем, в присутствии второго экспедитора, вскрываются постпакеты, мешки. Сверяется номер реестра с номером, указанным на адресном ярлыке постпакета. Сверяется с фактом количество приписанных к реестру отправлений, и поименно проверяются отправления. Делается запись в реестре о количестве отправлений. При этом расписываются оба работника. Проверяются оболочки постпакетов. После чего отправления по передаточным ведомостям отдаются для дальнейшей обработки.

Оболочки от вскрытых пакетов и адресные ярлыки хранятся на рабочих местах и уничтожаются не ранее чем через сутки после составления ведомости. Мешки по истечении указанного срока проверяются и используются в дальнейшем.

Транзитные мешки выборочно проходят контрольное взвешивание.

*Вручение отправлений курьерами.* Отправления, направленные адресатам, как правило, доставляются курьерами. По прибытии в организацию:

- у принимающего должностного лица уточняется, для какого учреждения принимаются отправления;
- проверяется наличие доверенности на их получение у ответственных лиц, если принимает не адресат лично;
- каждое отправление выдается получателю в отдельности, после сличения данных с записью в реестре, при этом вслух прочитывается адресование;
- получателю предлагается пересчитать отправления, сверить их адресования с наименованием организации, куда они сдаются, вручить реестр, в котором получатель расписывается за их получение с расшифровкой подписи, указанием количества отправлений

прописью, даты, времени, заверенными оттиском печати получателя. При отсутствии печати корреспонденция может быть сдана лицу, ответственному за ее прием по предъявлении им паспорта или другого документа, удостоверяющего личность. В этом случае в реестре дополнительно указывается номер этого документа, кем и когда он выдан, а также телефон получателя. Правильность подтверждается подписью получателя;

- курьер получает обратно реестр, проверяет, нет ли расхождений в наименовании организации, указанного в реестре, с наименованием, указанным на оттиске печати, проверяет, за все ли отправления имеется расписка;

- расписывается в журнале (при его наличии) за количество врученных отправлений, осматривает место сдачи и следует дальше по маршруту.

Вручение отправлений в офисах производится, если адресат находится за пределами районного центра или если по условиям договора получатель обязуется сам забирать отправления.

В этом случае после поступления отправления в подразделение получателю высыпается извещение (или сообщается по телефону, телеграфу, электронной почтой) о его получении с предложением прибыть в офис компании лично или направить своих представителей с доверенностью. Об этом делается отметка в соответствующем журнале.

**Контролирующая деятельность.** Контролирующая деятельность компании представлена процессом мониторинга. Мониторинг — систематический или непрерывный сбор, обработка и анализ информации о контролируемых объектах и их параметрах, которая может быть использована для обеспечения деятельности соответствующих служб ООО «Экспресс-доставка», поддержки процесса принятия решения руководством и информирования клиентов.

В процессе мониторинга объектов решаются следующие основные задачи:

- определение местонахождения ( поиск ) объекта;
- контроль параметров и состояния ( статуса ) объекта;
- выявление соответствия параметров контролируемого объекта установленным правилам, ограничениям и обязательствам.

Интерес представляют направления:

- мониторинг отправлений;
- мониторинг транспорта;
- мониторинг курьеров.

**Мониторинг отправлений.** Мониторинг отправлений осуществляется специалистами управления логистики. Процессы мониторинга сводятся к мероприятиям по контролю за соблюдением контрольных сроков нахождения отправлений на каждом из этапов их обработки и доставки, поиску по запросу отправлений в основном

хранилище данных компании. Как правило, мероприятия по мониторингу отправлений осуществляются либо по условиям договора, либо по требованию клиента, либо по факту недоставки отправления в срок.

Для контроля отправлений широко применяется система нумерации всех отправлений, при этом технологии штрихового кодирования, которые позволяют ускорить процесс идентификации отправлений, внедрены не повсеместно, сканеры штрих-кодов используются только в некоторых филиалах.

**Мониторинг транспорта.** Мониторинг транспорта осуществляется сотрудниками автобазы.

На сегодняшний день система мониторинга транспортных средств позволяет:

- получать по номеру автомобиля информацию о его местонахождении и о его состоянии (двигается, стоит на месте);
- связываться с экипажем автомобиля;
- прослушивать его салон.

**Мониторинг курьеров.** На сегодняшний день в ООО «Экспресс-доставка» мониторинг курьеров осуществляется путем контрольных звонков на их мобильные телефоны («прозвон»). Каких-либо средств автоматизации данного вида мониторинга не применяется.

В соответствии с должностной инструкцией курьер по прибытии в организацию докладывает по телефону о факте прибытия, причинах опоздания (если таковое имело место быть). Позже он сообщает о факте приема отправления и о причинах задержки приема (если она была).

Для контроля курьеров косвенно используется система мониторинга транспортных средств, которая отслеживает местоположение автомобиля, в котором находится курьер. Однако средства мониторинга не позволяют осуществлять мониторинг курьера, если он вышел из транспортного средства.

### 8.1.2. Описание ИТ-инфраструктуры

Под ИТ-инфраструктурой будем понимать аппаратное, прикладное и системное ПО, сетевое оборудование, инженерное и вспомогательное оборудование. В это понятие не входят: задействованный персонал, данные бизнеса, бизнес-процессы и связанная с ними документация. ИТ-инфраструктура предприятия должна позволять реализовать работу всех бизнес-приложений с требуемыми параметрами непрерывности, надежности, нагрузки и пр.

Для автоматизации бизнес-процессов в ООО «Экспресс-доставка» используется достаточно широкий набор программных средств, часто не связанных друг с другом. Перечень этих программ включает в себя как собственные разработки прикладных программ, так и купленное ПО.

## **Программное обеспечение**

**Общесистемное ПО.** В компании закуплены АРМ под управлением Windows XP. На некоторых компьютерах руководителей филиалов установлена Windows 7. Часть руководства центрального офиса пользуется компьютерами под управлением iOS. Базы данных ведутся локально в разных офисах под управлением СУБД Access.

На всех компьютерах установлен пакет Microsoft Office версий 2007, 2010.

В компании развернута почтовая инфраструктура на технологиях Microsoft Exchange.

**Прикладное ПО.** Два года назад в компании стартовали работы по внедрению ИС 1С: Предприятие. Эта информационная система используется для ведения бухгалтерского учета в компании. В двух филиалах были проведены дополнительные работы по интеграции с программным блоком управления «Оптовая доставка автотранспортом» для 1С. Работы по комплексному внедрению этой ИС до сих пор не завершены. В большинстве филиалов поставлены модули «Основные средства» и «Бухгалтерия». В двух филиалах система поставлена только в базовой конфигурации, доработка не произведена. Наблюдаются проблемы с качеством обучения персонала.

В настоящее время учет в ООО «Экспресс-доставка» не консолидирован, ведется в каждом филиале самостоятельно. В компании также используется картографическое ПО, ИС «Картография России», которое автономно применяется для разработки маршрутов, их отслеживания и анализа.

**Система мониторинга.** В ООО «Экспресс-доставка» работает частично функционирующая система мониторинга. Она имеет ряд ограничений. Так, используемые программные и технические средства системы мониторинга транспорта ООО «Экспресс-доставка» позволяют осуществлять контроль только автомобильного транспорта. При этом сегодня в разных регионах страны используются информационно не связанные серверы мониторинга. Вследствие этого в компании нет общего единого массива данных мониторинга, что препятствует получению агрегированных отчетных и аналитических данных.

Такое разделение обусловлено разнообразием аппаратных средств мониторинга, установленных на транспортных средствах. На каждом из серверов данные собираются по разным протоколам и обрабатываются разными алгоритмами, что практически исключает возможность корректного совместного использования этих данных в системе мониторинга отправлений.

На сегодняшний день в ООО «Экспресс-доставка» отсутствует фиксируемая техническими средствами информация о местонахождении курьера на маршруте. Информация о местоположении курье-

ра получается посредством телефонного звонка, но ничем не подтверждается.

Таким образом, можно сделать вывод о том, что на сегодняшний день в компании введена автоматизация отдельных функций, а не всех бизнес-процессов в совокупности.

*Аппаратное обеспечение.* Аппаратная платформа представлена серверами и сетевым оборудованием.

Сейчас в компании куплены и используются несколько серверов:

- для поддержания почтовой инфраструктуры;
- хранения данных;
- выделенный сервер для 1С;
- серверы действующей системы мониторинга.

АРМ пользователей представлены компьютерами российской сборки. АРМ внутри каждого филиала объединены в локальную сеть.

Сообщение с другими офисами осуществляется по сети Интернет.

*Состояние телекоммуникационной инфраструктуры.* ООО «Экспресс-доставка» не имеет ведомственной телекоммуникационной сети. В целях коммуникации используются открытые и закрытые каналы связи. Для этого используются телефонные кабельные каналы связи национальных операторов связи, каналы сотовой связи операторов мобильной связи ОАО «Мобильные ТелеСистемы» и ОАО «МегаФон», внутренние локальные сети и средства связи. Телефонные кабельные каналы используются для телефонной и факсимильной связи, а также для передачи данных, в том числе между офисами (посредством Интернета).

Каналы сотовой связи используются для передачи сведений о местоположении транспортных средств от бортовых терминалов на серверы ООО «Экспресс-доставка». С помощью этих каналов осуществляется связь (двусторонняя или односторонняя) с транспортным средством, а также обеспечивается корпоративная голосовая телефонная связь и передача SMS между сотрудниками ООО «Экспресс-доставка».

### 8.1.3. Отзывы о компании на сайте

**Отзыв 1.** Доставка произведена вовремя, спасибо!

**Отзыв 2.** Проблемы с доставкой!!! Всего-то нужно было получить документы, а затем после их подписания сразу отправить, но эти документы так до меня и не дошли. При отправке обещали, что все придет после 26-го, но ни 26-го, ни в следующие два дня они так и не были доставлены. Хуже всего то, что менеджер в ООО «Экспресс-доставка» отказался предоставить статус о местонахождении отправления, т. к. они не обладают этой информацией!!! По его словам, проблема в том, что когда в рамках заявки один и тот же адре-

сат сначала является получателем, а затем отправителем, то они не заводят эти заявки в систему!!!

**Отзыв 3.** Отправлял очень ценный заказ из Саратова в Сергиев Посад, но он все еще не пришел заказчику, хотя прошло 3 недели!! В компании сказали, что не могут сказать статус доставки, т. к. пункт отправления в одном регионе, а получения — в Московской области, и у них нет единой базы...

**Отзыв 4.** Проблемы с доставкой!!! Всего-то нужно было получить документы, а затем после их подписания сразу

**Отзыв 5.** Неплохой сервис. Единственная просьба — точнее указывать, когда ждать курьера. Почти у всех компаний есть временные интервалы, например с 9 до 12 и пр, а ваши курьеры звонят, когда уже на месте.

**Отзыв 6.** Присоединяюсь к предыдущим отзывам. Курьеры приезжают иногда ранним утром, иногда вечером, абсолютно непредсказуемо. Не знаю, как составляются маршруты, но компании с этим нужно что-то делать...

**Отзыв 7.** Было бы здорово при заказе получать номер для отслеживания на сайте. Если будет такая возможность, буду обращаться только к вам. Пока DHL/Fedex в этом отношении явно выигрывают (пусть даже у вас и дешевле)...

**Отзыв 8.** Наша компания работает уже не первый месяц с ООО «Экспресс-доставка», но сейчас впервые рискнули на свой страх и риск воспользоваться услугой сбора грузов из нескольких наших офисов и их доставки заказчику. В результате пришлось платить неустойку! Никакого мониторинга статуса, возможности узнать, где груз, т. к. так называемые «заявки на сбор от нескольких отправителей» вы не отслеживаете!!! Да они же наиболее важные и ценные! Ваш менеджер связался со мной по поводу компенсации, надеюсь вопрос будет разрешен.

## 8.2. Контрольное задание

### 8.2.1. Постановка Задачи

Цель данного учебного проекта — проведение анализа компании ООО «Экспресс-доставка» (деятельность которого описана выше) для выявления проблемных мест функционирования бизнеса и возможных путей их автоматизации.

Для достижения обозначенной цели потребуется решить следующие задачи:

- составить план проекта создания ИС для автоматизации проблемных мест ООО «Экспресс-доставка»;
- построить модели бизнес-процессов, функциональные и информационные модели компании ООО «Экспресс-доставка»;

- сформировать требования к информационной системе;
- проанализировать возможность использования готового программного обеспечения, представленного на рынке;
- провести объектно-ориентированный анализ и проектирование ПО;
- составить план тестирования ПО;
- произвести планирование развертывания и внедрения созданной ИС в промышленную эксплуатацию компанией ООО «Экспресс-доставка».

**Этапы проекта.** Работа над проектом состоит из ряда этапов. Требуется:

- 1) сформировать проектную команду с выбором руководителя проекта; проект может выполняться как одним студентом, так и группой (не более трех человек);
- 2) разработать Устав проекта;
- 3) разработать Реестр рисков и Форму регистрации рисков;
- 4) разработать План проекта, который включает:
  - Устав проекта,
  - Реестр рисков проекта,
  - иерархическую структуру работ,
  - ресурсы проекта,
  - роли и обязанности участников проекта,
  - управление календарем проекта,
  - оценку затрат на проект;
- 5) провести структурный анализ компании ООО «Экспресс-доставка», включающий:
  - моделирование бизнес-процессов компании ООО «Экспресс-доставка» с использованием нотации BPMN или EPC,
  - функциональный анализ компании ООО «Экспресс-доставка» с использованием нотации IDEF0 и анализ информационных потоков с использованием нотации DFD,
  - анализ информационной модели компании на основе диаграммы «сущность-связь» (ERD),
    - определение границ автоматизации;
- 6) разработать требования к ИС (на основе классификации FURPS+);
- 7) выбрать готовое программное решение из ПО, представленного на рынке;
- 8) провести объектно-ориентированный анализ и проектирование, включающие построение:
  - диаграмм прецедентов,
  - диаграммы классов,
  - диаграмм деятельности,
  - диаграмм последовательности,
  - диаграмм кооперации,

- диаграмм состояний,
  - диаграмм компонентов,
  - диаграмм развертывания;
- 9) сформировать План тестирования, который включает:
- этапы тестирования,
  - требования для тестирования,
  - стратегию тестирования,
  - ресурсы тестирования,
  - документацию для тестирования;
- 10) осуществить планирование развертывания и внедрения информационной системы в компании ООО «Экспресс-доставка».

### **8.2.2. Планирование проекта**

**Устав проекта.** Проектные работы часто начинаются с создания Устава проекта. Устав проекта — это специальный документ, который с формальной точки зрения начинает проект и описывает деятельность Заказчика и Исполнителя.

Обычно Устав проекта является внешним по отношению к проекту и отражает его видение со стороны Заказчика, хотя работу над Уставом ведут обе стороны (причем Исполнитель чаще прикладывает больше усилий).

Устав проекта формально издается Куратором (спонсором) проекта, который находится на достаточно высоком уровне, чтобы осуществлять контроль и управление ресурсами. В Уставе проекта определяются Руководители проекта (с обеих сторон), Кураторы проекта (с обеих сторон), определяются команды проекта. Устав содержит также описания основных стадий проекта.

Пример Устава проекта для рассматриваемого кейса приведен в подп. 8.2.9 и Приложении 1.

**Реестр рисков проекта.** Управление рисками — важная часть любого проекта. Управление рисками начинается в момент формального запуска проекта и продолжается вплоть до последних стадий.

Управление рисками может осуществляться в разных форматах, но часто с этой целью создают специальный документ, содержащий так называемый Реестр рисков. Форма Реестра рисков — это таблица, которая включает в себя определение риска, оценку риска, вероятность реализации риска и стратегию минимизации риска. Как правило, Реестр содержит достаточно большое число рисков, и управление осуществляется по видам риска.

Ведением Реестра рисков обычно занимается ответственный за это человек либо непосредственный руководитель проекта. При этом инициатором добавления риска может стать любой член проектной команды.

Пример Реестра рисков приведен в подп. 8.2.9 и Приложении 2. Обратите внимание, что Реестр рисков обновляется по ходу выпол-

нения проекта. Для этого используется специальная Форма регистрации риска. Пример формы регистрации риска приведен в подп. 8.2.9 и Приложении 3.

**Иерархическая структура работ.** Работа над сложными и масштабными проектами не может быть эффективной без четкого понимания последовательности и сроков выполнения работ. Поэтому Руководитель проекта занимается построением ИСР.

Для построения ИСР потребуется программное решение Microsoft Project, входящее в MS Office. В данном случае можно использовать любую версию MS Project (рис. 8.3).

	Реж. зад.	Название задачи	Длительнс	Начало	Окончание
1		- Формирование проектной команды	5 дней	Пн 11.01.16	Пт 15.01.16
2		Распределение ролей и обязанностей	1 день	Пн 11.01.16	Пн 11.01.16
3		Формирование Устава проекта	1 день	Вт 12.01.16	Вт 12.01.16
4		Формирование Реестра рисков	1 день	Ср 13.01.16	Ср 13.01.16
5		Формирование технико-экономического обоснования	1 день	Чт 14.01.16	Чт 14.01.16
6		Этап формирования проектной команды завершен	0 дней	Чт 14.01.16	Чт 14.01.16
7		- Проведение структурного анализа	39 дней	Пт 15.01.16	Вт 15.03.16
8		Построение карты процессов	3 дней	Пт 15.01.16	Вт 19.01.16
9		Моделирование бизнес-процессов	11 дней	Ср 20.01.16	Ср 03.02.16
10		Функциональный анализ	14 дней	Пт 15.01.16	Ср 03.02.16
11		Построение и анализ информационной модели	10 дней	Чт 04.02.16	Ср 17.02.16
12		Определение границ автоматизации	15 дней	Чт 18.02.16	Вт 15.03.16
13		Проведение структурного анализа завершено	0 дней	Вт 15.03.16	Вт 15.03.16
14		Формулирование требований к ИС	7 дней	Ср 16.03.16	Чт 24.03.16
15		Формирование требований к ИС завершено	0 дней	Чт 24.03.16	Чт 24.03.16

Рис. 8.3. Пример иерархической структуры работ, сформированной в MS Project 2010 (фрагмент)

Источником информации о наборе работ и их порядке будет служить тема 3 и раздел «Введение» данного задания, а также Приложение 3, «Описание потоков работ RUP».

В рамках данного кейса будем исходить из того, что проект начнется 11 января 2016 г. и закончится 30 августа 2016 г. Промежуточные даты должен определить руководитель проекта на основании своего опыта, требований заказчика и различных методов оценки длительности работ.

Построение ИСР в MS Project обычно не вызывает затруднений. В первой строке обычно указывают название первого этапа. В рамках данного кейса первым этапом будет «Формирование проектной команды». В столбце «Режим задачи» здесь и далее выбирают «Автоматическое планирование».

Далее будут указаны конкретные работы, например «Получение методических указаний». Для каждой из них нужно указать значение длительности в соответствующем столбце. После этого значения в столбцах «Начало» и «Окончание» будут сгенерированы автоматически. В ИСР нужно отразить все работы, которые будут включены в проект.

Обратите внимание, что приведенная на рис. 8.3 иерархическая структура работ включает так называемые *вехи проекта*. Вехи проекта обозначают события, которые свидетельствуют о завершении важной для проекта группы работ. Сделать работу вехой в MS Project очень просто: в столбце «Длительность» в соответствующей строке достаточно указать значение «0 дней».

MS Project позволяет визуализировать ИСР в виде диаграммы Ганта. Для этого нужно определить порядок выполнения работ. Особое внимание нужно обратить на работы, начало которых требует завершения работ-предшественниц. К примеру, функциональное тестирование не может начаться, пока не будет проведено модульное тестирование. И наоборот, Разработка документации и Создание установочных пакетов могут происходить параллельно.

Создав ИСР и установив связи между работами в MS Project, можно переходить к управлению ресурсами проекта.

**Ресурсы проекта.** Для управления ресурсами MS Project содержит справа внизу специальную вкладку под названием «Лист ресурсов». В этой вкладке нужно указать ресурсы (включая трудовые и материальные), которые будут использоваться в проекте.

Для формирования списка трудовых ресурсов следует обратиться к Приложению 3 «Описание потоков работ RUP» (см. все таблицы «Роли»). Важно понимать, что трудовые ресурсы могут варьироваться от проекта к проекту, поэтому не обязательно использовать все роли, приведенные в Приложении 3. Допускается самостоятельный выбор ролей (в том числе из других источников).

Используемые трудовые ресурсы необходимо внести в Лист ресурсов (рис. 8.4). Для каждого ресурса можно указать:

- название;
- тип:
  - трудовой,
  - материальный,
  - затраты;
- максимум единиц;
- стандартная ставка;

- затраты на использование;
- начисление:
  - в начале,
  - пропорционально,
  - по окончании;
- базовый календарь.

