

Библиотека компании «Кейсистемс»



А. Перерва, В. Иванова

# ПУТЬ АНАЛИТИКА

Практическое руководство  
IT-специалиста

 **КЕЙСИСТЕМС** информационные технологии

 **ПИТЕР**®

*В. Иванова, А. Перерва*

## Путь аналитика. Практическое руководство IT-специалиста

2-е издание

Заведующий редакцией	<i>С. Клебанов</i>
Ведущий редактор	<i>Н. Гринчик</i>
Литературный редактор	<i>И. Говорун</i>
Художник	<i>В. Шимкевич</i>
Корректоры	<i>О. Андриевич, Е. Павлович</i>
Верстка	<i>А. Барцевич</i>

ББК 32.973.2-018-02

УДК 004.414

**Иванова В., Перерва А.**

И21 Путь аналитика. Практическое руководство IT-специалиста. 2-е изд. — СПб.: Питер, 2015. — 304 с.: ил.

ISBN 978-5-496-01679-7

Перед вами настольная книга для системных аналитиков, программистов, архитекторов программного обеспечения, менеджеров проектов и начальников отделов по разработке программ. Кроме того, книга будет полезным учебным пособием для преподавателей, студентов и аспирантов кафедр IT в технических вузах.

Как воплотить неясные ожидания заказчика в блестящий и прибыльный проект? Как избежать ошибок на начальном этапе? Как стать эффективным аналитиком?

Авторы отвечают на эти вопросы и делятся своими ноу-хау, которые позволят вам стать гуру в разработке программного обеспечения. Главное достоинство книги — ее практическая направленность. В ней собрана полезная информация со ссылками на теоретические материалы из разных областей разработки программного обеспечения: анализа, архитектуры, управления проектами, лидерства и управления персоналом — все, что понадобится в реальных производственных проектах.

Помимо этого в книге содержится анализ разнообразных кейсов и ситуаций, а также примеры документов и шаблонов, необходимых для разработки ПО. Авторы структурируют огромный массив теоретической информации исходя из ее практической ценности на каждом этапе профессиональной карьеры. Книга написана простым и доступным языком.

Авторы многие годы шли к высшему уровню профессионализма, а вас отделяет от тех же знаний только прочтение этой книги.

**6+** (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ISBN 978-5-496-01679-7

© ООО Издательство «Питер», 2015

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

ООО «Питер Пресс», 192102, Санкт-Петербург, ул. Андреевская (д. Волкова), 3, литер А, пом. 7Н.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12.000 —

Книги печатные профессиональные, технические и научные.

Подписано в печать 26.03.15. Формат 70×100/16. Усл. п. л. 24,510. Тираж 1550. Заказ 0000.

Отпечатано в ОАО «Первая Образцовая типография». Филиал «Чеховский Печатный Двор».

142300, Московская область, г. Чехов, ул. Полиграфистов, 1.

Сайт: [www.chpk.ru](http://www.chpk.ru). E-mail: [marketing@chpk.ru](mailto:marketing@chpk.ru) факс: 8(496) 726-54-10, телефон: (495) 988-63-87

# Содержание

<b>Предисловие автора</b> .....	6
<b>Об авторах</b> .....	8
<b>Благодарности</b> .....	10
<b>Введение</b> .....	12
Предыстория .....	12
Для кого эта книга .....	13
Чего вы НЕ найдете в этой книге .....	14
Как читать эту книгу .....	14
<b>1. Общие понятия</b> .....	15
1.1. Что такое информатизация бизнеса и зачем мне надо это знать? .....	15
1.2. Что такое корпоративное лидерство? .....	17
1.3. Что такое архитектура, какой она бывает и зачем мне надо это знать? .....	18
1.4. Что такое требование? .....	20
1.5. Почему надо знать теории и методологии разработки ПО? .....	21
1.6. Почему бизнес- и системный анализ объединены в одной книге? .....	24
1.7. Что, кроме дисциплин анализа, должен знать хороший аналитик? .....	24
<b>2. Профиль и квалификация аналитиков</b> .....	26
2.1. Таблица квалификационных навыков .....	30
<b>3. Младший аналитик</b> .....	34
3.1. Личностные навыки .....	34
3.1.1. Аналитический стиль мышления .....	35
3.1.2. Умение конспектировать .....	41
3.1.3. Коммуникационные способности .....	43
3.1.4. Управление эмоциональным фоном .....	45
3.1.5. Полезные мелочи .....	45