1	Название ресурса	Тип	Макс. единиц	Стандартная ставка	Начисление	Базовый календарь
1	Руководитель проекта	Трудовой	100%	3 200,00р./час	Пропорциональное	Стандартный
2	Бизнес-аналитик	Трудовой	200%	1 000,00р./час	Пропорциональное	Стандартный
3	Системный аналитик	Трудовой	200%	1 300,00р./час	Пропорциональное	Стандартный
4	Системный архитектор	Трудовой	100%	1 600,00р./час	Пропорциональное	Стандартный
5	Проектировщик БД	Трудовой	100%	1 200,00р./час	Пропорциональное	Стандартный
6	Программист	Трудовой	300%	1 300,00р./час	Пропорциональное	Стандартный
7	Тестировщик	Трудовой	100%	1 000,00р./час	Пропорциональное	Стандартный
8	Технический писатель	Трудовой	100%	900,00р./час	Пропорциональное	Стандартный
9	Системный администратор	Трудовой	100%	900,00р./час	Пропорциональное	Стандартный
10	Специалист по сопровождению	Трудовой	100%	1 000,00р./час	Пропорциональное	Стандартный
11	Специалист по обучению	Трудовой	100%	1 200,00р./час	Пропорциональное	Стандартный
12						
13	Серверы	Затраты			В начале	
14	Планшеты	Затраты			В начале	
15	Рабочие места	Затраты			В начале	
16	Сетевое оборудование	Затраты			В начале	

Рис. 8.4. Пример листа ресурсов, сформированного в MS Project 2010

На отдельных столбцах следует остановиться подробнее. Так, в столбце «Тип» для всех сотрудников указывается тип «Трудовой». Краткое название указывают для удобства, и оно должно в удобной форме отражать полное название ресурса. Если в рамках проекта ожидаются затраты (к примеру, закупка планшетов или рабочих мест), то для них следует выбрать тип «Затраты». Указание фактического размера затрат в листе ресурсов не производится и будет подробно рассмотрено далее.

Столбец «Максимум единиц» показывает максимальное число единиц того или иного ресурса, доступное для проекта. Значение указывается в единицах или в процентах. К примеру, значение 400 % для трудового ресурса «Тестировщик» будет означать, что в проекте принимают участие четыре тестировщика.

Столбец «Стандартная ставка» отражает заработную плату трудового ресурса. Значения в этом столбце следует указать на основе текущей ситуации на рынке труда.

Столбец «Затраты на использование» показывает соответствующие затраты. В нем отражаются издержки и затраты на эксплуатацию материальных ресурсов. В рамках данного кейса их можно не указывать, поскольку большинство материальных ресурсов представлено в форме затрат или принадлежат Заказчику.

Для трудовых ресурсов в столбце «Начисление» указывается значение «Пропорциональное», если они будут получать оплату по ходу

выполнения проекта. Если трудовые ресурсы получают оплату после выполнения каких-либо работ, то необходимо указать тип «По окончании». В случае предоплаты ставится значение «В начале».

И, наконец, требуется указать базовый календарь ресурса. Как правило, трудовые ресурсы работают по стандартному календарю (или в ночную смену). Для материальных ресурсов может быть указано значение «24 часа».

**Роли и обязанности участников проекта.** Для выполнения следующего шага потребуется связать иерархическую структуру работ с ресурсами проекта. Как и раньше, потребуется применение программного средства MS Project.

Необходимо открыть вкладку «Диаграмма Ганта» и обратиться к столбцу «Названия ресурсов». Для каждой работы в появившемся окне нужно указать используемые ресурсы (рис. 8.5).

Если для ИСР был выбран автоматический режим, то столбец «Затраты» будет заполнен автоматически.

№	Реж. зад?	Название задачи	Длительнк	Затраты	Начало	Окончание	Названия ресурсов
1		- Формирование проектной команды	5 дней	102 400,00р.	Пн 11.01.16	Пт 15.01.16	
2		Распределение ролей и обязанностей	1 день	25 600,00р.	Пн 11.01.16	Пн 11.01.16	Руководитель проекта
3		Формирование Устава проекта	1 день	25 600,00р.	Вт 12.01.16	Вт 12.01.16	Руководитель проекта
4		Формирование Реестра рисков	1 день	25 600,00р.	Ср 13.01.16	Ср 13.01.16	Руководитель проекта
5		Формирование технико-экономического обоснования	1 день	25 600,00р.	Чт 14.01.16	Чт 14.01.16	Руководитель проекта
6		Этап формирования проектной команды завершен	0 дней	0,00р.	Чт 14.01.16	Чт 14.01.16	
7		- Проведение структурного анализа	39 дней	652 800,00р.	Пт 15.01.16	Вт 15.03.16	
8		Построение карты процессов	3 дней	100 800,00р.	Пт 15.01.16	Вт 19.01.16	Руководитель проекта; Бизнес-аналитик
9		Моделирование бизнес-процессов	11 дней	88 000,00р.	Ср 20.01.16	Ср 03.02.16	Бизнес-аналитик
10		Функциональный анализ	14 дней	112 000,00р.	Пт 15.01.16	Ср 03.02.16	Бизнес-аналитик
11		Построение и анализ информационной модели	10 дней	232 000,00р.	Чт 04.02.16	Ср 17.02.16	Системный аналитик; Системный
12		Определение границ автоматизации	15 дней	120 000,00р.	Чт 18.02.16	Вт 15.03.16	Бизнес-аналитик
13		Проведение структурного анализа завершено	0 дней	0,00р.	Вт 15.03.16	Вт 15.03.16	
14		Формулирование требований к ИС	7 дней	308 000,00р.	Ср 16.03.16	Чт 24.03.16	Системный аналитик; Бизнес-аналитик; проекта
15		Формирование требований к ИС завершено	0 дней	0,00р.	Чт 24.03.16	Чт 24.03.16	

*Рис. 8.5. Пример иерархической структуры работ (с указанными ресурсами и рассчитанными затратами), сформированной в MS Project 2010 (фрагмент)*

MS Project автоматически визуализирует ИСР в формате диаграммы Ганта (рис. 8.6).

Обратите внимание, что указать стоимость ресурсов с типом «Затраты» можно только теперь. После того как ресурс с типом «Затраты» назначен для задачи, нужно открыть окно «Сведения о задаче» и на вкладке «Ресурсы» установить нужные значения (рис. 8.7).

Назначенные ресурсы автоматически отобразятся на диаграмме Ганта. Если лимит по ресурсам превышен, то соответствующие работы и ресурсы будут выделены красным цветом. Это будет озна-

чать, что требуется либо изменить последовательность работ, либо увеличить количество используемого ресурса.

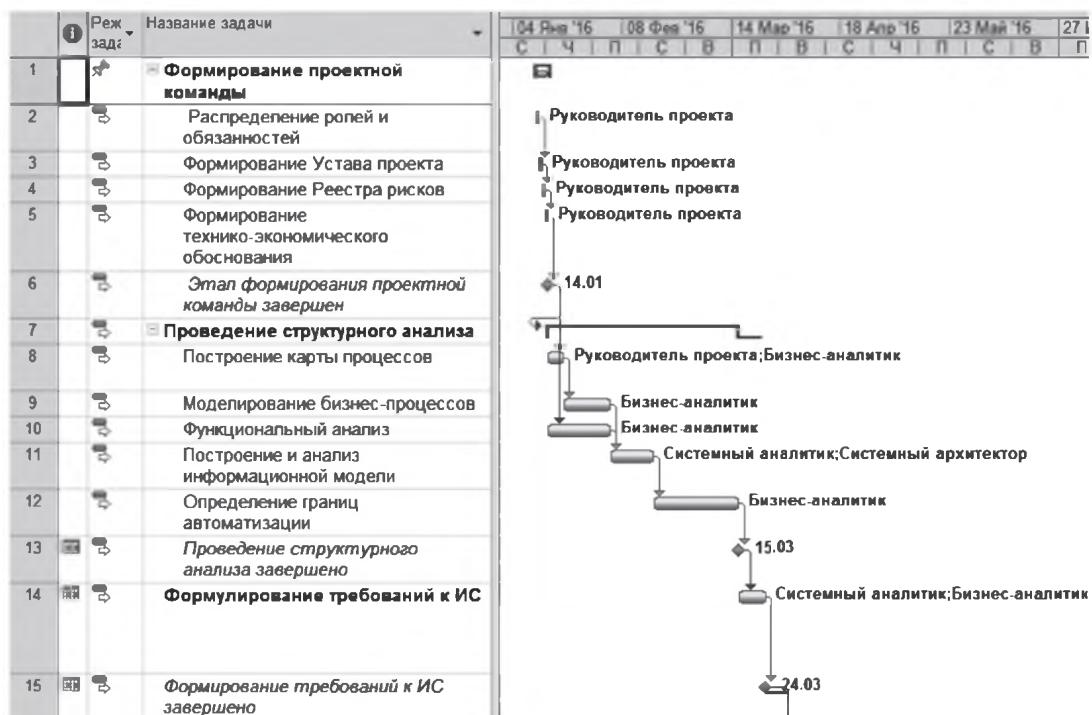


Рис. 8.6. Фрагмент диаграммы Ганта

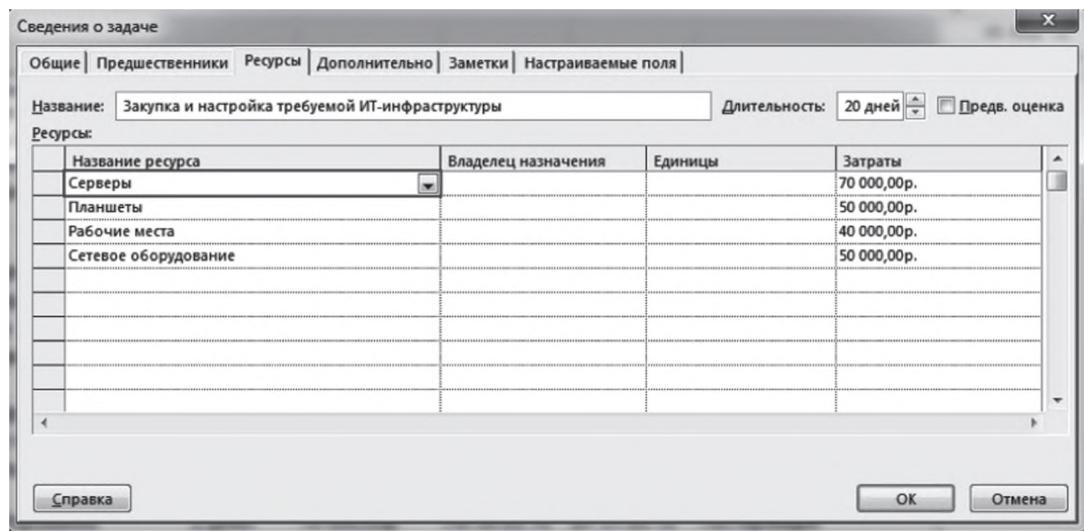


Рис. 8.7. Указание значения затрат для задачи «Закупка и настройка требуемой ИТ-инфраструктуры»

**Календарный план проекта.** Для формирования календарного плана проекта в MS Project нужно выбрать вверху вкладку «Проект», а затем нажать на кнопку «Изменить рабочее время».

В появившемся окне (рис. 8.8.) можно настроить Стандартный календарь проекта (для всех трудовых ресурсов). По умолчанию стандартный календарь использует рабочую неделю 5/2 и время

работы с 8:00 до 17:00. Нужно изменить рабочий день так, чтобы он начинался в 9:00 и заканчивался в 18:00. Это можно сделать на вкладке «Параметры».

Также требуется рассмотреть область «Исключения». MS Project не учитывает официальные праздничные дни, поэтому их нужно добавить вручную. Для каждого праздника понадобится добавить новое исключение.

Обратите внимание, что в MS Project можно завести индивидуальный календарь для каждого участника проекта. Это нужно для учета отпусков и доступности сотрудников. В данном проекте можно не добавлять индивидуальные календари.

**Оценка затрат на проект.** Оценка затрат на проект может выполняться в форме отдельного документа, называемого «Технико-экономическое обоснование».

В случае привлечения внешнего исполнителя такой документ предоставляется заказчику исполнителем. Он содержит все прямые расходы на проект — трудозатраты, материальные расходы, а также расходы по внешним подрядчикам. Этот документ будет также содержать все налоги, рассчитанные накладные расходы, норму прибыли по согласованным нормативам.

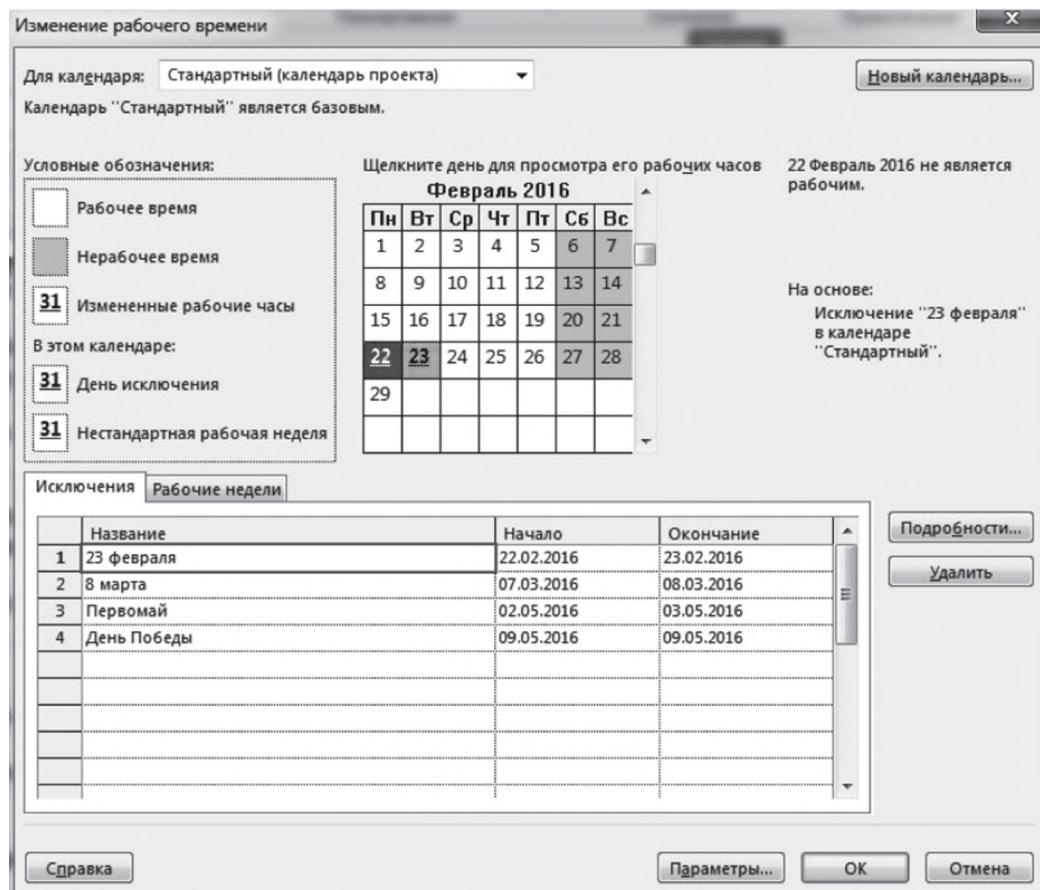


Рис. 8.8. Пример стандартного календаря проекта в MS Project

В случае внутреннего выполнения работ обычно составляется оценка трудозатрат по проекту и ожидаемых материальных расходов. В п. 7 Устава проекта приведена оценка затрат на проект, выполняемый внешним исполнителем. Этот расчет произведен при варианте разработки программного обеспечения по требованиям, сформированным на этапе структурного анализа. Расчет оплаты работ по этапам произведен из расчета трудозатрат с учетом ставки работы участников проекта, которая включает налоги, накладные расходы, норму прибыли и пр.

### 8.2.3. Структурный анализ деятельности компании

**Моделирование бизнес-процессов.** Очень часто моделирование бизнес-процессов начинается с построения карты процессов. На карте процессов отражаются бизнес-процессы верхнего уровня, которые детализируются средствами моделирования бизнес-процессов компании. В результате формируется набор диаграмм, которые отражают карту бизнес-процессов компании.

Рассмотрим построение карты бизнес-процессов в нотации EPC. Описание элементов графической нотации приведено в подп. 4.4.2.

Для построения карты процессов воспользуется программным средством ARIS Express и перечнем бизнес-процессов компании «Экспресс-доставка» (подпараграф 8.1.1). Карта процессов компании должна включать все бизнес-процессы, отраженные в описании кейса.

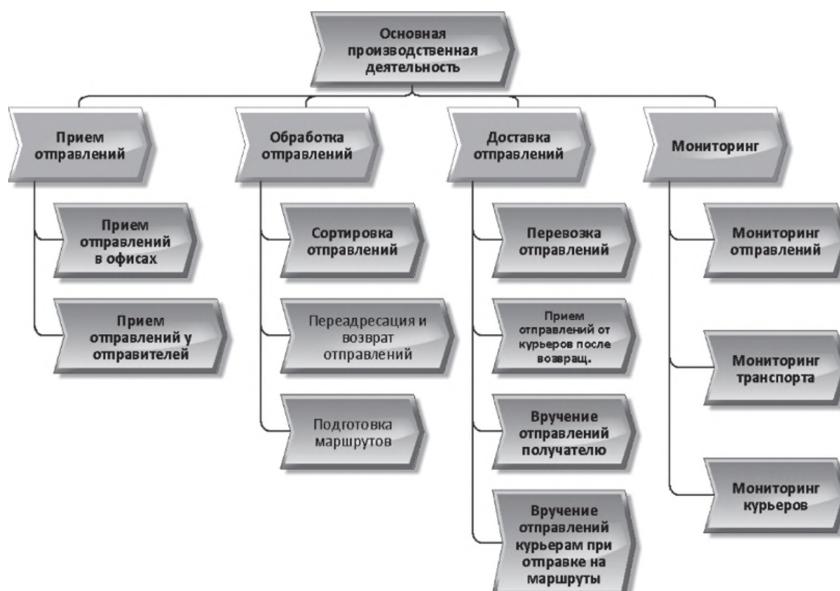


Рис. 8.9. Карта процессов в нотации EPC (программное средство ARIS Express)

Из рис. 8.9 видно, что бизнес-процессы компании «Экспресс-доставка», описанные в подп. 8.1.1, сгруппированы в четыре блока:

- прием отправлений;
- обработка отправлений;
- доставка отправлений;
- мониторинг.

Каждый блок содержит конкретные бизнес-процессы. Соответственно, следующим шагом будет моделирование описанных бизнес-процессов.

**Моделирование бизнес-процессов в нотации BPMN.** Диаграммы бизнес-процессов нужно построить для всех бизнес-процессов, отраженных на карте процессов.

Для этого воспользуемся средством Bizagi Modeler, которое распространяется бесплатно и применяется для графического описания бизнес-процессов в нотации BPMN 2.0. Для начала моделирования бизнес-процесса необходимо просто открыть Bizagi Modeler. Сразу после этого можно приступать к моделированию. Описание нотации BPMN приведено в подп. 4.4.2.

В качестве примера смоделируем бизнес-процессы из группы процессов «Прием отправлений»: «Прием отправлений в офисах» и «Прием отправлений у отправителей».

Теперь можно переходить к непосредственному моделированию бизнес-процессов «Прием отправлений в офисах» и «Прием отправлений у отправителей» (рис. 8.10—8.11).

Обратите внимание, что бизнес-процесс пронизывает деятельность предприятия насквозь, проходя через несколько функций. Результаты бизнес-процессов приема инициируют бизнес-процесс «Сортировка».

**EPC как альтернативная нотация моделирования бизнес-процессов.** Моделирование бизнес-процессов также может проводиться на основе нотации EPC. Для моделирования бизнес-процессов в ARIS Express нужно создать новую диаграмму бизнес-процесса (File → New → Business Process). Описание нотации EPC приведено в подп. 4.4.2.

В качестве примера смоделируем бизнес-процесс «Прием отправлений в офисах». Из-за особенностей размещения элементов диаграммы EPC традиционно занимают большой объем, поэтому будет приведен только фрагмент (рис. 8.12).

Из диаграммы EPC видно, что бизнес-процесс не автоматизирован ИС или иными программными средствами. Всю работу выполняет офис-менеджер компании «Экспресс-доставка», что негативно сказывается на эффективности процесса и скорости его выполнения.