3.2. Профессиональные и специальные навыки . . . . .	46
3.2.1. Использовать специальные инструменты . . . . .	48
3.2.2. Понимать методологии разработки ПО . . . . .	49
3.2.3. Выявлять ожидания ЗЛ и управлять ими . . . . .	63
3.2.4. Эффективно взаимодействовать с командой и ЗЛ по почте. . . . .	74
3.2.5. Определять границы системы и создавать концепции . . . . .	77
3.2.6. Выделять подсистемы и определять их функции . . . . .	80
3.2.7. Выявлять пользовательские требования . . . . .	84
3.2.8. Выявлять нефункциональные требования . . . . .	85
3.2.9. Выявлять требования к пользовательскому интерфейсу . . . . .	88
3.2.10. Управлять состояниями требований. . . . .	89
3.2.11. Формировать спецификации требований . . . . .	90
3.2.12. Понимать основные принципы тестирования . . . . .	90
3.2.13. Полезные мелочи . . . . .	91
3.3. Типичные проблемы и вопросы. . . . .	92
3.4. Заключение. . . . .	94
<b>4. Аналитик. . . . .</b>	<b>97</b>
4.1. Личностные навыки . . . . .	97
4.1.1. Самомотивация . . . . .	97
4.1.2. Самоорганизация. . . . .	97
4.1.3. Не упускать важное . . . . .	100
4.1.4. Культура речи . . . . .	101
4.1.5. Корпоративная культура и этика . . . . .	101
4.1.6. Полезные мелочи . . . . .	102
4.1.7. Продолжайте... . . . .	102
4.2. Профессиональные и специальные навыки . . . . .	102
4.2.1. Проектные коммуникации. . . . .	103
4.2.2. Проверьте себя. . . . .	106
4.2.3. План управления требованиями . . . . .	107
4.2.4. Концептуальная модель типов требований . . . . .	109
4.2.5. Общие принципы управления требованиями . . . . .	117
4.2.6. Моделирование . . . . .	119
4.2.7. Соответствие типов требований и моделей . . . . .	140
4.2.8. Трассировки . . . . .	148
4.2.9. Минутка расслабления . . . . .	151
4.3. Типичные проблемы и вопросы. . . . .	154
4.4. Заключение. . . . .	156
<b>5. Старший/ведущий аналитик . . . . .</b>	<b>157</b>
5.1. Личностные навыки . . . . .	157
5.1.1. Самомотивация . . . . .	157
5.1.2. Навыки публичного выступления. . . . .	157
5.1.3. Продолжайте... . . . .	159

5.2. Профессиональные и специальные навыки . . . . .	160
5.2.1. Проведение презентаций и тренингов . . . . .	161
5.2.2. Реверс-инжиниринг требований/обратное проектирование системы . . . . .	163
5.2.3. OOM и паттерны проектирования . . . . .	168
5.2.4. Риски, требования к качеству продукта и WBS аналитика. . . . .	169
5.2.5. SDLC проекта . . . . .	177
5.2.6. SDLC продукта: важность управления требованиями и основные методики инженерии требований к продуктам. . . . .	195
5.2.7. Основы управления . . . . .	207
5.2.8. Минутка расслабления . . . . .	209
5.3. Типичные проблемы и вопросы. . . . .	211
5.4. Заключение. . . . .	212
<b>6. Начальник отдела анализа . . . . .</b>	<b>214</b>
6.1. Личностные навыки . . . . .	214
6.1.1. Самомотивация . . . . .	214
6.1.2. Лидерство . . . . .	216
6.2. Профессиональные и специальные навыки . . . . .	225
6.2.1. Командная работа . . . . .	227
6.2.2. Политика. Баланс интересов. Переговоры . . . . .	230
6.2.3. Принципы и методы принятия управленческих решений . . . . .	241
6.2.4. Корпоративная культура . . . . .	246
6.2.5. Системы управления деятельностью сотрудников. . . . .	251
6.2.6. Управление персоналом . . . . .	252
6.2.7. Процессный менеджмент. Аналитические практики . . . . .	254
6.3. Типичные проблемы и вопросы. . . . .	264
6.4. Заключение. . . . .	266
<b>7. Итак... . . . . .</b>	<b>269</b>
<b>Приложение . . . . .</b>	<b>270</b>
Запросы заинтересованного лица . . . . .	270
<b>Дополнительные материалы . . . . .</b>	<b>275</b>
<b>Использованная литература . . . . .</b>	<b>276</b>
<b>Использованные интернет-ресурсы . . . . .</b>	<b>278</b>
<b>Глоссарий . . . . .</b>	<b>279</b>
<b>Словарь терминов . . . . .</b>	<b>298</b>
<b>Послесловие . . . . .</b>	<b>304</b>

# Предисловие автора

---

---

## Андрей Перерва

Верите ли вы, что есть рецепт построения успешной карьеры в области разработки программного обеспечения? Что проект по разработке программного обеспечения может быть завершен в срок и с заранее определенным бюджетом? Что есть ноу-хау, которые позволят вам стать гуру в разработке программного обеспечения? А знаете ли вы, как и что надо сделать, чтобы этого достичь?

Давайте немного порассуждаем об этом.

«Промедление с легким делом превращает его в трудное, а промедление с трудным — в невозможное» ©.

Как часто данный лозунг срывается в области разработки программного обеспечения только лишь потому, что каждый участник команды по созданию какой-либо новой программы не обладает всеми нужными практическими знаниями и ноу-хау, которые нельзя почерпнуть из теории! Как часто менеджеры проектов просто не могут понять, о чем говорит команда, зачем все это надо делать, если требуется «просто написать программу». Да и сама команда в целом довольно часто разговаривает на разных языках, превращая проект в «вавилонскую башню», а деньги инвесторов — в неконтролируемый поток инвестиций на «не пойми что».

Серьезные компании тратят огромные деньги на разработку собственных сводов знаний и правил в области создания программ, чтобы дело разработки программного обеспечения не превращалось в невозможное. Эти компании тщательно охраняют такие знания, потому что мы живем в век экономики знаний и инноваций и многие знания — многие возможности. С другой стороны, на рынке, особенно на русскоязычном, практически нет книг, которые могли бы стать настольными для системных аналитиков, программистов, системных архитекторов. Вы без труда найдете настольную книгу финансового директора, менеджера, агента по продажам и даже венчурного инвестора, но разыскать аналогичную для практика в разработке программ практически невозможно. Чаше встречаются книги, в которых слишком много либо теории, либо специфики, не всегда применимой в реальной жизни.

В различных организациях мне зачастую приходилось заново строить процессы разработки программ, и я всегда начинал с одного и того же — с наведения связующих «мостов» между уже имеющимися в организации теоретическими знаниями и моими практическими навыками и конкретными приемами. Иногда для этого приходилось

проводить тренинги и обучающие курсы с целью повышения квалификации и расширения знаний моих подчиненных и коллег. В конце концов родилась идея о создании некой «дорожной карты» (road map), в которой было бы четко показано, что нужно знать и уметь применять на каждом этапе работы в индустрии программного обеспечения.

Эта книга и есть такая «дорожная карта» — Путь аналитика.

«Почему “аналитика”?» — спросите вы. Есть две причины.

Первая заключается в том, что разработка любой программы начинается с анализа, поэтому ошибки, совершенные на этом этапе, обходятся в сто раз дороже, чем на этапе непосредственно программирования. Согласитесь, что лучше уметь избегать подобных ошибок и знать, как это сделать. Вторая причина — книга отчасти автобиографична и показывает, какой путь прошел лично я — от программиста через бизнес- и системного аналитика, архитектора до менеджера проекта, начальника отдела анализа департамента разработки и, наконец, до технического директора (CIO/CTO). Я хочу протянуть вам руку и показать, как можно пройти этот путь быстрее чем за 15 лет.

Наше время дает неограниченные возможности использования аутсорсинга как компаниям, так и профессионалам своего дела. Эта гибкая модель бизнеса применяется все шире и шире. В данном случае компания не будет тратить средства на обучение аутсорсеров практическим приемам и методам, однако спрос за требуемое качество конечного продукта останется на том же уровне.

Эта книга даст вам возможность получить нужные приемы и навыки, чтобы быть уверенными в том, что ваши проекты будут прибыльными, а заказчики и инвесторы — довольными. Я поделюсь с вами моими ноу-хау, которые позволят вам стать гуру в разработке программного обеспечения.

# Об авторах

---

---

## Андрей Дмитриевич Перерва

41 год. Master of Business Administration/Master of Business Information.

Опыт работы в индустрии разработки программного обеспечения — более 18 лет. На руководящих должностях — семь лет, из них три года в топ-менеджменте. В настоящее время специализируется на вопросах стратегического и тактического планирования, операционном управлении и управлении проектами компании по разработке ПО.

В своем карьерном развитии прошел все возможные позиции в сфере разработки ПО: работал программистом, проектировщиком и разработчиком БД, бизнес-аналитиком, системным аналитиком, архитектором, менеджером проекта, начальником отдела бизнес- и системного анализа. Принимал участие и руководил разработкой таких информационных систем, как система управления потоками работ (Workflow), Boeing Reference Engineering Data Automated Retrieval System, Customer Relationship Management System, система управления предприятием (Enterprise Management System), корпоративные порталы и системы комплексной автоматизации бизнеса. Имеет опыт организационного проектирования, создания функциональных отделов анализа с нуля, построения и развития команды, проведения обучающих тренингов для сотрудников и менеджеров компании.

Имеет практический опыт создания системы менеджмента качества (QMS) компании, построения процесса разработки ПО (SDLC) в соответствии с методологиями и стандартами RUP, Iconix, Agile, PMBOK, CMMI, ISO9001.

## Вера Алексеевна Иванова

Опыт работы в ИТ-сфере — 17 лет. Первые семь лет работала программистом в государственных учреждениях. Приобрела опыт разработки прикладного ПО с использованием СУБД, системного администрирования, технической поддержки пользователей. Участвовала во внедрении и сопровождении автоматизированных систем финансового учета регионального уровня.

---

В 2003 году начала осваивать новую специальность — системный аналитик — и продолжила работу в компаниях, специализирующихся на разработке программного обеспечения. Участвовала в разработке веб-приложений, программных продуктов по защите информации. Выполняла разные роли в проектах по разработке ПО: системного аналитика, разработчика баз данных, архитектора, тест-дизайнера, менеджера проекта. Основная специализация — системный аналитик. Последние два года специализируется в области методологии системного анализа и разработки ПО.

# Благодарности

---

---

## Андрей Перерва

В первую очередь я хочу сказать спасибо Вере Ивановой за ее идеи, умение гореть самой и поддерживать огонь в других, за соавторство и огромную проделанную работу над дополнительными материалами к книге. Тщательность и последовательность в замечаниях Веры, несомненно, принесли пользу книге и сделали изложение идей максимально точным.

Спасибо преподавателям Севастопольского государственного технического университета за заложенную основу и за то, что выполнили основную задачу вуза — научили учиться. Отдельная благодарность Владимиру Бондареву и Виктору Чернеге — вы дали путевку в жизнь многим хорошим специалистам.

Хочу поблагодарить Владимира Новикова за умелое руководство и наставничество в начале моего карьерного пути, за помощь в формировании окончательного понимания моего предназначения и выборе анализа в качестве основного направления профессиональной деятельности. Владимир всегда был для меня примером настоящего руководителя, умеющего разглядеть в каждом участнике команды его лучшие и сильнейшие стороны.