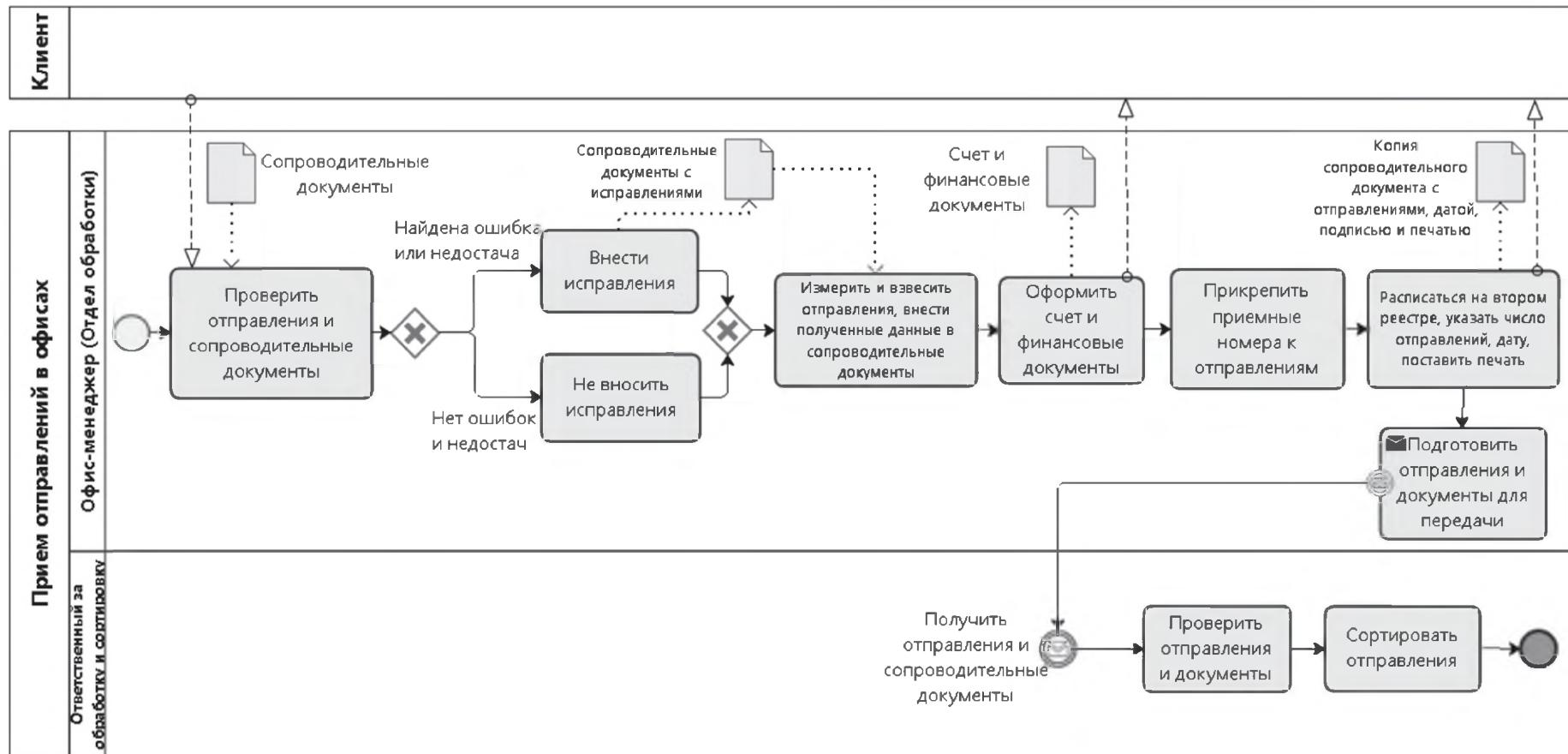
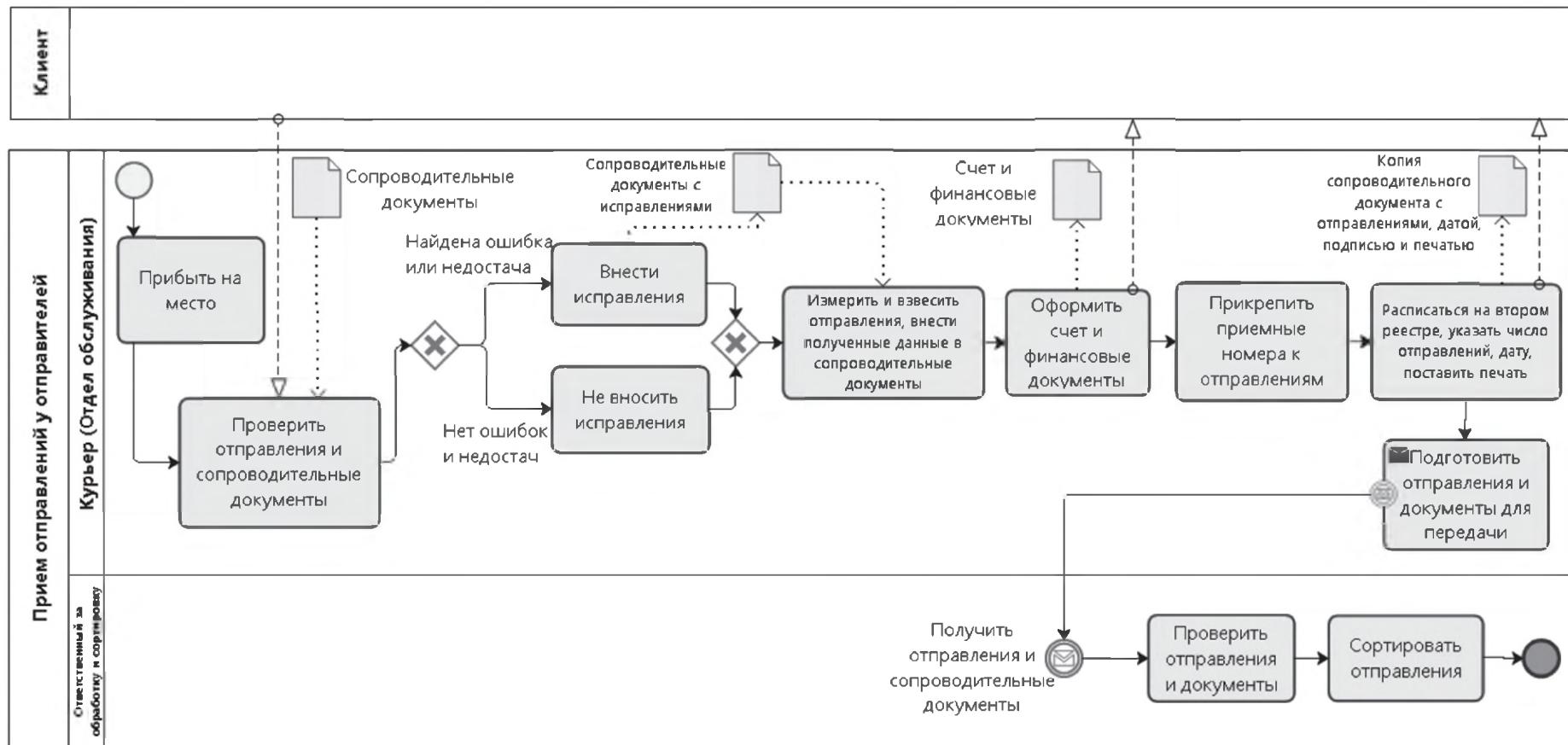


Рис. 8.10. Бизнес-процесс «Прием отправлений в офисах» в нотации BPMN



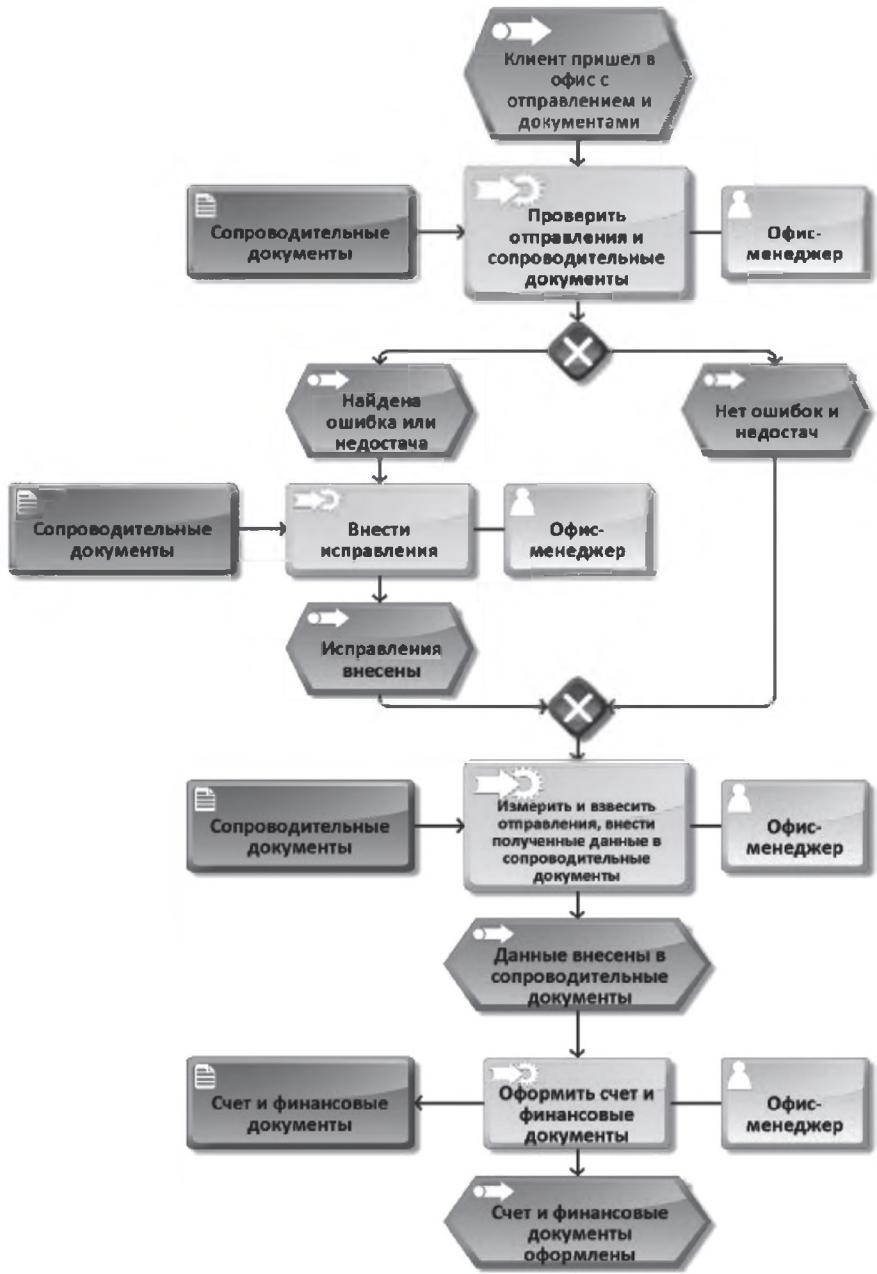


Рис. 8.12. Фрагмент бизнес-процесса «Прием отправлений в офисах» в нотации ЕРС (программное средство ARIS Express)

**Интерпретация результатов моделирования бизнес-процессов.** Карта процессов, построенная на первом шаге, отражает основные верхнеуровневые процессы основной производственной дея-

тельности. Верхнеуровневые бизнес-процессы детализируются до уровня конкретных бизнес-процессов, существующих в компании. Иными словами, карта процессов показывает, как и за счет чего организация выполняет свою деятельность.

На следующем шаге производится моделирование конкретных бизнес-процессов компании. Моделирование бизнес-процессов позволяет понять, как конкретно работает компания. В дальнейшем будет производиться автоматизация рассмотренных бизнес-процессов и создание новых бизнес-процессов. При оптимизации бизнес-процессов требуется опираться на результаты моделирования бизнес-процессов. В противном случае ИС не будет соответствовать потребностям предприятия.

### **Функциональный анализ**

**Построение диаграмм IDEF0.** Структурный анализ может начинаться как с моделирования бизнес-процессов, так и с анализа функциональной области предприятия. Дальнейший анализ функций предприятия будет проведен на основе диаграмм IDEF0.

Для построения диаграмм IDEF0 будет использоваться инструмент Ramus Educational 1.1.1. Элементы графической нотации IDEF0 рассмотрены в подп. 4.2.3.

Первый шаг в анализе функций предприятия при помощи диаграмм IDEF0 — это построение контекстной диаграммы. Контекстная диаграмма в самом общем виде описывает деятельность компании «Экспресс-доставка». Единственный функциональный блок, который присутствует на контекстной диаграмме, отражает основную производственную деятельность компании «Экспресс-доставка».

Результат построения контекстной диаграммы приведен на рис. 8.13. Обратите внимание, что контекстная диаграмма требует минимальной детализации.

Теперь основную производственную деятельность нужно разбить на составные функции. Для этого требуется произвести декомпозицию. Учитывая масштаб компании «Экспресс-доставка», первая декомпозиция должна выделить только основные функциональные блоки.

Основные функциональные блоки становятся очевидными после прочтения «Описания бизнес-процессов» (подп. 8.1.1). В данном случае можно выделить следующие функции: «Прием отправлений», «Обработка отправлений» и «Доставка отправлений».

Результаты построения диаграммы декомпозиции первого уровня приведены на рис. 8.14.

Разумеется, на таком результате нельзя остановиться. Декомпозицию нужно продолжать. Очевидно, что каждый функциональный блок содержит в себе какие-то конкретные функции. Их и нужно выделить на диаграмме декомпозиции второго уровня.

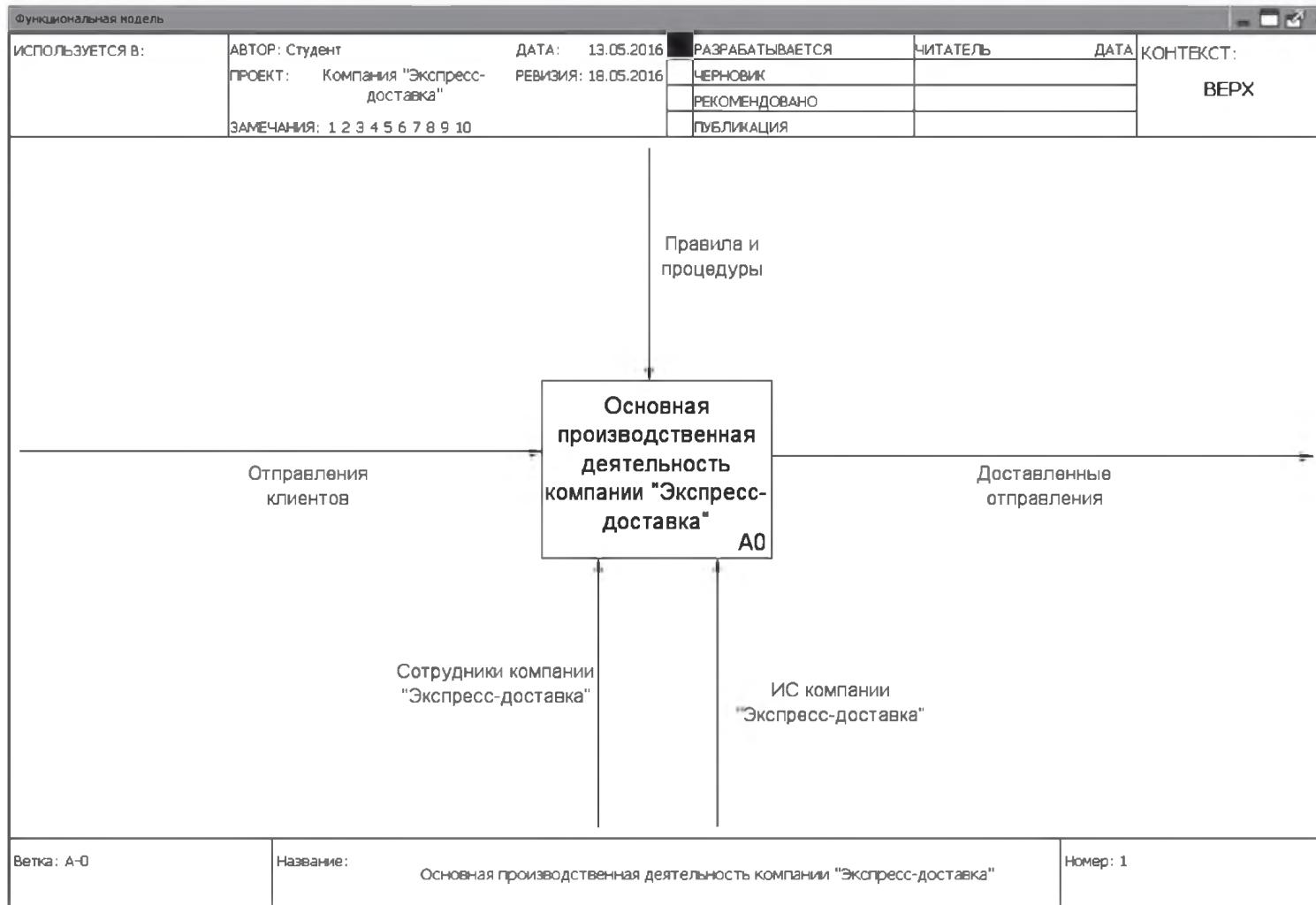


Рис. 8.13. Контекстная диаграмма для предприятия «Экспресс-доставка»

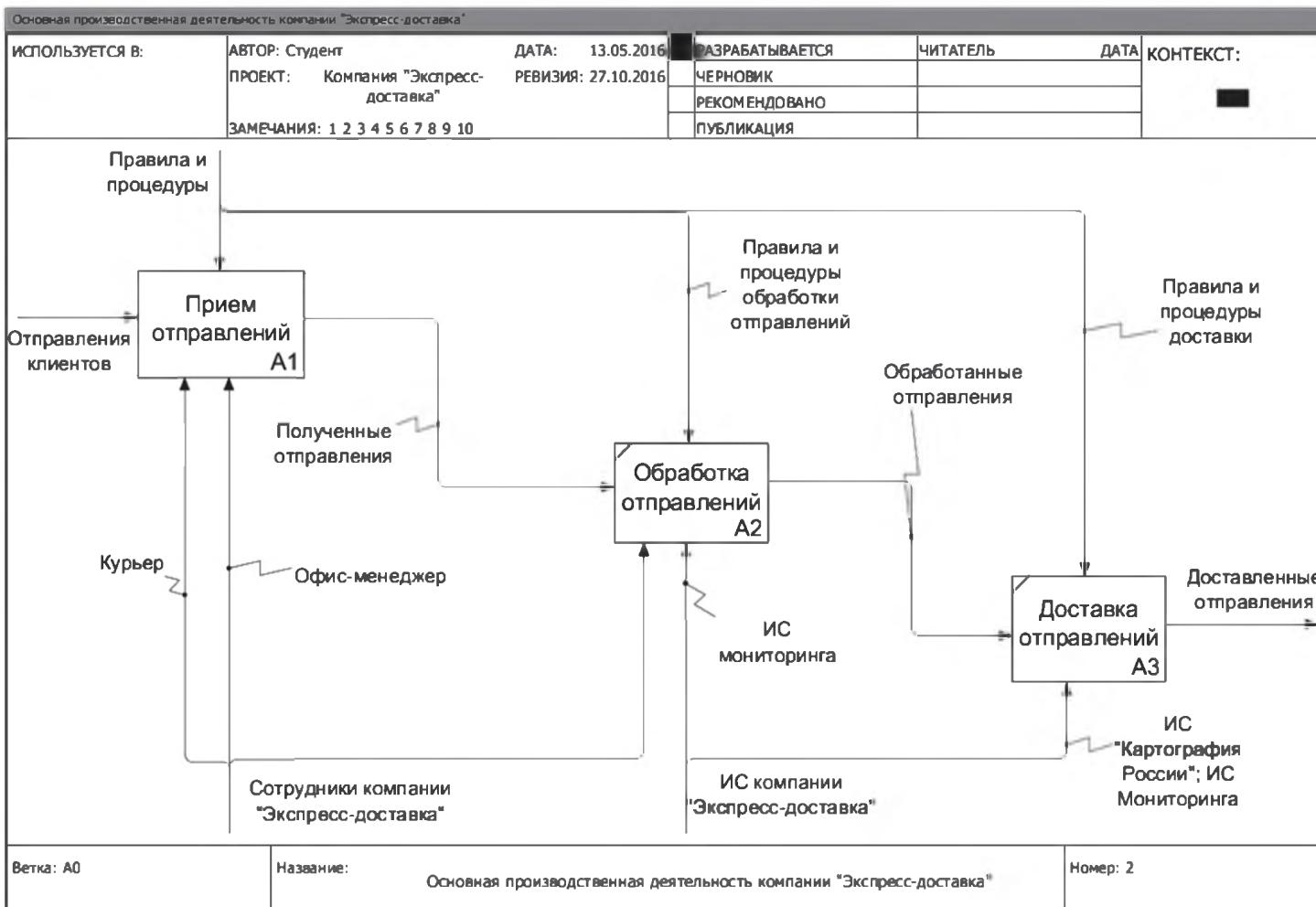


Рис. 8.14. Диаграмма декомпозиции первого уровня (группа основных функций)

Для примера возьмем функциональный блок «Прием отправлений». Из описания деятельности компании становится очевидным, что здесь можно выделить две функции: «Прием отправлений в офисах» и «Прием отправлений у отправителей».

Отразим эти функции на диаграмме декомпозиции второго уровня.

Обратите внимание, что предварительно в программе Rarus Educational необходимо выбрать функциональный блок «Прием отправлений» и проводить именно его декомпозицию.