Особая признательность — компании Luxoft за отличную профессиональную школу и всем моим коллегам в этой компании за настоящий командный дух в «самом сложном проекте компании 2005 года»: Марине Олейник, Надежде Гнездиловой, Майклу Лобалзо, Андрею Шемсудову, Богдану Гурбичу, Илье Бушмелеву, Дмитрию Харитонову и всем-всем-всем, с кем мне довелось поработать. Спасибо всем за профессионализм и харизму.

Благодарю Михаила Кумскова за его тренинги по разработке программного обеспечения и управлению проектами. Эти тренинги дали серьезный толчок к моему дальнейшему погружению в предмет.

Отдельное спасибо Сергею Кузнецову за высочайший профессионализм, советы, поддержку и совместно выполненный проект по созданию практического тренинга для бизнес- и системных аналитиков.

Спасибо Владимиру Кокареву за опыт организационного управления и планирования, полученный в сложнейшем и интереснейшем проекте по построению процессов разработки. Отдельное спасибо за мудрость руководителя и умение разрешать любые конфликты.

Благодарю преподавателей бизнес-школы МВА/МБИ в ГУУ за то, что научили мыслить стратегически, смотреть и — главное — видеть шире, а также использовать лучшие мировые практики в своей деятельности. Отдельное спасибо Владимиру Годину, Владимиру Морыженкову, Зинаиде Румянцевой, Владимиру Ананьину, Светлане

Бычковой, Александру Чернову и моему дипломному руководителю Владиславу Сироте.

Я благодарен Владимиру Денисову за его советы, совместные проекты и взаимостимулирующее развитие в профессии в течение вот уже более семи лет. Работа с Владимиром всегда строится на самом высоком уровне профессиональных стандартов и практически всегда привносит что-то новое в восприятие целостной картины мира.

Хочу поблагодарить моих друзей, родных и близких — всех, кто всегда был рядом со мной в сложных жизненных ситуациях и протягивал руку помощи.

Благодарю издательство «Питер» за профессиональный подход, за то, что они поверили в успех книги и сделали ее еще лучше.

Спасибо всем вам.

В заключение хочу сказать, что если бы не понимание, поддержка, помощь и настойчивость моей жены Анаит — эта книга не появилась и не была бы издана.

## Вера Иванова

Отдельное спасибо моему профессиональному наставнику и автору этой книги — Андрею Перерве. Андрей является создателем этой книги. Спасибо ему за терпение и настойчивость, а также за неиссякаемое трудолюбие. Мне приходилось наблюдать Андрея в разных сложных жизненных ситуациях, но он всегда оставался порядочным и верным своим принципам. Это требует мужества. Андрей никогда не унывал и всегда находил в себе силы идти вперед и вести команду за собой. Спасибо ему за все и за книгу в частности.

Я хочу поблагодарить всех преподавателей физико-математического факультета Марийского государственного университета, в котором я проучилась с 1991 по 1996 год, за бескорыстный труд и любовь к науке. Но особенно я благодарна доценту кафедры вычислительной математики Илье Георгиевичу Гажееву. Он рано ушел из жизни, но успел мне помочь обрести уверенность в себе и полюбить компьютерную инженерию. Благодаря ему я решила специализироваться в ИТ-области. Каждый раз, отмечая успех в своей работе, я мысленно говорю ему спасибо.

Я благодарю за участие в моей судьбе профессора кафедры математического анализа и теории функций Марийского государственного университета Михаила Юрьевича Кокурина. К сожалению, мои математические способности оказались слишком скромными, чтобы стать хорошим ученым-математиком. Михаил Юрьевич не отказывал мне в помощи и позволил посещать его аспирантские семинары, потратил много времени на индивидуальные консультации в моей подготовке к сдаче вступительного экзамена в аспирантуру. Я благодарна ему за терпение. Михаил Юрьевич всегда был и остается для меня примером талантливого ученого-математика. Я восхищаюсь его одаренностью, преданностью любимой науке — математике и неиссякаемым трудолюбием. Основные аналитические навыки я получила от Михаила Юрьевича.

Отдельно я хочу поблагодарить всех моих преподавателей английского языка — за их безграничное терпение, но особенно Ирину Владимировну Тер-Авакян, которая была моим наставником, помогла мне обрести уверенность в себе.

Хочу также поблагодарить всех моих коллег, как бывших, так и настоящих. Благодаря им я развиваюсь как профессионал и надеюсь, что становлюсь лучше.

# Введение

---

---

Существует распространенное мнение о том, что анализ — сугубо точная наука, сродни математике или логике. Вместе с тем практики прекрасно осознают, что анализ, как и программирование, находится на стыке философии, искусства и технологии. Функции аналитика не сводятся к механическому применению стандартов и методологий. Искусство аналитика заключается в том, чтобы из богатейшего арсенала стандартов и методологий выбрать именно те, которые максимально подходят для данной задачи.

Таким образом, чтобы стать эффективным аналитиком, наряду с получением теоретических знаний необходимо перенимать и учитывать опыт коллег, развивать навыки интервьюирования, обработки и структурирования информации, выявления и управления требованиями.

Эта книга — результат 15-летней работы авторов в области разработки программного обеспечения и информационных систем. Главное достоинство книги — ее практическая направленность. В ней содержится анализ разнообразных кейсов и ситуаций, приведены примеры документов и шаблонов, необходимых для разработки ПО. Книга также структурирует огромный массив теоретической информации в зависимости от того, что прошло проверку на практике и может быть использовано в работе на каждом этапе карьеры.