Диаграммы декомпозиции второго уровня нужно построить для каждого функционального блока диаграмм декомпозиции первого уровня (т. е. в данном случае должно получиться три диаграммы декомпозиции второго уровня).

Результаты построения диаграммы декомпозиции второго уровня приведены на рис. 8.15. Аналогичные диаграммы нужно построить для функциональных блоков «Обработка отправлений» и «Доставка отправлений».

И, наконец, можно провести завершающую декомпозицию (рис. 8.16). В качестве примера возьмем функцию «Прием отправления в офисах». Исчерпывающее описание функции «Прием отправлений у отправителей» содержится в подпараграфе 8.1.1. На основе этого описания требуется произвести декомпозицию.

Обратите внимание, что диаграмму декомпозиции необходимо построить для каждой функции, отраженной в диаграммах декомпозиции второго уровня.

**Построение диаграмм DFD.** Обычно функциональной декомпозиции недостаточно. Действительно, диаграммы IDEF0 отражают прежде всего функции предприятия. Дополнительно требуется отобразить информационное окружение этих функций.

Диаграммы DFD используются для моделирования и анализа информационных потоков предприятия (рис. 8.17). Общая логика проста: диаграммы декомпозиции IDEF0 описывают функции предприятия, а диаграммы DFD описывают информационные потоки каждой функции. Таким образом, диаграммы DFD необходимо построить для каждой диаграммы декомпозиции третьего уровня.

Диаграммы DFD в Rarus Educational строятся в два этапа. На первом этапе необходимо создать Новый проект и выбрать соответствующий тип диаграммы. В нем нужно разместить функциональный блок, который соответствует декомпозируемой функции. Не обязательно обозначать на контекстной диаграмме Входы, Выходы, Управление и Ресурсы: допускается сразу произвести декомпозицию.

На втором этапе происходит построение детализированной диаграммы DFD. Элементы графической нотации подробно рассмотрены в подп. 4.2.4.

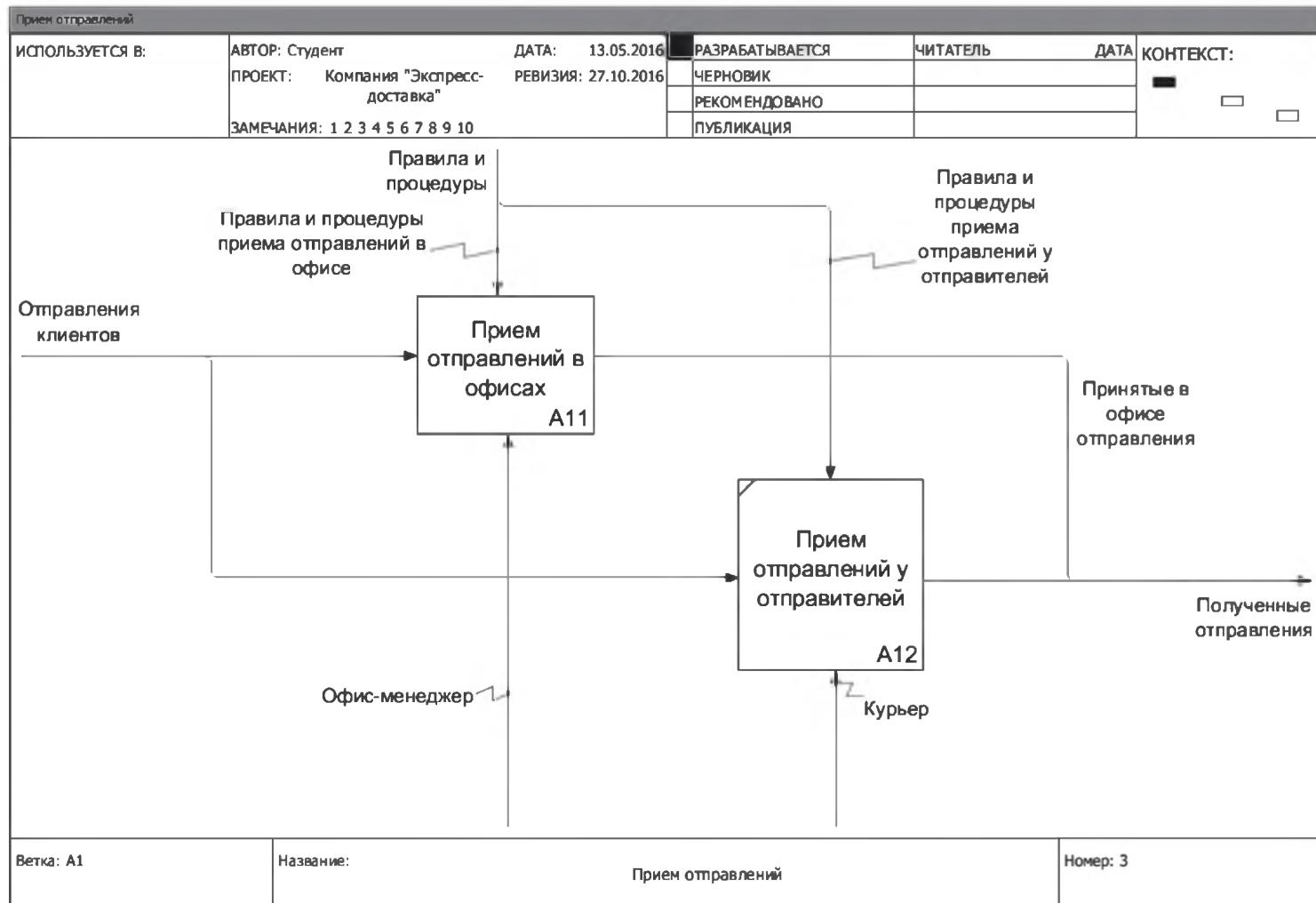


Рис. 8.15. Диаграмма декомпозиции второго уровня (для функционального блока «Прием отправлений»)

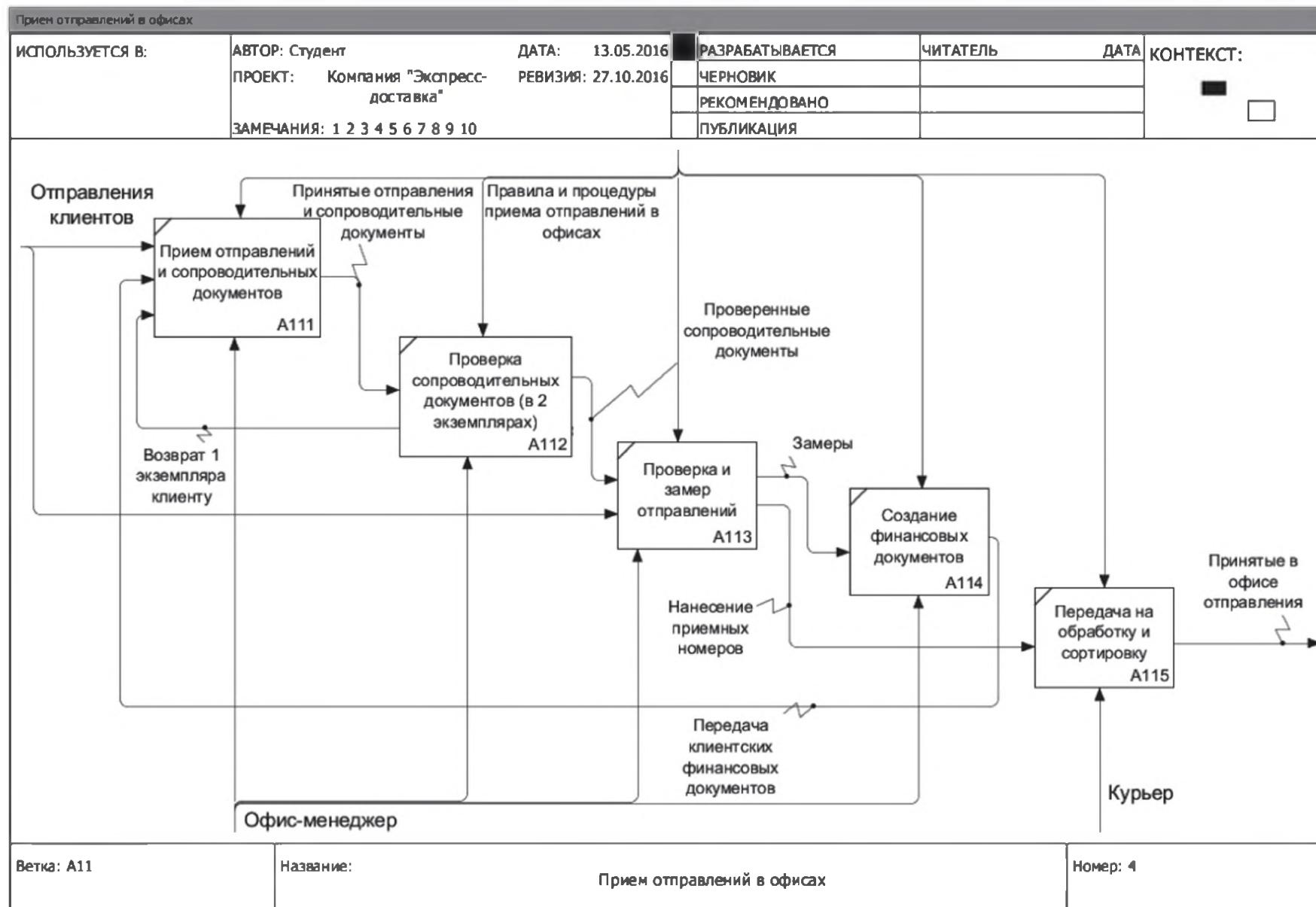


Рис. 8.16. Результат декомпозиции функции «Прием отправлений в офисах» (декомпозиция третьего уровня)

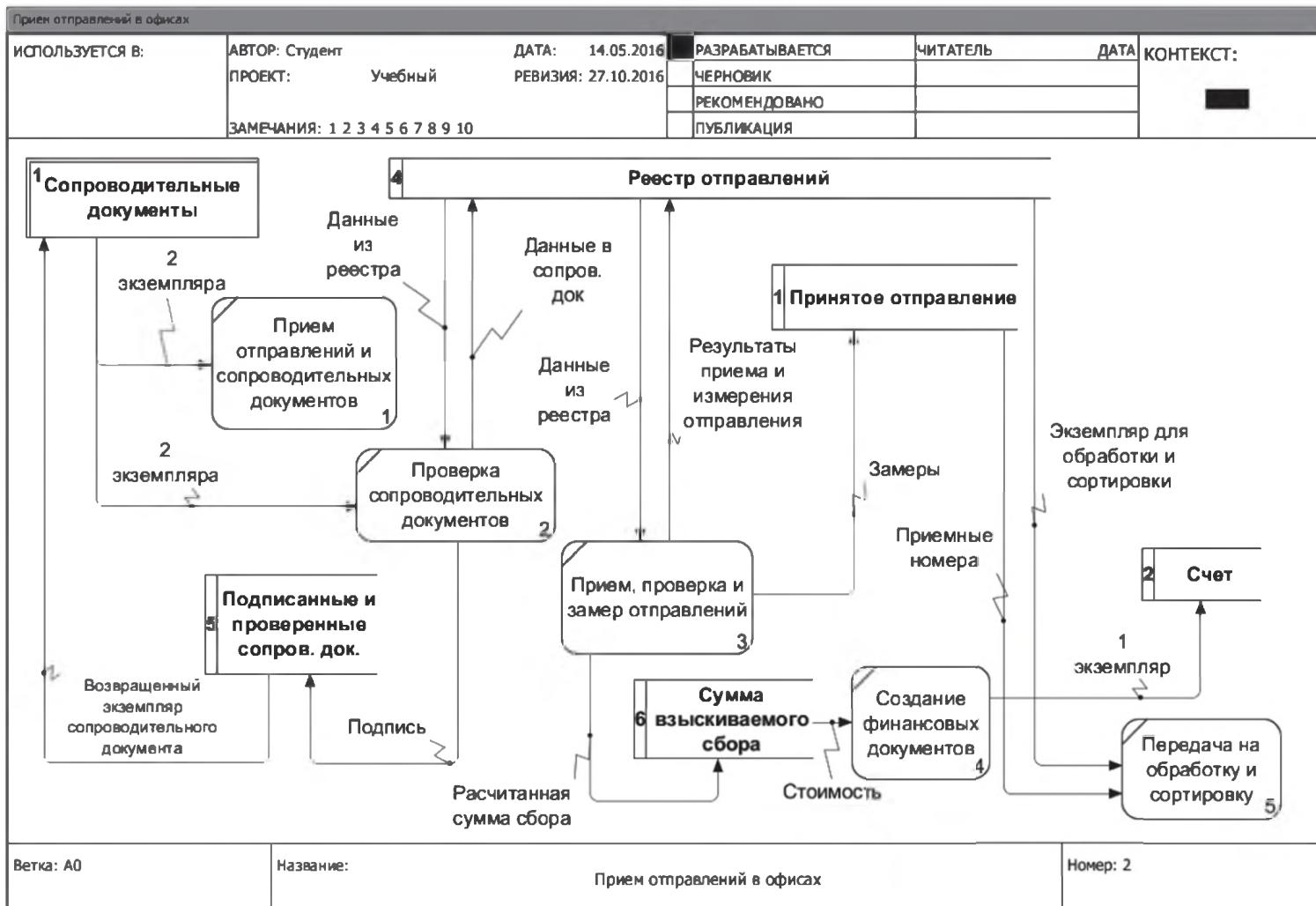


Рис. 8.17. Построение DFD-диаграммы для процесса «Прием отправлений в офисах»

Отметим, что нельзя дать название классификатора напрямую. Сначала требуется открыть блок «Классификаторы» (находится слева) и добавить туда требуемый классификатор. Затем Внешняя ссылка или Хранилище данных, добавленные на диаграмму, могут быть связаны с добавленным ранее классификатором.

Следует отметить несколько важных деталей:

- на диаграмме DFD не обозначают связи непосредственно между функциональными блоками;
- связи между классификаторами и функциональными блоками рекомендуется подписывать для лучшего восприятия диаграммы.

*Интерпретация результатов функционального анализа.* Главный результат функционального анализа — это исчерпывающее описание функциональной области предприятия и информационного окружения функций.

В результате построения диаграмм IDEF0 в руках разработчиков оказывается описание функциональной области предприятия. Это позволяет понять, чем конкретно занимается предприятие, какими функциями представлена его деятельность.

Построение диаграмм DFD позволяет понять информационное окружение каждой функции. В результате разработчики имеют не только «голое» описание функциональной области, но также и информационные потоки исследуемого предприятия.

Описание функциональной области предприятия позволяет представить возможности компании. Именно поэтому требуется создавать так много диаграмм IDEF0 и DFD: функции предприятия и соответствующие информационные потоки должны быть охвачены целиком, чтобы в дальнейшем выбрать область автоматизации, а также не упустить какие-либо важные бизнес-асpekты или проблемные места.

### **Анализ информационной модели**

*Диаграммы ERD.* В рамках структурного анализа также должна быть построена модель «сущность-связь» в форме ER-диаграммы. Она позволяет построить концептуальную схему предметной области. ER-модель включает только ключевые сущности и связи между ними.

Для ее построения можно воспользоваться программным продуктом Microsoft Visio, который хоть и не является бесплатным, но очень широко распространен. Версии MS Visio после 2010 поддерживают различные нотации ER-диаграмм, включая нотацию П. Чена и нотацию Г. Эвереста. В данном случае воспользуемся наиболее распространенной нотацией П. Чена. Элементы графической нотации приведены в подп. 4.3.3.

Исходные данные, на основе которых построены диаграммы, приведены выше в описании компании.

В приведенном выше примере построена ER-диаграмма для предметной области, связанной с бизнес-процессами «Прием отправлений в офисах» и «Прием отправлений у отправителей» (рис. 8.18). В рамках контрольного задания необходимо построить ER-диаграмму, которая будет включать концептуальную схему предметной области для всего предприятия «Экспресс-доставка».

**Определение границ автоматизации.** Анализ моделей бизнес-процессов, функциональной области и информационной модели предприятия позволяет понять, какие конкретно процессы и функции должны быть автоматизированы создаваемой информационной системой. В рамках данного примера будет автоматизироваться функциональная область «Прием отправлений», состоящая из функций «Прием отправлений в офисах» и «Прием отправлений у отправителей».

Этот выбор мы сделаем ввиду отсутствия автоматизации этой области. Соответственно, автоматизация именно этой функциональной области позволит предприятию существенно снизить издержки и повысить эффективность.

Стоит отметить, что в ряде случаев область автоматизации может быть выбрана еще до анализа функциональной и информационной модели. Но, как правило, такой подход характерен только для небольших ИС.

#### 8.2.4. Требования к информационной системе

При формировании требований к информационной системе воспользуемся классификацией FURPS+ (см. подп. 3.2.3). Данная классификация включает:

- функциональные требования;
- требования к удобству использования;
- требования к надежности;
- требования к производительности;
- требования к поддержке;
- ограничения.

Важно понимать, что в результате анализа формируется простой список требований, разделенный по группам. Далее при работе с этими требованиями аналитики часто присваивают им атрибуты.

Для демонстрации примера требований с атрибутами воспользуемся методом расстановки приоритетов MoSCoW и рекомендациями RUP. В данном случае воспользуемся набором RUP, который включает следующие атрибуты:

- статус (Status):
  - предложенные (Proposed),
  - одобренные (Approved),
  - отклоненные (Rejected),
  - включенные (Incorporated);

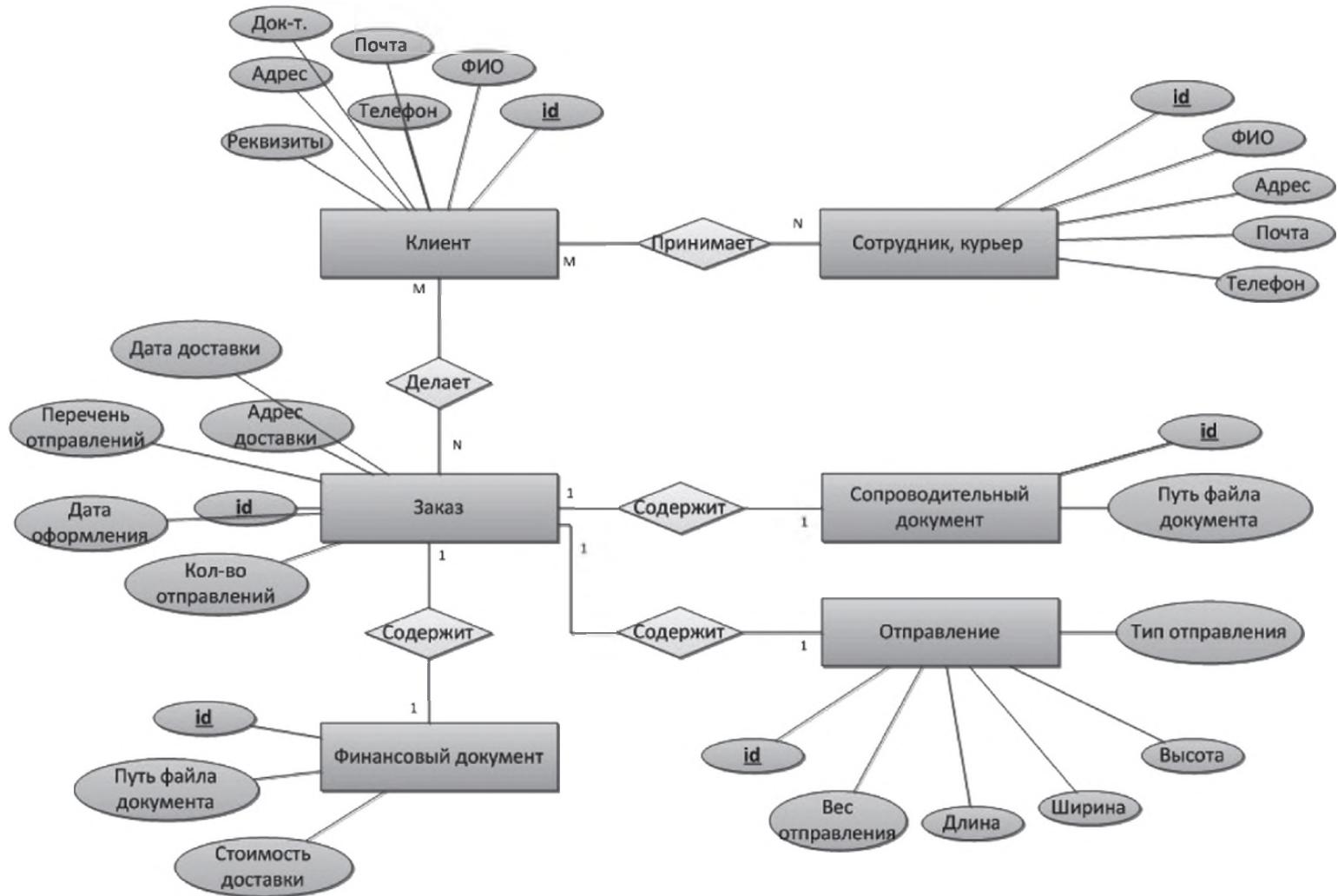


Рис. 8.18. ER-диаграмма в MS Visio 2010 в нотации Питера Чена

- полезность (Benefit):
  - критическое (Critical),
  - важное (Important),
  - полезное (Useful);
- трудоемкость (Effort) — выражается в финансах, человеко-часах и т. п.;
- риск (Risk) — оценивается как Высокий, Средний или Низкий (В, С, Н);
- стабильность (Stability) — оценка вероятности того, что требование будет изменено (В, С, Н).