## Предыстория

Собравшись в очередной раз в неформальной обстановке, мы с коллегами делились проблемами становления отдела анализа и производственными трудностями, с которыми сталкивались в работе. Обсуждали разные приемы и методы анализа и управления проектами, подыскивая оптимальные варианты, и активно дискутировали. В тот вечер я в очередной раз убедился, что одна из наиболее актуальных проблем в области разработки ПО сегодня — отсутствие унифицированного понимания в худшем случае и в лучшем случае различные представления о роли аналитика и методах анализа у разных участников проектных команд, да и в среде самих аналитиков, что зачастую порождает массу конфликтов в проектах.

Поскольку мне не раз приходилось заниматься созданием отдела и построением команд бизнес- и системного анализа в различных компаниях с нуля, я сам неоднократно сталкивался с подобными трудностями. Я очень хорошо представлял себе весь тернистый путь, который придется пройти коллегам до решения этой проблемы. Вспомнил, сколько сил и энергии мне приходилось тратить, чтобы вырастить профессиональных аналитиков, приобретая на рынке бывшего выпускника вуза с двух-трехлетним опытом работы; чтобы разработать методики и практики анализа, согласованные с другими участниками производственного процесса. Сколько нервов и времени я мог бы сэкономить себе и коллегам, если бы в свое время знал то, что знаю сейчас...

В тот момент мне и пришла в голову идея написания этой книги. Мне захотелось поделиться собственным опытом: как я развивался, какие ошибки совершал и что сделал бы сейчас совсем по-другому; чему учился, что из этого оказалось ненужным, что — преждевременным, а на что, напротив, стоило обратить внимание ранее. Я поделился своей идеей с Верой Ивановой, которая поддержала меня и любезно согласилась выступить соавтором книги.

В 2007–2008 годах мы с Верой занимались созданием методологии анализа и разработкой требований в одной из ведущих компаний по производству ПО по защите информации на российском рынке. Создавая методологию, мы должны были учесть весь наш предыдущий опыт работы и знания, лучшие мировые практики и стандарты, а также специфику работы компании. В работе принимали участие многие специалисты и эксперты компании: аналитики, архитекторы, менеджеры проектов и продуктов. Эта работа завершилась созданием концепции системы управления требованиями, многие положения и идеи которой нашли свое отражение в данной книге. Кроме того, в ней обобщен профессиональный опыт авторов книги и наших коллег, выходящий за рамки этой концепции.

Надеемся, что книга поможет вам не наступать на те же грабли, на которые наступали мы, не повторять наших ошибок, а главное — эффективно построить собственную профессиональную карьеру и пройти путь от аналитика до начальника отдела анализа.

Этот путь мы и назвали «Путь аналитика».

## Для кого эта книга

Для молодых специалистов, начинающих аналитиков, аналитиков с незначительным практическим опытом, стремящихся к развитию и самосовершенствованию через изучение лучших практик других компаний и опыта специалистов.

Для профессиональных аналитиков, руководителей групп в качестве материала по планированию своей карьеры, обучению подчиненных, а также для собственного развития и самосовершенствования через изучение лучших практик других компаний и опыта специалистов.

Для начальников отделов анализа в качестве материала для планирования развития своих подчиненных и отдела.

Мы постараемся рассказать о том, какие навыки и теоретические знания вам понадобятся на каждом шаге вашего профессионального роста и в какой последовательности их лучше приобретать и развивать.

Книга не ограничивается практиками системного и бизнес-анализа. Вы найдете здесь рекомендации и приемы работы, критические для лидера или ведущего аналитика и затем — для начальника отдела.

Уникальность этой книги в том, что в ней вы найдете практическую информацию со ссылками на теоретические материалы из разных областей разработки программного обеспечения — анализа, архитектуры, управления проектами, лидерства и управления персоналом — все то, что понадобится вам в работе на реальных производственных проектах.

Мы надеемся, что книга поможет вам составить личный план развития и максимально эффективно построить карьеру.

На сайте <http://saway4ru.codeplex.com> вы можете скачать дополнительные материалы к этой книге, в том числе некоторые шаблоны спецификаций и документов.

## Чего вы НЕ найдете в этой книге

В этой книге вы не найдете голой теории.

Мы будем приводить только те теоретические основы, которые необходимы для иллюстрации практик и методов, полагая, что читатель владеет теорией.

Анализ и другие затрагиваемые в книге дисциплины разработки ПО — довольно зрелые на сегодняшний день дисциплины, и вам не составит труда найти соответствующие материалы и учебники.

Мы не ставим цель «теоретически образовывать» читателя. Книга ориентирована на активного специалиста, который уже понял, что теоретическую базу он должен искать и изучать самостоятельно.

Мы также не стремимся доказать, что предлагаемый нами Путь аналитика — единственно правильный и эффективны только описываемые нами методики и приемы, отнюдь — к одной и той же цели ведут разные пути. Мы хотим показать вам тот, который прошли сами, в котором уверены и по которому провели молодых специалистов, выросших в серьезных профессионалов.

## Как читать эту книгу

Книга разбита на главы «Младший аналитик», «Аналитик», «Старший/ведущий аналитик», «Начальник отдела анализа».