Далее сформируем спецификацию требований к информационной системе на основе классификации FURPS+ и зададим атрибуты для них на основе набора RUP. Еще раз отметим, что для первичного формирования спецификации требований к ИС, которые готовят бизнес- или системный аналитик, было бы достаточно составления простого списка без атрибутов.

**Функциональные требования.** Применение классификации FURPS+ обычно начинается с формирования функциональных требований. В табл. 8.2 приведены примеры функциональных требований к разрабатываемой информационной системе и их атрибутов.

Таблица 8.2

**Функциональные требования и их атрибуты**

Функциональные требования	Атрибуты				
	Статус	Полезн.	Труд	Риск	Стаб.
Создание нового отправления	Одобр.	Критич.	10 ч	Н	В
Внесение основных характеристик отправления (наименование, адрес, срочность, габариты)	То же	То же	10 ч	Н	В
Внесение информации об отправителе (ФИО, номер телефона, e-mail)	—"—	—"—	10 ч	Н	С
Указание типа отправления (корреспонденция, посылки и упаковки, ценные отправления, опасные отправления)	—"—	—"—	5 ч	Н	В
Удаление/внесение изменений в информацию об отправлении	—"—	Важное	5 ч	С	В
Возможность выбрать адрес отправления и назначения из списка	Предлож.	Полезн.	15 ч	В	Н

Функциональные требования	Атрибуты				
	Статус	Полезн.	Труд	Риск	Стаб.
Возможность внести адрес отправления и назначения вручную	Одобр.	Критич.	5 ч	Н	В
Расчет общей суммы сбора для отправления	То же	То же	7 ч	Н	В
Присвоение отправлению индивидуального номера после подтверждения корректности внесенной информации об отправлении	—“—	—“—	10 ч	С	В
Составление реестра с указанием количества позиций в отправлении, их наименовании, адресом назначения, датой и временем приема, ФИО принимающего, номером принимающего офиса	—“—	—“—	5 ч	В	С
Печать сформированного реестра	Пред-лож.	Полезн.	10 ч	Н	С
Автоматическое отправление номера отправления отправителю на телефон, e-mail	Отклон.	Полезн.	20 000 руб.	В	Н
Обновление времени приема при внесении корректировки в информацию об отправлении на этапе приема в случае замечаний со стороны отправителя (старая версия не сохранена)	Одобр.	Важное	2 ч	С	В
Присвоение отправлению статуса «принят» после подтверждения принимающим сотрудником	То же	Важное	2 ч	Н	С
Хранение информации об отправлении	—“—	Критич.	70 000 руб.	С	В

После описания функциональных требований начинается разработка нефункциональных требований. В классификации FURPS+ первая группа нефункциональных требований — это требования к удобству использования.

**Требования к удобству использования.** Требования к удобству использования и их атрибуты приведены в табл. 8.3.

Таблица 8.3

**Требования к удобству использования и их атрибуты**

Требования к удобству использования	Атрибуты				
	Статус	Полезн.	Труд, ч	Риск	Стаб.
К информационной системе должна быть разработана инструкция по ее использованию	Одобр.	Важное	40	Н	В
В системе должны быть предусмотрены форматы заполнения определенных полей (в поле «Номер телефона» нельзя вбить буквы; список типов отправлений ограничен и задан заранее и др.)	Предлож.	Полезн.	15	В	С

Параллельно должна быть начата разработка требований к надежности.

**Требования к надежности** (табл. 8.4) очень важны, поскольку от их точного указания зависит стабильная работа создаваемой ИС.

Таблица 8.4

**Требования к надежности и их атрибуты**

Требования к надежности	Атрибуты требования				
	Статус	Полезн.	Труд, ч	Риск	Стаб.
Частота сбоев не выше чем 1 раз/мес	Одобр.	Критич.	20	В	В
Среднее время устранения сбоя до 1 ч	Одобр.	Важное	10	С	С
После сбоя система должна быть восстановлена без потери данных	Предлож.	Критич.	10	В	В
Режим работы ИС — 7 дней в неделю (1 ч в день — профилактический перерыв)	Одобр.	То же	10	С	В
Система должна иметь механизм реализации автоматического дублирования наиболее важных компонентов	То же	—“—	15	В	В
Система должна обеспечивать резервирование наиболее важных пользовательских данных и возможность их автоматического восстановления	—“—	—“—	10	В	В

Требования к надежности	Атрибуты требования				
	Статус	Полезн.	Труд, ч	Риск	Стаб.
Система должна иметь механизм уведомления системного администратора о возникновении ошибок	Предлож.	Важное	5	Н	С

После завершения разработки требований к надежности или параллельно ей можно начинать формулировать требования к производительности.

**Требования к производительности.** Требования к производительности (табл. 8.5) определяют производительность ИС: пропускную способность, скорость обработки информации, пиковую нагрузку и т. п. — в зависимости от контекста использования ИС.

Таблица 8.5

#### Требования к производительности и их атрибуты

Требования к производительности	Атрибуты требования				
	Статус	Полезн.	Труд, ч	Риск	Стаб.
Время отклика системы на запрос пользователя не должно превышать 1 с	Одобр.	Критич.	10	С	В
Допустимое количество одновременно работающих вместе пользователей не должно быть более 500 чел. (при расширении филиальной сети до 3000 чел.)	Обсужд.	Важное	15	В	В
Время запуска или перезапуска ИС не должно превышать 5 мин	Отклон.	Полезн.	20	С	Н

Далее необходимо сформулировать последние нефункциональные требования — требования к поддержке.

**Требования к поддержке.** Важность этих требований (табл. 8.6) нельзя недооценивать, поскольку именно от них зависит, какими силами будет поддерживаться ИС (силами заказчика, исполнителя или передана внешним организациям).

Таблица 8.6

#### Требования к поддержке и их атрибуты

Требования к поддержке	Атрибуты требования				
	Статус	Полезн.	Труд, ч	Риск	Стаб.
Система должна иметь базу знаний и сопроводительную документацию	Обсужд.	Важное	20	Н	В

Требования к поддержке	Атрибуты требования				
	Статус	Полезн.	Труд, ч	Риск	Стаб.
Профилактические работы с системой должны быть осуществимы специалистами Заказчика без привлечения сторонней помощи	Одобр.	Критич.	12	В	В
Расширение функциональности системы должно осуществляться силами внутренней службы ИТ	То же	Критич.	3	В	В
Работоспособность системы не должна зависеть от стороннего ПО	—" —	Критич.	3	Н	В
Изменение программного кода системы должно быть возможным без участия разработчика	Обсужд.	Важное	1	В	В

И, наконец, необходимо определить ограничения создаваемой ИС.

**Ограничения.** Обратите внимание, что в классификации FURPS+ ограничения рассматриваются в качестве требований, поэтому атрибуты требований применяются без каких-либо доработок или дополнений (табл. 8.7).

Таблица 8.7  
Ограничения и их атрибуты

Ограничения	Атрибуты требования				
	Статус	Полезн.	Труд, ч	Риск	Стаб.
Модуль мобильных устройств должен быть совместим с iOS и Android, а серверная часть должна быть совместима с Windows XP, Windows 7, Windows 8, Windows 10	Одобр.	Критич.	20	С	В
ИС должна поддерживать документы форматов MS Office 2007, 2010, 2015 версий	Одобр.	Важное	8	Н	С

Теперь, когда функциональные и нефункциональные требования (включая ограничения) сформированы, можно приступить к следующему этапу проекта.

### 8.2.5. Выбор готового программного обеспечения

В ряде случаев не обязательно проектировать и реализовывать программный продукт с нуля. Если речь идет о компании с типовыми для рынка бизнес-процессами, очень часто можно найти гото-

вый продукт, который прекрасно подойдет для дальнейшего внедрения или доработки.

Как правило, на рынке представлены сразу несколько конкурирующих продуктов. Каждый из них обладает своими преимуществами и недостатками, сильными и слабыми сторонами. Для обоснования выбора воспользуемся методом весовых коэффициентов.

**Шаг 1. Составление списка решений.** На первом шаге составим список программных решений, потенциально пригодных для внедрения в компанию. После анализа рынка программных продуктов для работы с заказами аналитиками совместно с представителями заказчика были выделены три программных продукта:

- Экспедитор Стандарт (сокращение: ЭС);
- БИТ.Экспедирование (сокращение: БИТ);
- 1С: Предприятие 8. ТМС Логистика. Управление перевозками (сокращение: 1С).

Отметим свойства каждого из выбранных программных продуктов.

**Экспедитор Стандарт.** Экспедитор Стандарт — российская разработка, которая содержит пять основных функциональных блоков:

- Работа с клиентами (функции: Прием заказа на перевозку; Автоматическое формирование ответов; Формирование окончательной заявки; Информирование клиента о ходе работ);
- Работа с заявками (функции: Контроль выполнения работ; Поддержка справочников этапов, тарифов, схем; Выпуск документов);
- Календарь мероприятий (обеспечивает управление мероприятиями);
- Трейсинг (реализует трейсинг поставок);
- Справочники (содержит общие справочники и справочники компании).

**БИТ. Экспедирование.** В контексте рассматриваемого кейса необходимо акцентировать внимание на версии ПО «Мультимодальные перевозки». Функциональность ПО включает:

- формирование заявок на перевозку груза;
- подбор автотранспорта исходя из параметров груза;
- формирование прайс-листов компаний;
- оформление путевых листов;
- планирование работы водителей;
- планирование работы автотранспорта;
- формирование сопроводительных документов;
- формирование регламентированных документов и печатных форм;
- контроль документооборота по заявке;
- SMS-оповещение;

- аналитические возможности;
- дополнительные возможности для клиентов.

### **1С: Предприятие 8. ТМС Логистика. Управление перевозками.**

В состав данного программного продукта входят следующие подсистемы:

- управление нормативно-справочной информацией;
- управление потребностями в перевозке грузов;
- управление заданиями на перевозку грузов;
- формирование рейсов;
- управление ресурсами для обеспечения рейсов;
- контроль за выполнением рейсов;
- управление тарифной политикой;
- управление взаимодействиями;
- управление доступом;
- получение аналитической отчетности;
- визуализация информации на картах.

**Шаг 2. Формирование критерииев и определение их веса.** Для данного примера было сформировано 10 критериев и определен их вес (табл. 8.8).

Таблица 8.8

**Критерии и коэффициенты**

Критерий	Вес
Функциональные возможности	0,4
Совокупная стоимость владения	0,2
Возможности масштабирования	0,1
Возможности для доработок	0,05
Возможность интеграции с текущими системами	0,05
Качество технической поддержки	0,05
Надежность поставщика	0,05
Частота обновлений системы	0,05
Соответствие лучшим практикам	0,025
Удобство использования	0,025

На практике вес критерииев обычно расставляет сторона Заказчика.

В данном случае сумма всех коэффициентов равняется единице. Кроме того, от одного до трех критериев чаще всего рассматриваются в качестве основных, поэтому их коэффициенты существенно выше остальных.

Как правило, значение любого критерия варьируется от 0 до 10.

**Шаг 3. Формирование списка экспертов.** Для оценки нашего примера было приглашено 10 экспертов, среди которых представи-

тели Заказчика, представители Исполнителя и сторонние консультанты по вопросам использования рассматриваемых информационных систем.

**Шаг 4. Оценка ИС экспертами.** На четвертом шаге каждый эксперт должен произвести оценку каждой из выбранных информационных систем по десятибалльной шкале на основе представленных критериев. В результате опроса экспертов была сформирована табл. 8.9.

Таблица 8.9

**Средние значения экспертных оценок для каждой ИС**

Критерий	ЭС	БИТ	1С
Функциональные возможности	7	6,8	7,8
Совокупная стоимость владения	6,5	7,1	6,7
Возможности масштабирования	8,1	7,5	7,7
Возможности для доработок	7,2	7,2	6
Возможность интеграции с текущими системами	6,9	7	6,7
Качество технической поддержки	5,9	6,5	8
Надежность поставщика	7,4	6	8,7
Частота обновлений системы	7,1	8,1	6,2
Соответствие лучшим практикам	8,2	6,5	5,9
Удобство использования	7,8	7,5	6,2

Значения каждого критерий для каждой ИС рассчитаны как среднее арифметическое для индивидуальных экспертных оценок. В результате обоснования выбора без учета весовых коэффициентов экспертами был определен лучший продукт: Экспедитор Стандарт.

**Шаг 5. Оценка с учетом весовых коэффициентов.** Приведенная выше таблица (см. табл. 8.9) содержит средние оценки критериев, однако весовые коэффициенты не были использованы. На пятом шаге можно пересчитать значения критериев с учетом согласованных весов.

Таблица 8.10

**Результат применения метода весовых коэффициентов**

№	Критерий	Вес	ЭС	БИТ	1С
1	Функциональные возможности	0,4	2,8	2,72	3,12
2	Совокупная стоимость владения	0,2	1,3	1,42	1,34
3	Возможности масштабирования	0,1	0,81	0,75	0,77
4	Возможности для доработок	0,05	0,36	0,36	0,3

Окончание табл. 8.10

Nº	Критерий	Вес	ЭС	БИТ	1С
5	Возможность интеграции с текущими системами	0,05	0,345	0,35	0,335
6	Качество технической поддержки	0,05	0,295	0,325	0,4
7	Надежность поставщика	0,05	0,37	0,3	0,435
8	Частота обновлений системы	0,05	0,355	0,405	0,31
9	Соответствие лучшим практикам	0,025	0,205	0,1625	0,1475
10	Удобство использования	0,025	0,195	0,1875	0,155
<b>Итого</b>		<b>7,035</b>	<b>6,98</b>	<b>7,3125</b>	

Из табл. 8.10 видно, что наивысшую оценку получила ИС «1С: Предприятие 8. ТМС Логистика. Управление перевозками».

Как видите, метод весовых коэффициентов может существенно повлиять на итоговые оценки. Такая ситуация нередко встречается на практике, поскольку при выборе предпочтение отдается нескольким основным критериям. Поэтому при применении метода весовых коэффициентов нет необходимости формировать больший список критериев.

### 8.2.6. Проектирование

Сразу же нужно обратить внимание на тот факт, что, согласно RUP, процесс создания и доработки различных диаграмм продолжается в течение всех фаз проекта. Таким образом, все рассмотренные далее диаграммы создаются и уточняются не единовременно, а на протяжении нескольких этапов проекта.

**Диаграмма прецедентов.** Мнения специалистов о том, с каких диаграмм начинать разработку моделей, расходятся. Мы в нашем примере начнем разработку с построения диаграмм прецедентов. Диаграмма прецедентов будет рассматривать использование ИС для обеспечения процессов сбора отправлений.

Диаграмма прецедентов содержит не слишком много элементов графической нотации, которые были показаны в подп. 5.4.2 (см. рис. 5.6). Исходными данными для построения всех UML-диаграмм служит описание компании, приведенное в подп. 8.1.1.

Для построения всех UML-диаграмм воспользуемся бесплатным программным средством StarUML 5.0.

На первом шаге построим диаграмму прецедентов, которая охватывает только прецеденты наиболее высокого уровня (рис. 8.19).

Очевидно, что такая диаграмма нуждается в уточнении. С этой целью построим еще две диаграммы прецедентов, которые будут уточнять прецеденты «Прием отправлений в офисах» и «Прием отправлений у отправителей» (рис. 8.20—8.21).

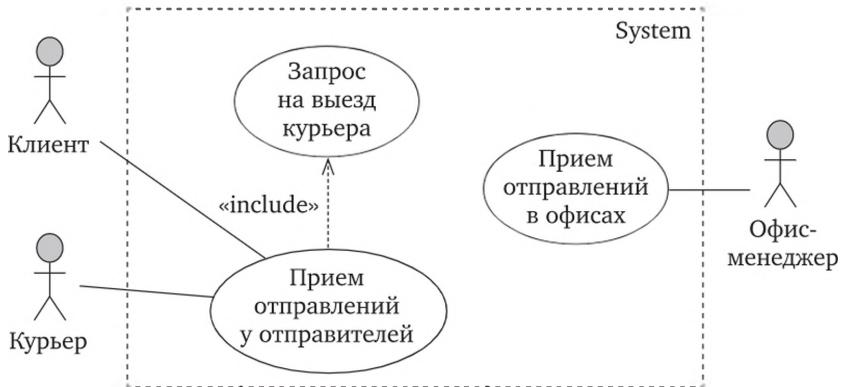


Рис. 8.19. Диаграмма прецедентов верхнего уровня

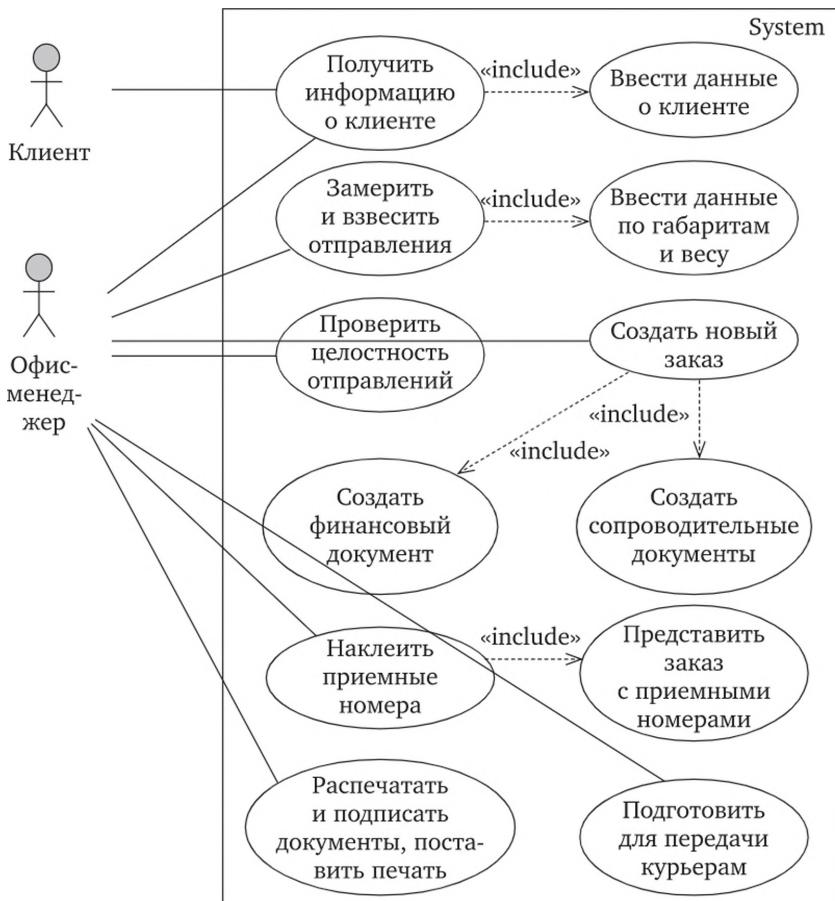
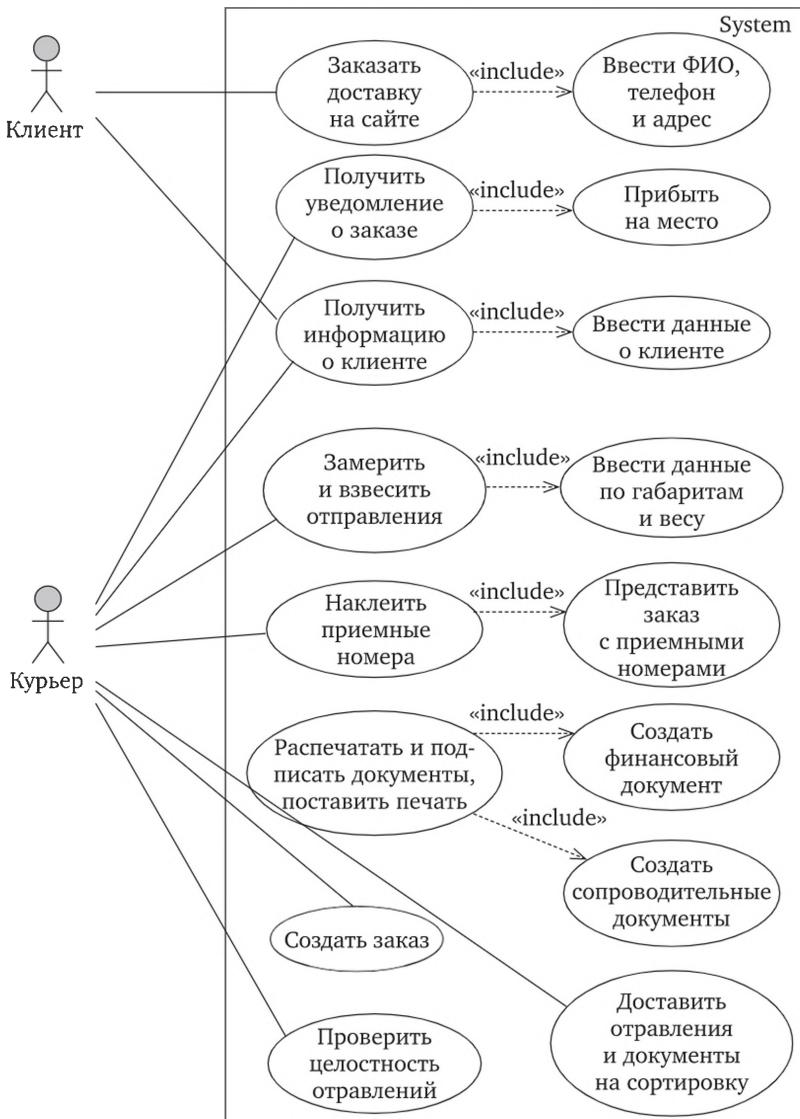


Рис. 8.20. Диаграмма прецедентов для прецедента «Прием отправлений в офисах»



*Рис. 8.21. Диаграмма прецедентов для прецедента «Прием отпправлений у отправителей»*

**Спецификации прецедентов.** Для каждого прецедента необходимо создать спецификацию, которая будет демонстрировать характеристики прецедентов: пред- и постусловия, этапы прецедента, акторов. Приведем пример спецификации для прецедента «Прием отпправлений в офисах» (табл. 8.11).