Вы можете начать чтение с любого интересующего вас раздела любой главы в зависимости от вашей квалификации. Особенно важные места выделены специальными пиктограммами, значение которых приведено ниже.

Глава «Общие понятия», разделы «Таблица квалификационных навыков», «Полезные мелочи» и «Типичные проблемы и вопросы» каждой главы могут быть полезны для обращения после прочтения книги. Рекомендуем также активно работать с дополнительными материалами на сайте книги <http://saway4ru.codeplex.com/>.

*Примеры в книге помечены значком*



*Теоретическая информация помечена значком*



*Наиболее важные, с нашей точки зрения, методологические моменты, на которые мы рекомендуем вам обратить особое внимание, помечены значком*



Несмотря на то что эта книга обобщает практический опыт, накопленный множеством людей, профессионально занимающихся бизнес- и системным анализом в проектах по производству ПО, все дальнейшее повествование ведется от первого лица единственного числа — для простоты изложения и восприятия.

# 1. Общие понятия

---

---

## 1.1. Что такое информатизация бизнеса и зачем мне надо это знать?

Любая работающая на рынке компания создана для того, чтобы сделать ее владельцев богаче. Это главная цель практически любого бизнеса, за некоторыми исключениями.

Под словами «сделать... богаче» подразумеваются не только ежегодные дивиденды, которые получают владельцы бизнеса. Результат деятельности каждой организации можно измерить в трех показателях — прибыль, капитализация и репутация. Если деятельность предприятия не направлена на улучшение этих показателей, то ее генеральный директор управляет им неэффективно.

Основным (но не единственным) источником прибыли и капитализации является операционный поток наличности — выручка, которую приносит компании ее непосредственная деятельность. Поскольку у любой компании существуют конечные потребители, количество которых ограничено, и она практически никогда не работает в сегменте рынка одна, у конкурентов необходимо забирать максимально возможный операционный поток наличности. Эта задача достигается путем определения и реализации стратегии бизнеса. Иными словами, если цель бизнеса — экономические, социальные или иные стремления, ради которых существует предприятие, то стратегия бизнеса — это единый набор планов и действий, направленных на достижение основных целей бизнеса организации.

Бизнес-стратегия часто разрабатывается по нескольким направлениям: стратегии роста (продуктовая, повышение эффективности, капитализация, партнерство, построение сети), инновационные стратегии, конкурентные стратегии («гонка за лидером», дифференциации, лидерства в издержках, «война стандартов», выживания, поглощения).

Все эти стратегии преследуют свои цели. Цели бизнеса и цели бизнес-стратегии компании должны быть согласованы. Кроме того, бизнес-стратегия зависит от текущего и перспективного состояния рынка.



Например, если цель бизнеса — увеличить выручку до 150 млн руб., а текущая выручка равна 100 млн руб. (7 % на рынке), то стратегии на стабильном рынке и растущем рынке будут разными. На стабильном рынке необходимо наращивать производственную мощность, чтобы дать продукцию для генерации дополнительных 50 млн выручки, и, кроме того, расширять сегмент рынка (отбирать у конкурентов). На растущем рынке нужно просто наращивать производственную мощность.

После определения бизнес-стратегии соответствующих дивизионов компании, выполняющих свои функции бизнеса (маркетинг, производство и т. д.), устанавливают способ достижения целей конкурентной стратегии. Такие стратегии называются функциональными и чаще всего создаются по направлениям «Маркетинг», «Финансы», «Производство/операционная стратегия», «Стратегия НИОКР», «ИТ».

Разработка стратегии ИТ — важный и сложный процесс, который должен ответить на главный вопрос: как должна измениться архитектура ИТ в соответствии с архитектурой (моделью) бизнеса, которая, в свою очередь, корректируется с целью достижения новых требований бизнес- и функциональных стратегий.



В рассмотренном примере, исходя из определенных бизнес-стратегий, будут разработаны разные стратегии ИТ.

<p>На стабильном рынке необходимо наращивать производственную мощность, чтобы дать продукцию для генерации дополнительных 50 млн руб. выручки, и, кроме того, расширять сегмент рынка (отбирать у конкурентов)</p>	<p>Наращивание производственной мощности, возможно, потребует замены существующих ИС более оперативными и масштабными ИС уровня ERP, разработки/доработки ИС собственного производства, внедрения новых ИС сторонних производителей.</p> <p>Расширение сегмента рынка, возможно, приведет к изменению сайта компании, открытию коммерческого портала, внедрению CRM-системы, найму digital-агентства для проведения разовой маркетинговой кампании</p>
<p>На растущем рынке нужно просто наращивать производственную мощность</p>	<p>Очевидно, что в данном случае потребуется только первая часть мероприятий, описанных выше</p>

Естественно, приведенный пример не отражает реального процесса разработки ИТ-стратегии, но демонстрирует основной принцип построения интегрированной с бизнесом ИТ-стратегии: направления развития ИТ должны быть полностью интегрированы с направлениями развития бизнеса.

В условиях динамичного развития рынка информация становится для бизнеса таким же стратегическим ресурсом, как традиционные материальные, финансовые и энергетические ресурсы.

Информатизация — комплекс мероприятий, направленных на то, чтобы модель бизнеса компании наполнялась с помощью ИТ реальным информационным содержанием и получала долгосрочные конкурентные преимущества.

Архитектура ИТ — это согласованная система представлений об организации информационной среды компании. Данная система должна соответствовать модели бизнеса, определять целевое состояние ИТ, быть основой долгосрочного планирования, обеспечиваться процессами и инфраструктурой ИТ. Совокупность модели бизнеса и архитектуры ИТ представляет собой корпоративную архитектуру.

Зачем все это надо знать аналитику?



---

Аналитик должен всегда помнить, что любая разрабатываемая ИС существует не сама по себе, а в конкретной конечной ИТ-архитектуре, которая регламентируется ИТ-стратегией, согласована с ней и зависит от бизнес-архитектуры. Последняя, в свою очередь, постоянно изменяется под влиянием рынка и целей бизнеса.

---

Таким образом, разработчики системы всегда должны быть готовы к тому, что требования к разрабатываемой ИС будут постоянно меняться, и, следовательно, ее создатели должны контролировать ситуацию. Количество и степень изменений зависят от стабильности бизнеса и рынка в целом.

Чтобы разрабатывать эффективные приложения, аналитик должен понимать все аспекты, описанные выше, не только владеть информацией об ожиданиях от разрабатываемой системы, но и осознавать ее место в ИТ-архитектуре компании заказчика, тренды изменений этой архитектуры и бизнес-выгоды, которые принесет внедрение новой системы.

## 1.2. Что такое корпоративное лидерство?

Корпоративное лидерство — сочетание трех важных факторов: эффективной организации, бизнес-синергии и менеджерской команды.

Эффективная организация подразумевает создание корпоративного видения, управленческой команды, архитектуры бизнеса и корпоративной культуры, развитие и поддержание высоких корпоративных моральных стандартов во всех делах, чтобы компания могла создавать повышенную ценность для всех участников бизнеса.



---

Бизнес-синергия включает в себя создание синергии между корпоративными стратегиями, ресурсами и способностями, заполнение «белых пятен» между бизнес-единицами и дивизионами, создание эффективных связей между ними.

Менеджерская команда: создание здоровой команды корпоративных лидеров и поддержание ее эффективности на высоком уровне, выращивание

новых лидеров, создание системы мотивации сотрудников, наблюдение за тем, чтобы только лучшие люди принимались на работу, развивались дальше, вознаграждались и росли, вкладывая в бизнес свои лучшие силы и способности.

© <http://www.cecsi.ru/coach/synergy.html>

Каждый сотрудник компании должен вносить свой вклад в формирование эффективной организации всех ее составляющих. Менеджеры всех звеньев должны стимулировать участие рядовых сотрудников в этом процессе.

Конечно же, компания не станет лидером на рынке при развитии только этих факторов, но если перечисленные составляющие будут развиваться гармонично, у организации будет гораздо больше шансов стать лидером в своем сегменте. Кроме того, такая компания будет управляемой по целям, прозрачной, совершенствующейся как «сверху», так и «снизу», сможет иметь такие весомые коммерческие преимущества, как гибкость и готовность к изменениям.

К сожалению, на практике в российских компаниях вы редко встретите такой подход, но именно в ваших силах в будущем изменить эту картину.

### 1.3. Что такое архитектура, какой она бывает и зачем мне надо это знать?



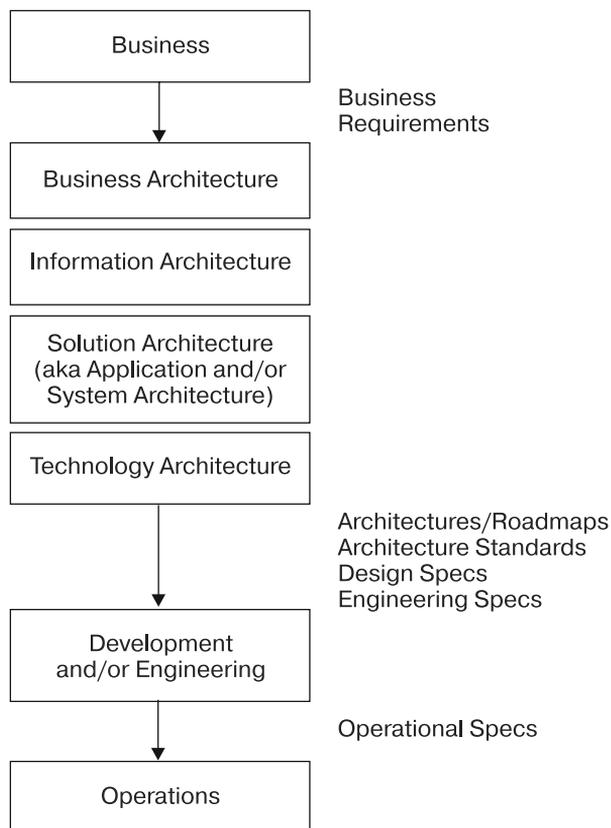
Существует много различных определений архитектуры. Все они имеют свои плюсы и минусы. Мне импонирует определение архитектуры, данное в стандарте ANSI/IEEEStd 1471-2000: «Архитектура — это базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы».

#### Архитектура предприятия/корпоративная архитектура (Enterprise Architecture)

Это высокоуровневая архитектура всего предприятия, отражающая бизнес-потребности компании, реализуется с помощью ИТ-средств. Корпоративная архитектура фокусируется на определении потоков и бизнес-процессов, действий, функций, информации, данных и технологий предприятия, а также на вызовах, стоящих перед ИТ, которые необходимы для того, чтобы эффективно применить технологию в ответ на изменение бизнес-потребностей.