Таблица 8.11

## Спецификация прецедента «Прием отправлений в офисах»

<b>Прецедент:</b> прием отправлений в офисах
<b>ID:</b> 1
<b>Краткое описание:</b> офис-менеджер принимает отправление у отправителя, пришедшего в офис
<b>Главные акторы:</b> клиент, офис-менеджер
<b>Второстепенные акторы:</b> курьер
<b>Предусловия:</b> клиент пришел в офис с отправлением
<b>Основной поток:</b> 1) прецедент начинается, когда клиент приходит в офис и сообщает о желании отправить отправление; 2) офис-менеджер вводит в ИС данные о клиенте; 3) офис-менеджер вводит в ИС данные о количестве отправлений; 4) офис-менеджер проверяет целостность отправлений, измеряет физические параметры и вводит их в ИС; 5) офис-менеджер создает новый заказ; 6) ИС генерирует и распечатывает приемные номера, офис-менеджер на克莱ивает их на отправления; 7) ИС генерирует и распечатывает счет и финансовые документы, офис-менеджер подписывает их, ставит печать и передает клиенту 1 экземпляр; 8) ИС представляет приемные номера, офис-менеджер наносит их на отправления; 9) офис-менеджер готовит отправления для передачи на сортировку
<b>Постусловия:</b> передача отправлений на сортировку
<b>Альтернативные потоки:</b> нет

Аналогичные спецификации необходимо сформировать для каждого прецедента.

В данном контрольном задании будет целесообразным перейти далее к построению Диаграммы классов, хотя на практике такая последовательность не является обязательной.

**Диаграмма классов.** Исходные данные для ее построения (описание двух автоматизируемых бизнес-процессов) приведены в описании компании, в подп. 8.1.1. Как и ранее, будет использоваться программное средство StarUML 5.0. Подробное описание диаграммы классов приведено в подп. 5.4.3.

Диаграмма классов строится не для всей компании в целом, а исключительно для проектируемого ПО, которое будет поддерживать два конкретных бизнес-процесса (рис. 8.22).

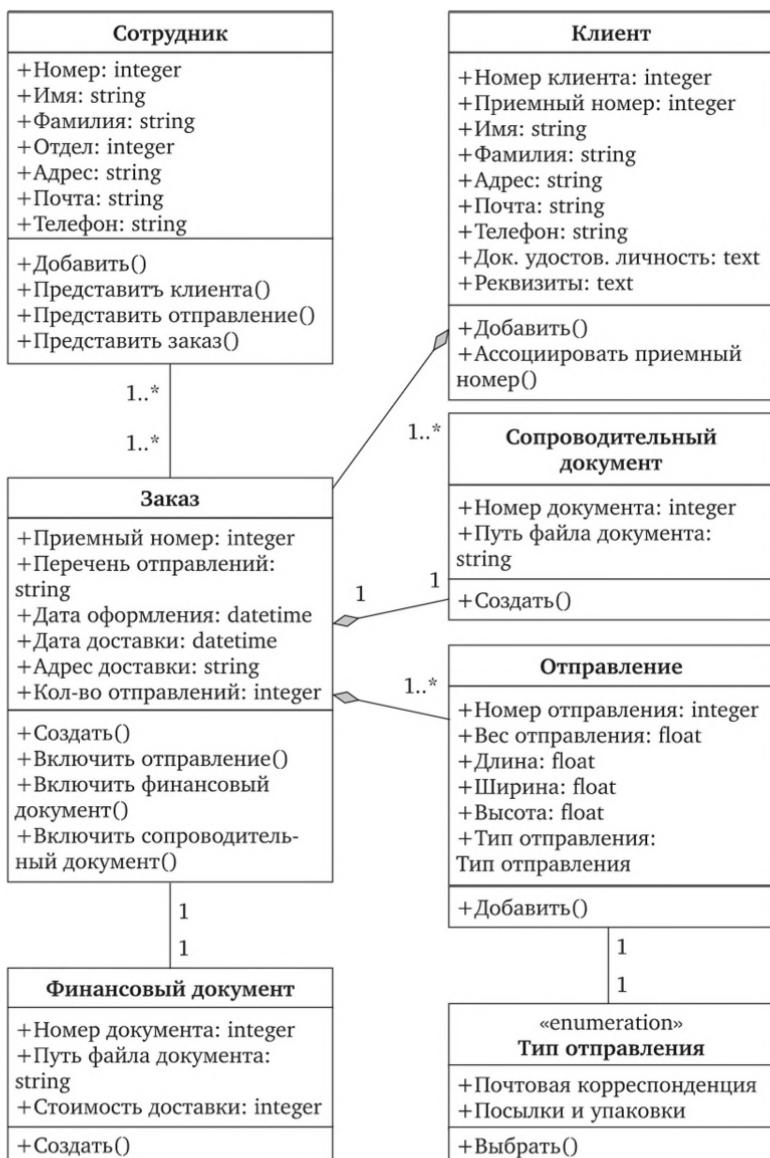


Рис. 8.22. Диаграмма классов в StarUML

**Диаграмма последовательности.** Как правило, при проектировании ИС не обязательно строить одновременно диаграммы последовательности и диаграммы кооперации: достаточно ограничиться

одним типом диаграмм. Тем не менее в рамках контрольного задания необходимо построить диаграммы обоих типов для каждого прецедента, отраженного на диаграмме прецедентов верхнего уровня. Элементы графической нотации Диаграммы последовательности приведены в подп. 5.4.4.

В качестве примера построим диаграмму последовательности для прецедента «Прием отправлений в офисах» (рис. 8.23). Обратите внимание, что названия всех объектов идентичны названиям в Диаграмме классов.

Рассмотрим другой вариант иллюстрации взаимодействия при помощи UML.

**Диаграмма кооперации.** Общие правила построения диаграмм кооперации (рис. 8.24) аналогичны правилам построения диаграмм последовательности. Диаграммы кооперации также строятся для всех прецедентов, отраженных в диаграмме прецедентов верхнего уровня. Элементы графической нотации Диаграммы кооперации приведены в подп. 5.4.4.

Следующим в рамках контрольного задания рассмотрим проектирование Диаграммы состояний.

**Диаграмма состояний.** Диаграммы состояний могут создаваться как для информационной системы в целом, так и для отдельных прецедентов. Элементы графической нотации Диаграммы состояний приведены в подп. 5.4.3.



Рис. 8.24. Пример диаграммы кооперации для прецедента «Прием отправлений в офисах»

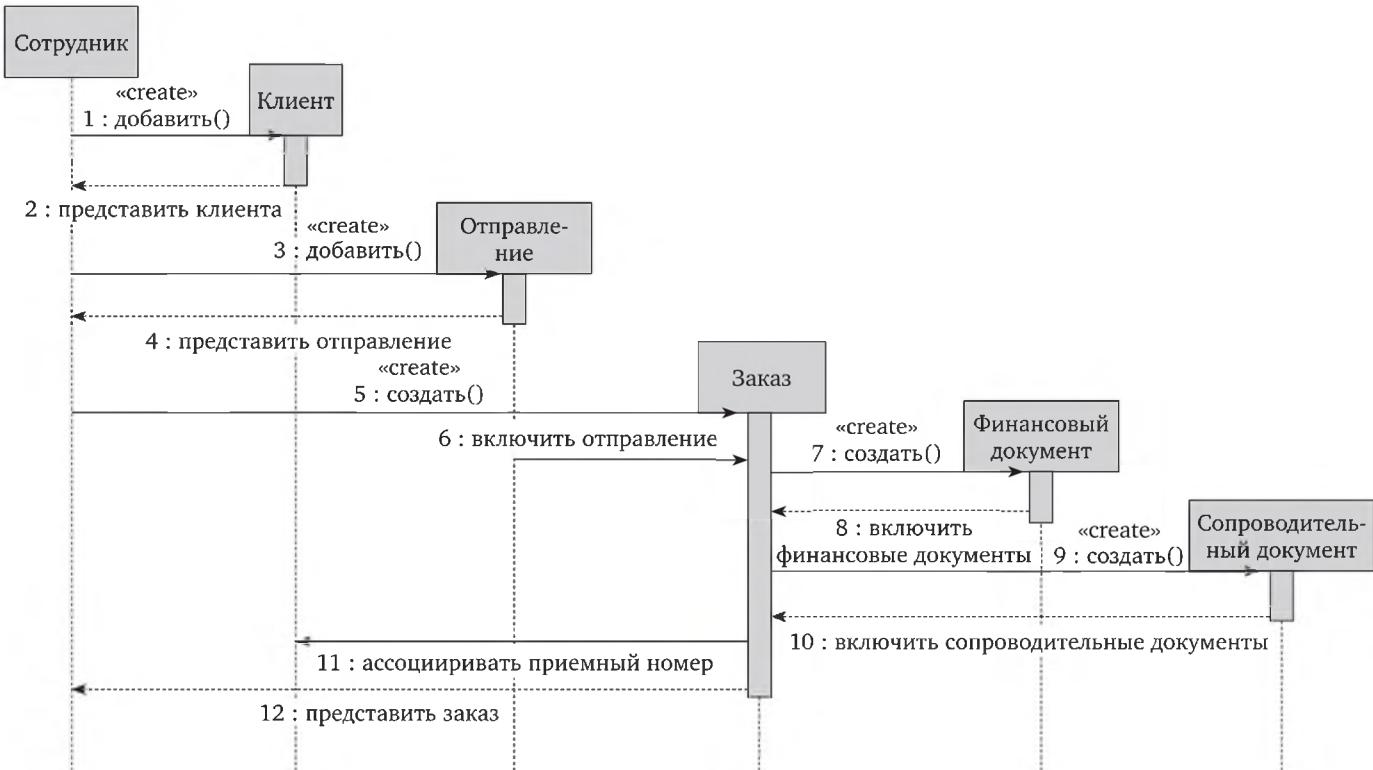


Рис. 8.23. Диаграмма последовательности для прецедента «Прием отправлений в офисах»

Построим Диаграмму состояний для прецедента «Прием отправлений в офисах» (рис. 8.25).

Далее будет целесообразным перейти к построению Диаграмм деятельности.

**Диаграмма деятельности.** Источник данных для построения данной Диаграммы деятельности — это описание соответствующего бизнес-процесса в подп. 8.1.1. Имеет смысл ориентироваться на построенные ранее бизнес-процессы в нотации BPMN или EPC, поскольку общая логика Диаграмм деятельности имеет много общего с моделированием бизнес-процессов. Элементы графической нотации Диаграммы деятельности подробно рассмотрены в подп. 5.4.3.

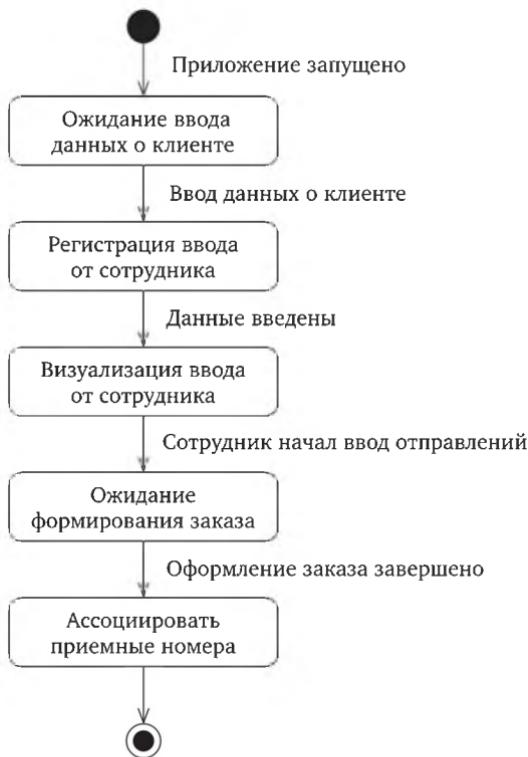


Рис. 8.25. Пример диаграммы состояний для прецедента «Прием отправлений в офисах»

Для примера построим Диаграмму деятельности, описывающую прецедент «Прием отправлений в офисах» (рис. 8.26).

Рассмотрим далее еще один тип UML-диаграмм — Диаграмму компонентов.



*Рис. 8.26. Пример Диаграммы деятельности для прецедента «Прием отправлений в офисах»*

**Диаграмма компонентов.** Диаграмма компонентов проектируется для информационной системы в целом на основании построенных ранее диаграмм UML. Элементы графической нотации Диаграммы компонентов приведены в подп. 5.4.4.

На рис. 8.27 приведен пример Диаграммы компонентов для проектируемой ИС.

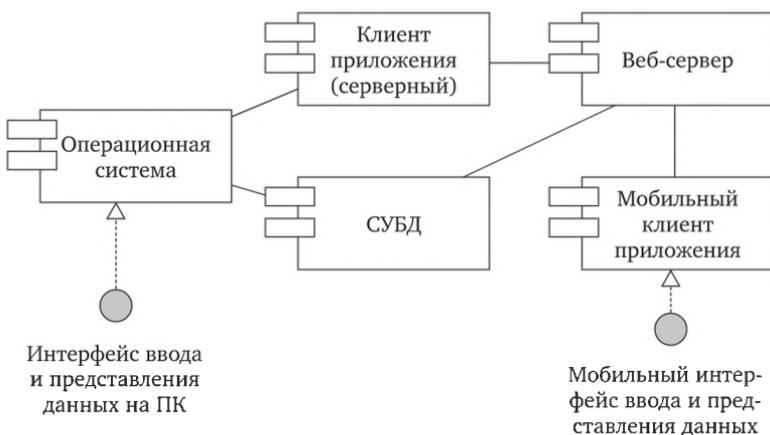


Рис. 8.27. Пример Диаграммы компонентов для проектируемой ИС

Диаграмма компонентов тесно связана с Диаграммой развертывания.

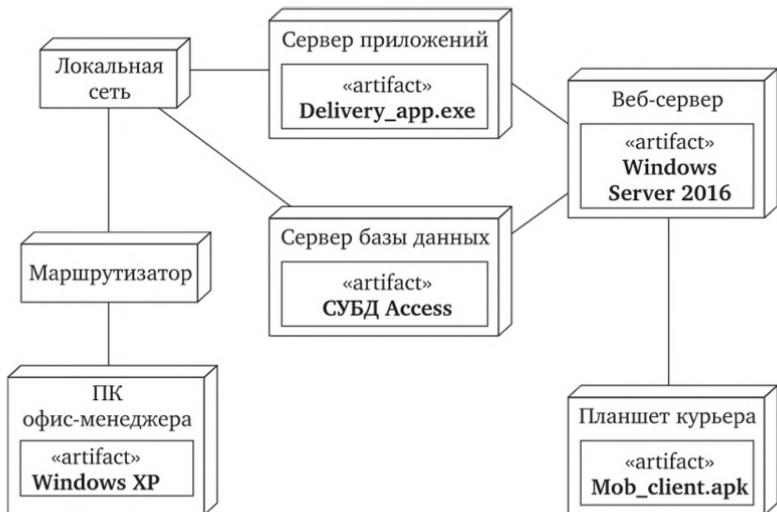
**Диаграмма развертывания.** На завершающем этапе обычно производится формирование диаграммы развертывания. Диаграмма развертывания формируется для ИС в целом на основании построенных ранее диаграмм. Элементы графической нотации Диаграммы состояний приведены в подп. 5.4.3.

Построим пример Диаграммы развертывания для проектируемой ИС (рис. 8.28).

Управление тестированием обычно начинается с составления Плана тестирования. В соответствии с рекомендациями RUP План тестирования должен включать в себя следующие разделы:

- общая информация:
  - цель документа,
  - предмет и цели тестирования,
  - основные этапы тестирования,
  - требования для начала тестирования,
  - стратегия тестирования;
- этапы тестирования:
  - модульное тестирование,
  - функциональное тестирование,

- приемочное тестирование,
- нагрузочное тестирование;
- ресурсы тестирования:
  - роли и обязанности в тестировании,
  - тестовая среда;
- документация тестирования.



*Рис. 8.28. Пример Диаграммы развертывания для проектируемой ИС*

### 8.2.7. Тестирование

Пример Плана тестирования приведен в Приложении 4 к контрольному заданию (подп. 8.2.9).

### 8.2.8. Заключение к контрольному заданию

Для выполнения поставленной в начале контрольного задания цели был решен целый ряд задач. На начальном этапе было осуществлено планирование проекта, которое включало формирование Устава проекта, формирование Реестра рисков (включая Форму регистрации риска), иерархическую структуру работ, лист ресурсов проекта, формирование ролей и обязанностей участников проекта, управление календарем проекта и формирование технико-экономического обоснования.

В ходе работы над проектом был проведен структурный анализ компании ООО «Экспресс-доставка», в ходе которого были выявлены проблемные места и определены границы автоматизации. С этой целью было произведено построение карты процессов и моделирование бизнес-процессов предприятия; был произведен функ-

циональный анализ компании ООО «Экспресс-доставка»; была построена и проанализирована информационная модель компании.

На основании проведенного структурного анализа были сформулированы требования к информационной системе, которая должна автоматизировать отдельные бизнес-процессы компании ООО «Экспресс-доставка». В результате выполнения данного этапа были выделены функциональные требования, требования к удобству использования, требования к надежности, требования к производительности, требования к поддержке и ограничения. Для всех требований были назначены атрибуты.

В рамках проекта был произведен анализ рынка на предмет выбора готового программного решения для компании ООО «Экспресс-доставка», основанный на методе весовых коэффициентов.

Также было произведено проектирование информационной системы с использованием языка UML. В рамках проектирования были построены диаграммы прецедентов (включая спецификации прецедентов), диаграмма классов, диаграммы последовательности, диаграммы кооперации, диаграммы состояний, диаграммы деятельности, диаграмма компонентов и диаграмма развертывания.

Планирование тестирования включало в себя формирование Плана тестирования. План тестирования содержит основную информацию по тестированию, по этапам тестирования, по ошибкам, по критериям начала и окончания этапов тестирования, по метрикам, по ресурсам тестирования и по основным документам тестирования.

В рамках проекта использовались следующие программные средства:

- Microsoft Project (для планирования проекта);
- ARIS Express (для построения карты процессов и для моделирования бизнес-процессов в нотации EPC);
- Bizagi Modeler (для моделирования бизнес-процессов в нотации BPMN);
- Ramus Educational 1.1.1 (для функционального анализа в нотации IDEFO и для анализа информационных потоков в нотации DFD);
- Microsoft Visio (для построения ER-диаграммы в нотации П. Чена);
- StarUML 5 (для анализа и проектирования на основе языка UML).

В рамках проекта использовались различные подходы, методы и методологии, рассмотренные в учебнике.

### 8.2.9. Примеры документов

Далее приведены примеры документов, которые должны возникнуть в процессе выполнения проекта: устав проекта с приложениями к нему, реестр и форма регистрации рисков, план тестирования.

## **Приложение 1 к контрольному заданию**

### **ПРОЕКТ «ЭКСПРЕСС-ДОСТАВКА»**

#### **Устав проекта**

Москва, 2016

Лист контроля над документом  
Запись изменений

Дата	Автор	Роль	Версия	Ссылка на изменение
11.01.2016	Иванов И. И.	РП	01	Предыдущая версия отсутствует
21.01.2016	Николаев Н. Н.	РП	02	Уточнены сроки проекта
01.03.2016	Николаев Н. Н.	РП	03	Передвинут срок формулирования требований

Согласование

№	Дата	Наименование	Автор замечания	Подпись
1	19.01.2016	Перенос сроков	Николаев Н. Н.	
2	26.02.2016	Изменение сроков формулирования требований	Николаев Н. Н.	

Обработка замечаний

№	Дата	Версия с учетом замечания	Исполнитель	Подпись
1	21.01.2016	02	Иванов И. И.	
2	01.03.2016	03	Иванов И. И.	

Распространение

№ копии	ФИО ответственного	Местонахождение документа
1	Иванов И. И.	Проектная библиотека Исполнителя
2	Николаев Н. Н.	Проектная библиотека Заказчика

**1. Введение**

Цель данного документа: утверждение целей проекта, требований к результатам, границ проекта, организационной структуры и ответственности в проекте, процедур проекта.

Основание для проведения работ:

Договор № 100 от «11» января 2016 г.

Ссылки на документы:

1. Договор № 100 от «11» января 2016 г.

**2. Содержание проекта**

**2.1. Цели и задачи проекта**

Цель — автоматизация отдельных видов деятельности компании ООО «Экспресс-доставка».

Задачи проекта — в период с 11 января 2016 г. по 31 августа 2016 г.:

- 1) разработать документ «Модель основных бизнес-процессов ООО “Экспресс-доставка”» по типу «как есть»;
- 2) разработать документ «Функциональная модель предприятия ООО “Экспресс-доставка”»;
- 3) разработать документ «Высокоуровневая модель данных предприятия ООО “Экспресс-доставка”»;
- 4) разработать документ «Предложение по автоматизации ООО “Экспресс-доставка”»;
- 5) разработать документ «Модель автоматизируемых бизнес-процессов ООО “Экспресс-доставка”» по типу «как должно быть»;
- 6) разработать документ «Требования к разрабатываемой информационной системе»;
- 7) разработать документ «Результаты анализа и проектирования информационной системы»;
- 8) провести реализацию информационной системы;
- 9) провести тестирование информационной системы и разработать документ «Результаты тестирования информационной системы»;
- 10) произвести развертывание и внедрение информационной системы в промышленную эксплуатацию компанией ООО «Экспресс-доставка».

## **2.2. Допущения и ограничения**

### **Допущения**

Персонал, критически важный для реализации проекта, не покинет компанию.

Исполнитель вправе привлекать сторонних подрядчиков для выполнения работ.

### **Ограничение по времени**

Все задачи, распределенные по этапам Проекта, планируется выполнить в период между 1 января 2016 г. и 31 августа 2016 г.

### **Ограничения по бюджету**

Совокупная стоимость Проекта не должна превысить 3100 тыс. руб.

### **Ограничения по документам и материалам**

В рамках проекта осуществляется разработка документов и моделей в соответствии с пунктом 2.1.

Для разработки документа «Модель основных бизнес-процессов ООО “Экспресс-доставка”» по типу «как есть» будет применяться методология EPC и BPMN.

Для разработки документа «Функциональная модель предприятия ООО “Экспресс-доставка”» будет применяться методология IDEFO и DFD.

Для разработки документа «Высокоуровневая модель данных предприятия ООО “Экспресс-доставка”» будет применяться методология ERD.

Документ «Предложение по автоматизации» будет включать перечень бизнес-процессов, подлежащих автоматизации.

Для разработки документа «Модель автоматизируемых бизнес-процессов предприятия ООО “Экспресс-доставка”» по типу «как должно быть» будет применяться методология BPMN.

Для разработки документа «Требования к разрабатываемой информационной системе» будет применяться методология FURPS+.

Для разработки документа «Результаты анализа и проектирования» будет использоваться объектно-ориентированный подход в формате языка визуального моделирования UML.

Реализация информационной системы будет осуществляться в среде MS Visual Studio 2010/2013 на языке C#.

#### **Организационные границы**

Участники проекта:

Заказчик: ООО «Экспресс-доставка».

Исполнитель: Организация «Студенты».

Ограничения по количеству командировок команды Исполнителя: пять командировок.

Ограничения по количеству командировок команды Заказчика: по необходимости.

#### **Функциональные границы**

Масштабы: основные функции компании ООО «Экспресс-доставка»; основные бизнес-процессы компании ООО «Экспресс-доставка».

#### **Географические границы**

ООО «Экспресс-доставка» (Москва).

Региональные отделения выходят за границы проекта.

### **3. Основные вехи и результаты**

Название вехи	Срок	Результаты
Начало проекта	11.01.2016	Подготовлено и проведено стартовое совещание
Проектная команда сформирована	13.01.2016	Этап формирования проектной команды завершен
Структурный анализ завершен	28.03.2016	<ul style="list-style-type: none"><li>— Разработана модель основных бизнес-процессов предприятия ООО «Экспресс-доставка» по типу «как есть»;</li><li>— разработана функциональная модель предприятия ООО «Экспресс-доставка»;</li><li>— разработана высокоуровневая модель данных предприятия ООО «Экспресс-доставка»;</li></ul>

*Окончание таблицы*

Название вехи	Срок	Результаты
		<ul style="list-style-type: none"> <li>— сформирован документ «Предложение по автоматизации ООО “Экспресс-доставка”»;</li> <li>— разработана модели автоматизируемых бизнес-процессов предприятия ООО «Экспресс-доставка» по типу «как должно быть»;</li> <li>— сформирован документ «Требования к разрабатываемой информационной системе»</li> </ul>
Анализ и проектирование завершены	18.04.2016	<ul style="list-style-type: none"> <li>— Спроектированы подсистемы;</li> <li>— спроектированы классы;</li> <li>— спроектированы прецеденты;</li> <li>— спроектированы БД</li> </ul>
Реализация завершена	17.05.2016	<ul style="list-style-type: none"> <li>— Модули реализованы;</li> <li>— программный код проведен;</li> <li>— модули интегрированы</li> </ul>
Тестирование завершено	27.05.2016	<ul style="list-style-type: none"> <li>— Проведено модульное тестирование;</li> <li>— проведено функциональное тестирование;</li> <li>— проведено приемочное тестирование;</li> <li>— проведено нагрузочное тестирование</li> </ul>
Развертывание и внедрение завершено	30.08.2016	<ul style="list-style-type: none"> <li>— Разработана документация;</li> <li>— ИТ-инфраструктура закуплена;</li> <li>— пользователи обучены;</li> <li>— система развернута на рабочих местах;</li> <li>— проведены приемо-сдаточные испытания</li> </ul>
Конец проекта	31.08.2016	Формальное закрытие проекта

#### **4. Организационная структура проекта и ответственность**

##### **4.1. Организационная структура**

Для реализации задач проекта сформированы рабочие группы. Состав рабочих групп и данные для контактов в проекте приведены в Приложении 1 к данному Уставу.

Проектная команда обеспечивает своевременное и качественное выполнение работ по проекту.

Проектная команда включает сотрудников Исполнителя и специалистов Заказчика, работающих вместе.

Руководство проектом осуществляют руководитель проекта со стороны Заказчика и руководитель проекта со стороны Исполнителя.

За работой проектной команды наблюдают Кураторы проекта со стороны Заказчика и Исполнителя.

Со стороны Заказчика и Исполнителя формируется проектная группа из специалистов с необходимыми компетенциями, участие которых обеспечит успешное выполнение проекта.

## 4.2. Участники проекта и их ответственность

Название проектной структурной единицы / роли	Описание функций и ответственности
Кураторы проекта	<p>Функции Кураторов проекта:</p> <ul style="list-style-type: none"> <li>— продвижение проекта — обеспечение его успешного осуществления;</li> <li>— регулярный контроль над ходом проекта;</li> <li>— решение стратегических вопросов, утверждение основных изменений в объеме работ, сроках, этапах и в бюджете проекта</li> </ul>
Руководители проекта	<p>Ответственность Руководителя проекта со стороны Заказчика:</p> <ul style="list-style-type: none"> <li>— контроль выполнения работ в рамках согласованных сроков, бюджета и ресурсов;</li> <li>— информирование Куратора от Заказчика о ходе проекта, информирование других заинтересованных лиц со стороны Заказчика о целях и ходе выполнения работ по проекту;</li> <li>— обеспечение участия необходимого персонала Заказчика для выполнения работ проекта;</li> <li>— выявление и направление проблем и рисков проекта на уровень Куратора от Заказчика.</li> </ul>
	<p>Ответственность Руководителя проекта со стороны Исполнителя:</p> <ul style="list-style-type: none"> <li>— планирование и организация работ;</li> <li>— информирование Куратора и других заинтересованных лиц со стороны Заказчика о ходе проекта;</li> <li>— выявление рисков, проблем и информирование кураторов проекта со стороны Исполнителя и Заказчика;</li> <li>— приемка и согласование проектных документов, разработанных консультантом;</li> <li>— передача результирующих документов Заказчику;</li> <li>— обеспечение выполнения работ в рамках согласованных сроков, бюджета и ресурсов;</li> <li>— контроль объема выполняемых работ и обеспечение его соответствия контрактным обязательствам.</li> </ul> <p><b>Руководители проектов отвечают за итоги проекта в целом.</b></p> <p>В оперативном подчинении у руководителей проектов находятся другие участники проектной команды</p>
Специалисты проектной группы Заказчика	<p>Ответственность Специалистов проектной группы Заказчика:</p> <ul style="list-style-type: none"> <li>— своевременное предоставление запрашиваемой Исполнителем информации и материалов (в рамках своей компетенции), необходимых для подготовки результатов по проекту;</li> </ul>

Название проектной структурной единицы / роли	Описание функций и ответственности
	— своевременное рассмотрение, обсуждение и выдача замечаний по промежуточным и итоговым результатам по проекту
Консультанты проектной группы Исполнителя	<p>Ответственность Специалистов проектной группы:</p> <ul style="list-style-type: none"> <li>— определение, сбор, анализ необходимой информации для разработки методических материалов и электронной модели;</li> <li>— разработка в рамках своей компетенции, рабочих, промежуточных итоговых материалов по проекту;</li> <li>— анализ, обсуждение со специалистами Заказчика и отработка замечаний по промежуточным и итоговым материалам проекта</li> </ul>

## 5. Процедуры управления проектом

В данном разделе приведены порядок и требования, связанные с управлением проектом, направленные на эффективное выполнение проекта.

### 5.1. Управление коммуникациями

Средством коммуникации всех членов проектной команды является электронная почта. Посредством электронной почты осуществляется обмен основными документами проекта, планами, заданиями, протоколами и прочей проектной документацией, а также запросами на получение информации и ответами на нее. В качестве документов, подтверждающих общее понимание обсужденных вопросов, могут использоваться протоколы, подписываемые Заказчиком и Исполнителем на уровне Кураторов проекта, Руководителей проекта.

Документы передаются Заказчику только Руководителем проекта со стороны Исполнителя.

Руководитель проекта со стороны Заказчика передает замечания к документам Руководителю проекта со стороны Исполнителя.

#### 5.1.1. Планирование и порядок проведения совещаний

Для решения вопросов, возникающих в ходе проекта, проводятся совещания. Совещания могут проходить как в очной форме, так и по телефону.

Перед совещанием должны быть определены цель, повестка, состав участников, необходимые материалы.

Место и время совещания должны быть согласованы с участниками (с руководителями проектов, кураторами и другими участниками, не входящими в проектную команду). Материалы, требующие

изучения перед совещанием, должны быть разосланы участникам заранее с учетом времени на изучение.

Если на совещании планируется принять некоторое решение, сторонами должны быть проработаны варианты решений и доведены до другой стороны.

По результатам совещания готовится Протокол, в котором фиксируются принятые решения.

Протокол рассыпается участникам для согласования в течение восьми рабочих часов.

Порядок согласования отчета о встрече определяется в рабочем порядке.

Протокол согласовывается по электронной почте (без подписей бумажной копии документа).

Допускается «автосогласование» отчета: отчет считается согласованным при отсутствии ответного письма в течение двух дней.

### **5.1.2. Информирование о ходе проекта**

Отчетностью по ведению проекта являются еженедельные статус-отчеты проекта и актуальный план проекта с указанием фактического состояния работ на текущую дату.

Статус-отчет проекта готовится руководителем проекта со стороны Исполнителя и направляется руководителю проекта со стороны Заказчика (копия — заместителям руководителя проекта) по электронной почте каждый понедельник.

Шаблон статус-отчета приведен в Приложении 2.

### **5.1.3. Порядок решения проблем, рисков**

Любой участник проектной команды может поднять проблему, возникшую в ходе проекта. Проблема направляется вышестоящему руководителю при невозможности решить ее на своем уровне. При наличии проблемы участник проекта информирует руководителя проекта о наличии проблем, на уровне руководителя проекта — руководитель проекта информирует куратора и РП другой стороны о наличии проблемы.

При направлении проблемы вышестоящему руководителю инициатор должен подробно описать проблему, описать действия, которые предпринимались для ее решения, и по возможности предложить к обсуждению вариант решения проблемы.

Шаблон регистрации проблем, рисков — Приложение 3.

## **6. Процедуры приемки-сдачи**

Передача отчетных материалов и документов Заказчику осуществляется по мере их готовности в соответствии с Планом проекта.

Согласование документов осуществляется итерационно: для каждого документа сначала согласовывается шаблон и структура документа, затем на согласование Заказчику направляется первый драфт документа, и после устранения замечаний осуществляется согласование и утверждение итогового документа.

В целом на согласование каждого документа Заказчику отводится четыре дня, из них:

- один день — согласование шаблона и структуры документа;
- один день — согласование первого драфта;
- один день — согласование итогового документа, с устранимыми замечаниями;
- один день — утверждение документа.

Замечания к документу вносятся по тексту в режиме правки и (или) заносятся в Журнал замечаний.

Замечания к документам и материалам, выявленные Заказчиком, устраняются Консультантом в срок до пяти дней.

Замечания со стороны Заказчика по новым версиям отчетных материалов, исправленных Консультантом и предоставленных Заказчику для повторного рассмотрения, могут уточнять замечания, сделанные Заказчиком ранее, но не могут содержать новые критические замечания по сравнению с более ранними замечаниями.

#### **Список лиц, согласующих проектные материалы и документы**

№	Наименование документа	ФИО согласующего лица
1	Модель основных бизнес-процессов ООО «Экспресс-доставка» по типу «как есть»	Николаев Н. Н.
2	Функциональная модель предприятия ООО «Экспресс-доставка»	Николаев Н. Н.
3	Высокоуровневая модель данных предприятия ООО «Экспресс-доставка»	Петров П. П.
4	Предложение по автоматизации ООО «Экспресс-доставка»	Николаев Н. Н.
5	Модель автоматизируемых бизнес-процессов ООО «Экспресс-доставка» по типу «как должно быть»	Петров П. П.
6	Требования к разрабатываемой информационной системе	Петров П. П.
7	Результаты анализа и проектирования информационной системы	Николаев Н. Н.
8	Результаты тестирования информационной системы	Петров П. П.

#### **7. Оценка затрат на проект**

№	Статья затрат	Финансирование, руб.
Расходы на оборудование		
1	Серверы	70 000

*Окончание таблицы*

Nº	Статья затрат	Финансирование, руб.
2	Планшеты	50 000
3	Рабочие места	40 000
4	Сетевое оборудование	50 000
	<b>Итого:</b>	<b>210 000</b>
<b>Расходы на этапы проекта</b>		
1	Формирование проектной команды	102 400
2	Проведение структурного анализа	652 800
3	Проектирование и реализация	435 200
4	Тестирование	89 600
5	Развертывание и внедрение	1 230 800
	<b>Итого:</b>	<b>2 510 800</b>

Налоги выплачиваются согласно законодательству РФ.

Источники финансирования проекта: финансирование за счет средств заказчика.

Расчет финансовых издержек: по договоренности.

Схема и организация финансирования: по договоренности.

## Приложение 1 к Уставу. Состав рабочих групп

### 1.1. Кураторы проекта

№	ФИО	Роль в проекте	Контактные данные
1	Петров П. П.	Куратор со стороны Заказчика	Petrov-pp@mail.ru
2	Сергеев С. С.	Куратор со стороны Исполнителя	Sergeev-ss@mail.ru

### 1.2. Рабочая группа от Исполнителя

№	ФИО	Роль в проекте	Контактные данные
1	Иванов И. И.	Руководитель проекта со стороны Исполнителя	Ivanov-ii@mail.ru
2	Петренко Н. С.	Консультант	Petrenko-ns@mail.ru
3	Суханов С. С.	Консультант	Suhanov-ss@mail.ru
4	Архипов А. А.	Консультант	Arkipov-aa@mail.ru
5	Немаев Н. Н.	Консультант	Nemaev-nn@mail.ru

### 1.3. Рабочая группа от Заказчика

№	ФИО	Роль в проекте	Контактные данные
1	Николаев Н. Н.	Руководитель проекта со стороны Заказчика	Nikolaev-nn@mail.ru
2	Степанов А. Н.	Бизнес-аналитик Специалист по обучению Специалист по сопровождению	Stepanov-an@mail.ru
3	Мираев М. Н.	Проектировщик БД Программист	Miraev-mn@mail.ru
4	Кролов К. К.	Бизнес-аналитик Системный аналитик Системный архитектор	Krolov-kk@mail.ru
5	Некрасов Н. Н.	Программист Тестировщик	Nekrasov-nn@mail.ru
6	Невзоров Н. П.	Программист Технический писатель Системный администратор	Nevzorov-np@mail.ru

## Приложение 2 к контрольному заданию. Реестр рисков

Таблица 8.12

### Пример Реестра рисков

№	Определение риска	Оценка ущерба, руб.	Вероятность реализации	Стратегия минимизации
1	Увеличение цен при закупке ИТ-инфраструктуры	100 000	0,7	Найти вендора с более выгодным предложением
2	Возникновение ошибок в проектных разработках и документации	150 000	0,5	Выделить роль проверяющего на каждом этапе проекта
3	Приобретение недостаточно надежного оборудования	150 000	0,2	Провести анализ предложений на предмет поиска оптимального варианта оборудования
4	Изменение законодательства, регулирующего информационный обмен в компаниях доставки	100 000	0,1	Использовать стратегию принятия риска
5	Ошибки работы веб-интерфейса на различных браузерах клиентов	50 000	0,5	Увеличить бюджет и сроки на тестирование
6	Потребность в изменениях на финальных стадиях проекта	300 000	0,5	Увеличить время и бюджет на идентификацию требований. Увеличить число совместных собраний с представителями команды Заказчика

## Приложение 3 к контрольному заданию. Форма регистрации риска

*Таблица 8.13*

### Пример Формы регистрации риска

Форма регистрации риска			
Номер в реестре рисков: 1			
ФИО автора: Иванов И. И. Роль: Руководитель проекта  Фаза проекта: Разворачивание и внедрение	Приоритет: Высокий Дата запроса: 25.01.2016 Желаемая дата разрешения: до 30.05.2016		
<p>Описание риска: цены на объекты ИТ-инфраструктуры могут серьезно увеличиться из-за колебаний курса валюты или из-за изменения цен поставщиков.</p> <p>Предпринятые действия: по состоянию на 25.01.2016 действия по минимизации риска не предпринимались.</p> <p>Дата окончания действия риска: 03.06.2016</p>			
<p>Предпосылки:</p> <p>Вендор, у которого ранее закупалась аппаратура по специальной цене со скидкой, больше не поставляет сервера HP Proliant DL360, которые планировалось закупать изначально. Аналогичные предложения других вендоров в настоящий момент не содержат скидок, и итоговая цена существенно выше. Серверы-аналоги от постоянного вендора также обладают более высокой ценой.</p>			
<p>Варианты решения:</p> <p>Необходимо либо найти вендора, готового предложить скидку при закупке элементов ИТ-инфраструктуры, либо найти продукты-аналоги по сопоставимой цене.</p>			
Статус:	Дата	Комментарии к статусу	
Принят к рассмотрению	25.01.2016		
Включен в реестр рисков	26.01.2016		
Результаты анализа риска			
Вероятность	Влияние	Степень угрозы	Стратегия
0,7	100 000 руб.	Высокая	Избежать
<p>Обоснование выбора стратегии:</p> <p>Стратегия избежания позволит существенно снизить вероятность возникновения риска, при этом издержки на нее невысоки.</p>			
<p>Описание предложений по реализации стратегии:</p> <ul style="list-style-type: none"><li>— Произвести анализ рынка на предмет аналогов исходного оборудования.</li><li>— Провести поиск вендора с наиболее выгодным коммерческим предложением.</li></ul>			
Ответственный за риск: Сипягин А. В.			

Утвержденный вариант решения по минимизации риска:

— Провести поиск вендора с наиболее выгодным коммерческим предложением

Действие	Ответственный	Срок	Статус
1. Сформировать список потенциальных вендоров	Сипягин А. В.	31.03.2016	Выполнено
2. Запросить коммерческие предложения	Сипягин А. В.	05.04.2016	Выполнено
3. Провести анализ коммерческих предложений	Иванов И. И., Сипягин А. В.	15.04.2016	Выполнено
4. Начать заключение договора на закупку оборудования	Иванов И. И.	01.05.2016	Исполняется

Поля, отмеченные серым цветом, заполняются группой управления проектом или руководителем проекта. Поля, не имеющие цветового выделения, заполняются членом проектной команды (со стороны Заказчика или Исполнителя) и передаются на рассмотрение группе управления проектом или руководителю проекта со стороны Исполнителя.

## Приложение 4 к контрольному заданию

### ПЛАН ТЕСТИРОВАНИЯ

#### **1. Общая информация**

##### **1.1. Цель документа**

В качестве цели документа «План тестирования» выступает тестирование ИС для компании «Экспресс-доставка». Этот документ должен содержать описание мероприятий по тестированию системы, сроки их выполнения, ответственных, виды тестирования, перечень документации.

##### **1.2. Предмет и цели тестирования**

Цель тестирования в данном случае — это проверка удовлетворения функциональных и нефункциональных требований к ИС со стороны Заказчика.

Предмет тестирования — это информационная система для компании «Экспресс-доставка», которая предназначена для хранения, поиска, обработки информации о сборе отправлений. ИС выполнена в виде клиент-серверного приложения.

##### **1.3. Основные этапы тестирования**

В рамках этапа тестирования для данного примера будут проводиться следующие этапы.

- Модульное тестирование для проверки компонентов ИС на соответствие функциональным требованиям. С этой целью в системе будут выделены компоненты и интерфейсы для тестирования.
  - На этапе подготовки к тестированию будут созданы тест-кейсы.
  - На этапе проведения тестирования созданные тест-кейсы будут реализовываться.
- Регистрация ошибок в случае расхождения ожидаемого и фактического результата.
- Функциональное тестирование будет проводиться с целью контроля качества.
  - Будет проведен UAT (User Acceptance Test) для контроля качества ИС конечными пользователями.
  - Нагрузочное тестирование будет проводиться для определения производительности ИС, проверки надежности и отказоустойчивости, а также для выявления узких мест.
  - По итогам нагружочного тестирования может быть принято решение о доработке программного продукта с целью увеличения производительности.
  - Важно, что доработка функциональности после данного вида тестирования не предполагается.

##### **1.4. Требования для начала тестирования**

Для начала тестирования ИС для компании ООО «Экспресс-доставка» должна позволять выполнять следующие действия:

- создавать, редактировать и удалять информацию об отправлениях;

- формировать, редактировать, удалять, выводить на печать реестры с отправлениями;
- изменять статус отправления;
- хранить информацию об отправлении.

### 1.5. Стратегия тестирования

Для модульного, функционального и UAT-тестирования воспользуемся стратегией «черного ящика». Будут использованы методы эквивалентного разбиения, анализа граничных значений и метод предположений об ошибке.

С учетом предполагаемого расширения филиальной сети компании ООО «Экспресс-доставка» до 100 филиалов необходимо провести нагружочное тестирование с допущением подобного увеличения нагрузки.

## 2. Этапы тестирования

### 2.1. Общая информация об этапах тестирования

Таблица 2.1

**Критерии начала и окончания этапов тестирования**

Фаза этапа	Критерии
Критерии начала этапа тестирования	<p>Smoke-тесты для проверки работоспособности среды (в случае, если речь идет о модульном тестировании) успешно пройдены.</p> <p>Вся реализованная функциональность прошла предыдущий этап тестирования (для всех этапов, кроме модульного тестирования).</p> <p>План тестирования утвержден руководителями проекта со стороны Заказчика и Исполнителя.</p> <p>Все аппаратные средства тестирования и среда тестирования доступны для проведения тестирования</p>
Критерии окончания этапа тестирования	<p>Все разработанные тест-кейсы должны быть использованы.</p> <p>Все ошибки должны быть зарегистрированы в Протоколе тестирования.</p> <p>Все ошибки должны быть исправлены в соответствии с таблицей соответствия степени критичности ошибки и критерием приемки.</p> <p>Протокол тестирования должен быть составлен и согласован руководителями проекта с обеих сторон</p>

Таблица 2.2

**Классификация ошибок и критерии приемки**

Степень критичности ошибки	Описание	Критерий приемки
Критическая	Ошибка вызывает такой отказ ПО или такую потерю данных, которые не позволяют восстановить работоспособ-	Нет критических ошибок

Степень критичности ошибки	Описание	Критерий приемки
	ность ПО или восстановить данные. Ошибка серьезно нарушает базовое функционирование модуля и требует незамедлительного исправления. Ошибка делает невозможным проведение дальнейшего тестирования	
Высокая	Основные функции ИС невозможно выполнить, но есть альтернативные варианты выполнения данных функций	Нет ошибок высокой степени
Средняя	Ошибки приводят к нежелательным результатам или нежелательному функционированию компонента, но не мешают тестировщику продолжать проведение тестирования	Устранено не менее 80% средних и незначительных ошибок (от общего количества ошибок в Протоколе тестирования с данной степенью критичности)
Незначительная	Незначительные проблемы, требующие минимальных изменений, которые не влияют на функциональные аспекты работы ПО	

## 2.2. Модульное тестирование

Модульное тестирование осуществляется по мере реализации компонентов ИС.

Тестирование проводится тестировщиками по согласованным ранее Тестовым сценариям.

В Протокол тестирования заносятся все обнаруженные ошибки (со ссылкой на тестовый сценарий).

Протокол тестирования рассыпается всем ответственным лицам после проведения тестирования.

После проведения модульного тестирования Заказчик принимает решение об успехе или неудаче при прохождении модульного тестирования.

Заказчик принимает решение о начале следующего этапа тестирования (функционального тестирования).

## 2.3. Функциональное тестирование

Функциональное тестирование проводится тестировщиками по согласованным тестовым сценариям в тестовой среде.

Сотрудники компании ООО «Экспресс-доставка» могут принимать участие в проведении функционального тестирования.

В Протокол тестирования заносятся все обнаруженные ошибки (со ссылкой на тестовый сценарий).

Протокол тестирования рассыпается всем ответственным лицам после проведения тестирования.

После проведения модульного тестирования Заказчик принимает решение об успехе или неудаче при прохождении модульного тестирования.

Заказчик принимает решение о начале следующего этапа тестирования (приемочное тестирование).

#### **2.4. Приемочное тестирование (UAT)**

Приемочное тестирование проводится сотрудниками Заказчика по согласованным тестовым сценариям в соответствии с графиком работ.

Описание итогов по обнаруженным ошибкам в день тестирования отправляется менеджеру по тестированию (копии письма направляются всем участникам рабочей группы).

Менеджер тестирования проверяет корректность описания ошибки и заносит ее в Протокол тестирования (с учетом связи с тестовым сценарием).

Если найденные ошибки делают невозможным тестирование компонента, то производится переход к тестированию другого компонента.

После исправления и внутреннего тестирования ошибка переводится на ответственного сотрудника для подтверждения корректности исправления.

#### **2.5. Нагрузочное тестирование**

Цели нагрузочного тестирования:

- проверить соответствие производительности ПО предъявляемым требованиям;
- проверить работоспособность ПО в условиях предполагаемой нагрузки;
- выявить возможные проблемы функционирования ПО под нагрузкой.

#### **Профили нагрузки**

Ожидается расширение сети филиалов компании, вместе с чем возрастет и общее число пользователей ПО. Соответственно, в начале будет проведено нагрузочное тестирование на предмет соответствия ИС текущим условиям нагрузки (текущему числу филиалов). Далее будет проведено нагрузочное тестирование с предполагаемым увеличением числа пользователей ПО.

Время нагрузки: 12 ч.

Каждая операция будет моделироваться отдельным сценарием нагрузочного тестирования, который будет выполняться отдельной группой виртуальных пользователей.

Все группы виртуальных пользователей будут работать независимо друг от друга, поэтому все операции будут выполняться одновременно и независимо друг от друга.

Интенсивность выполнения операций в течение теста не изменяется, т. е. производительность операций равномерно распределена

по тесту. При этом на втором этапе число виртуальных пользователей увеличивается.

Таблица 2.3

**Профили нагрузки**

№	Название операции	Интенсивность выполнения операции каждым виртуальным пользователем в час	Количество виртуальных пользователей в группе	
			Этап 1	Этап 2
1	Занесение информации о новом отправлении в базу	12	20	200
2	Сортировка новых отправлений	12	20	200

**Критерии успешности и метрики**

Нагрузочное тестирование считается успешно пройденным в случае, если работоспособность системы не меняется под нагрузкой (ПО не блокируется, время отклика на запросы не превышает 1 с).

В ходе нагрузочного тестирования сохраняются следующие метрики:

- потребление ресурсов центрального процессора;
- потребление оперативной памяти;
- пропускная способность сетевых ресурсов;
- пропускная способность сервера;
- время отклика сервера;
- время ожидания ввода/вывода;
- время выполнения запроса;
- время генерации страницы для одного пользователя;
- число ошибок при выполнении тестов, вызванных нагрузкой на ИС.

Метрики, а также результаты их анализа, заносятся в Протокол тестирования вместе с рекомендациями по устраниению или минимизации.

**3. Ресурсы тестирования**

**3.1. Роли и обязанности в тестировании**

Таблица 3.1

**Роли участников тестирования со стороны Исполнителя**

Роль	Обязанности
Руководитель проекта	Общий контроль за ходом тестирования. Координация работ по проведению тестирования со стороны Исполнителя. Формирование Плана тестирования. Составление Протоколов тестирования и Протоколов ошибок

Роль	Обязанности
Тестировщик	Разработка тестовых сценариев. Проведение тестирования по утвержденным тестовым сценариям (по графику). Ведение ошибок и поддержка их в актуальном состоянии. Проверка корректности исправления ошибок
Системный аналитик	Проведение анализа ошибок. Внесение исправлений в документацию
Программист	Исправление ошибок

Таблица 3.2

**Роли участников тестирования со стороны Заказчика**

Роль	Обязанности
Руководитель проекта	Общий контроль за ходом тестирования. Координация работ по проведению тестирования со стороны Заказчика. Организация и поддержка тестовой среды
Конечные пользователи ИС	Проведение UAT по тестовым сценариям. Обсуждение с сотрудниками Исполнителя обнаруженных ошибок
Системный администратор	Разворачивание тестовых стендов. Загрузка данных на тестовые стойки. Поддержание базы данных

**3.2. Тестовая среда**

Проведение тестирования производится в тестовой среде. Серверная часть включает:

- Windows Server 2013;
- сервер БД MySQL.

На момент проведения тестирования база данных должна включать всю информацию, имеющуюся на дату, предшествующую дате тестирования.

**4. Документация тестирования**

В результате проведения тестирования требуется сформировать документы, перечисленные в следующей таблице.

Таблица 4.1

**Документы тестирования**

Документ	Описание
Протокол модульного тестирования	Составляется по результатам модульного тестирования. Включает описание всех заранее утвержденных и проведенных тест-кейсов.

Документ	Описание
	Включает все обнаруженные ошибки в виде ссылки на Протокол ошибок. Составляется после каждого модульного тестирования
Протокол функционального тестирования	Составляется по результатам функционального тестирования. Включает описание всех заранее утвержденных и проведенных тест-кейсов. Включает все обнаруженные ошибки в виде ссылки на Протокол ошибок. Составляется после каждого функционального тестирования
Протокол приемочного тестирования	Составляется по результатам приемочного тестирования. Включает описание всех заранее утвержденных и проведенных тест-кейсов. Включает все обнаруженные ошибки в виде ссылки на Протокол ошибок. Составляется после каждого приемочного тестирования
Протокол ошибок	Составляется на каждую тестовую дату в рамках каждого этапа тестирования
Протокол нагрузочного тестирования	Составляется по результатам нагрузочного тестирования. Содержит выявленные ошибки и несоответствие требованиям. Содержит значения оцениваемых метрик
План-график работ	График прохождения этапов тестирования. Основан на Плане проекта и Иерархической структуре работ

### **8.3. Дополнительное задание по кейсу: разработка проектного решения системы мониторинга отправлений, курьеров и транспортных средств**

#### **8.3.1. Обследование мониторинга деятельности (As-Is)**

На основании общего описания деятельности компании-заказчика и выделенных при старте проекта целей и задач **составить список вопросов по мониторингу деятельности к интервью с руководством компании и отдельных подразделений.**

- Использовать полученные на предыдущем этапе модели бизнес-процессов основной деятельности As-Is, построить недостающие модели, указать на них ответственные подразделения, входящие/исходящие данные и автоматизирующие их приложения.

- Оценить степень покрытия процессов мониторинга деятельности текущим ландшафтом ИТ-систем.
- В соответствии с моделью зрелости CMMI определить текущий уровень зрелости выбранных процессов и с учетом бизнес-целей предприятия предложить для них целевой уровень зрелости и необходимые для этого мероприятия.
- Провести анализ организационной структуры и бизнес-процессов в области мониторинга деятельности (в том числе опираясь на информацию из отзывов клиентов на сайте). Составить список выделенных проблем.
- На основе выделенных проблем идентифицировать риски, их последствия и возможности снижения рисков.

### **8.3.2. Разработка модели (To-Be)**

- На основании анализа в Интернете предложений по использованию систем электронного мониторинга выделить лучшие практики для автоматизации процесса мониторинга в компании.
- Проанализировать эффективность процессов мониторинга в компании и предложить меры по их развитию с помощью информационных технологий (в первую очередь — RFID и другие технологические решения).
- Обосновать потребность в реинжиниринге бизнес-процессов и предоставить модель «To-Be» (включая диаграмму потоков данных между приложениями и модулями).
- Определить целевую функциональность системы электронного мониторинга.
- Разработать целевую модель ИТ-инфраструктуры в рамках проекта внедрения современной системы мониторинга.

### **8.3.3. Разработка модели перехода к целевой модели (Roadmap)**

- Проанализировать по сформированным требованиям различные ИТ-решения на рынке и выбрать оптимальное для задач проекта.
- Сформировать требования к новой системе мониторинга на основе FURPS+.
- Предложить «дорожную карту» проекта внедрения, обеспечивающего переход от модели процессов «As-Is» к модели «To-Be» (описать необходимые активности, этап работ, ресурсы, определить ценность для заинтересованных сторон). В случае необходимости сделать необходимые предположения.
- Для реализации в перспективе комплексного проекта по развитию системы мониторинга определить первоочередные шаги (например, доработка отдельных модулей эксплуатирующихся си-

стем, оснащение определенными программными/аппаратными средствами и пр.).

- Используя MS Project, составить поэтапный план внедрения выбранного ИТ-решения с указанием ресурсов, длительности и стоимости.

## **Нормативные документы**

1. ГОСТ 34.601—90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.
2. ISO/IEC 12207:2008 Information technology — Software life cycle processes (Информационные технологии. Процессы жизненного цикла программного обеспечения).
3. ГОСТ Р ИСО/МЭК 12207—2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.
4. ISO/IEC 15288 Systems engineering. System life cycle processes (Системотехника. Процессы жизненного цикла системы).
5. ГОСТ Р ИСО/МЭК 15288—2005 Информационная технология. Системная инженерия. Процессы жизненного цикла систем.
6. ISO/IEC 19510:2013 Информационные технологии. Модель и нотация процесса менеджмента объекта в групповом бизнесе.
7. ISO/IEC 19505:2012. Информационные технологии. Унифицированный язык моделирования группы по управлению объектами (OMG UML).
8. ГОСТ 19.201—78 Техническое задание, требования к содержанию и оформлению.
9. ГОСТ 34.602—89 Техническое задание на создание автоматизированной системы.
10. ГОСТ 34.201—89 Виды, комплектность и обозначение документов при создании автоматизированных систем.
11. ГОСТ 12119 ИТ. Пакеты программ. Требования к качеству и тестирование.
12. ГОСТ 12207 ИТ. Процессы жизненного цикла программных средств.
13. ГОСТ 14764 ИТ. Сопровождение программных средств.
14. ГОСТ 15504 ИТ. Оценка процессов.
15. ГОСТ 9126 ИТ. Оценка программной продукции. Характеристики качества и руководства по их применению.
16. ГОСТ серии 19. Единая система программной документации (ЕСПД).
17. IEEE 829 Стандарт документирования тестирования программного обеспечения и систем.
18. IEEE 1219 Standard for Software Maintenance.

19. CobiT (Control Objectives for Information and Related Technology).
20. ITIL (IT Infrastructure Library).
21. SWEBOK (Guide to the Software Engineering Body of Knowledge).
22. PMBOK (Project Management Body of Knowledge).
23. PRINCE2 (PRojects IN Controlled Environments).
24. ISO 21500:2012 Guidance on project management.
25. ГОСТ Р ИСО 21500—2014. Международный стандарт по Управлению Проектами.
26. ГОСТ Р 54869—2011 Проектный менеджмент. Требования к управлению проектом.
27. ГОСТ Р 54871—2011 Требования к управлению программой.
28. ГОСТ Р 54870—2011 Требования к управлению портфелем проектов.
29. ISO 10006:2003 Quality management systems — Guidelines for quality management in projects.
30. ГОСТ Р ИСО 10006—2005 Системы менеджмента качества. Руководство по менеджменту качества при проектировании.
31. ISO 8402:94 Управление качеством и обеспечение качества.
32. ISO 9000:2005 Системы менеджмента качества. Основные положения и словарь.
33. ISO 9001:2008 Системы менеджмента качества. Требования.
34. ISO 9004:2009 Менеджмент с целью достижения устойчивого успеха организации. Подход с позиции менеджмента качества.
35. ISO 10001:2007 Менеджмент качества. Удовлетворенность потребителя. Руководящие указания по кодексу поведения для организаций.
36. ISO 10005:2005 Системы менеджмента качества. Руководящие указания по планам качества.
37. ISO 10006:2005 Системы менеджмента качества. Руководящие указания по менеджменту качества проектов.
38. ISO 10007:2003 Системы менеджмента качества. Руководящие указания по менеджменту конфигурации.
39. ISO/TR 10013:2001 Рекомендации по документированию систем менеджмента качества.
40. ISO/TR 10014:2006 Менеджмент качества. Руководящие указания по реализации финансовых и экономических выгод.
41. ISO 10015:1999 Управление качеством. Руководящие указания по обучению.
42. ISO 19011:2011 Руководящие указания по аудиту систем менеджмента.
43. ISO 10006:2003 Управление качеством. Руководящие указания по менеджменту качества проектов.
44. ГОСТ Р ИСО 10006—2005 Системы менеджмента качества. Руководство по менеджменту качества при проектировании.

45. ISO 15504:2004 Information Techology. Process Assessment / SPICE (Software Process Improvement and Capability Determination).
46. ГОСТ Р ИСО/МЭК 15504—2012. Информационные технологии. Оценка процессов.
47. Contingency Planning Guide for Information Technology Systems (NIST).
48. Ананьевин, В., Зимин, К. Ценность ИТ. Теория и практика // CIO congress Kirov. 28.04.2012.
49. Блиннов, Д. Требования к системе: классификация FURPS+ // Портал BeamTeam.ru, 2010. URL: <http://beamteam.ru/2010/09/furps/>.
50. URL: <http://www.swell-services.be/cmsms/index.php?page=it-system-decommissioning>.
51. URL: [http://www.intuit.ru/studies/courses/2196/267/print\\_lecture/6796](http://www.intuit.ru/studies/courses/2196/267/print_lecture/6796).
52. URL: <http://tsconsulting.ru/upload/iblock/37b/37b27c616e326793c95eb98b325a0748.pdf>.
53. URL: [http://www.pqm-online.com/assets/files/standards/gost\\_r\\_iso\\_10006-2005.pdf](http://www.pqm-online.com/assets/files/standards/gost_r_iso_10006-2005.pdf).

**Наши книги можно приобрести:**

**Учебным заведениям и библиотекам:**

в отделе по работе с вузами

тел.: (495) 744-00-12, e-mail: vuz@urait.ru

**Частным лицам:**

список магазиновсмотрите на сайте urait.ru

в разделе «Частным лицам»

**Магазинам и корпоративным клиентам:**

в отделе продаж

тел.: (495) 744-00-12, e-mail: sales@urait.ru

**Отзывы об издании присылайте в редакцию**

e-mail: gred@urait.ru

**Новые издания и дополнительные материалы доступны  
на образовательной платформе «Юрайт» urait.ru,  
а также в мобильном приложении «Юрайт.Библиотека»**

*Учебное издание*

**Зараменских Евгений Петрович**

# **УПРАВЛЕНИЕ ЖИЗНЕННЫМ ЦИКЛОМ ИНФОРМАЦИОННЫХ СИСТЕМ**

Учебник и практикум для вузов

Формат 70×100<sup>1</sup>/16.

Гарнитура «Charter». Печать цифровая.

Усл. печ. л. 38,56.

**ООО «Издательство Юрайт»**

111123, г. Москва, ул. Плеханова, д. 4а.

Тел.: (495) 744-00-12. E-mail: izdat@urait.ru, www.urait.ru