Enterprise Architecture объединяет Business Architecture, Information Architecture, Solution Architecture и Technology Architecture [6] (рис. 1).

**Business Architecture.** Описывает все бизнес-процессы, бизнес-факторы, бизнес-сущности и бизнес-правила с точки зрения бизнеса. Business Architecture не зависит от применяемых ИТ-технологий.



**Рис. 1.** Архитектура предприятия/корпоративная архитектура

**Information Architecture.** Определяет структуры данных и описывает все потоки данных, которые используются для поддержки Business Architecture. Такие операции, как идентификация, систематизация, категоризация, хранение данных, относятся к Information Architecture. Может представляться в виде Data Model.

**Solution (System/Application) Architecture.** Архитектура программного обеспечения, которое реализует функции Business Architecture.

**Technology Architecture.** Описывает архитектуру ИТ-окружения, которое используется для поддержки Information Architecture и Solution (System/Application) Architecture.

Различные представления архитектуры (Business Architecture, Information Architecture, Solution Architecture и Technology Architecture) тесно взаимосвязаны, и их интеграция – необходимое условие успешного построения Enterprise Architecture.

С точки зрения разработки ПО наиболее важное понятие – Solution (System/Application) Architecture, основными составными частями которой являются System Architecture, Data Architecture и Software Architecture.

**System Architecture.** Представление системы, которое показывает реализацию функциональных возможностей системы аппаратными средствами и компонентами программного обеспечения, устанавливает связь архитектуры программного обеспечения и архитектуры аппаратных средств, а также регламентирует взаимодействие пользователей с этими компонентами. Существуют и другие определения System Architecture, например: ряд взаимосвязанных шаблонов (паттернов), которые структурируют модули и данные и обеспечивают требуемое поведение системы (см. определение Data Architecture). System Architecture является составной частью Solution Architecture.

**Software Architecture.** Составная часть System Architecture. Описывает организацию системы с точки зрения программных компонентов, из которых она состоит, и связи между компонентами.

**Data Architecture.** Составная часть System Architecture. Описывает структуры данных и логические связи между данными (hierarchical model, network model, relational model, object model и т. д.).

Аналитик должен понимать смысл вышеприведенных понятий, их предназначение и взаимосвязь. С точки зрения *развития продукта* важно **понимание Enterprise Architecture, разработки продукта** — Software Architecture, System Architecture, Data Architecture.

## 1.4. Что такое требование?



Согласно определению Institute of Electrical and Electronics Engineers, Inc., ведущей мировой профессиональной ассоциации по развитию технологий, требование — это:

- (а) условие или способность, необходимая пользователю, чтобы решить проблему или достигнуть цели;
- (б) условие или способность, которыми должны обладать система либо компонент системы для удовлетворения контракта, стандарта, спецификации или другого формально установленного документа;
- (в) документированное представление условия или способности, показанной в определении «а» или «б».

© (IEEE Std 610.12-1990)

Под требованием заказчика понимается потребность или ожидание, которое:

- ◆ установлено;
- ◆ предполагается (подразумевается);
- ◆ является обязательным.

Рассмотрим это на примере. В беседе заказчик выразил ожидание, которое звучит так: «Сделайте мне редактор писем, чтобы я мог выделять разные слова разным цветом». В данном случае:

- ◆ **установлено:** программа должна уметь читать формат писем, редактировать письма и сохранять их;
- ◆ **подразумевалось:** «естественно, я хочу, чтобы это был встроенный в почтовый клиент редактор» (в идеале должно быть установлено, а значит, сформулировано аналитиком явно);
- ◆ **обязательно:** выделять разные слова разным цветом.

Приведенный пример показывает, что недостаточно просто зафиксировать пожелание заказчика так, как он его высказал. Необходимо провести анализ и добиться того, чтобы зафиксированная информация обладала следующими основными характеристиками требования.



---

Требование должно быть **тестируемым**.

Требование должно быть **реализуемым**.

Требование должно быть **ясным для понимания**.

Требование должно быть **законченным и полным**.

---

В разработку поступают не любые утверждения заинтересованных лиц. Требование — это результат анализа. Требование должно отвечать на вопрос «что должна делать система», а не как она будет это делать.

Требования бывают разных типов и уровней.

Так, например, по К. Вигерсу [8] структура уровней требований выглядит следующим образом (рис. 2).

В Rational Unified Process (RUP) структура уровней требований выглядит иначе (рис. 3). На данной иллюстрации архитектурные представления не показаны. Более подробно вы можете изучить их в работе [5].

Обычно в компании по созданию ПО разрабатывается своя методология разработки и управления требованиями, которая адаптируется из признанных мировых стандартов. В рамках определения этой методологии формируется и структура уровней требований, или иерархия типов требований. Подробно типы требований будут рассмотрены далее.

## 1.5. Почему надо знать теории и методологии разработки ПО?

Около десяти лет назад дисциплина системного анализа в том виде, в котором она существует сейчас, находилась в России в стадии становления. Нет, системным анализом занимались и раньше, причем не менее профессионально, чем сейчас. В Советском Союзе существовали специальности «инженер-системотехник», «инженер-математик», наиболее близкие к нынешнему «системному аналитику». Однако в области разработки ПО использовались в основном

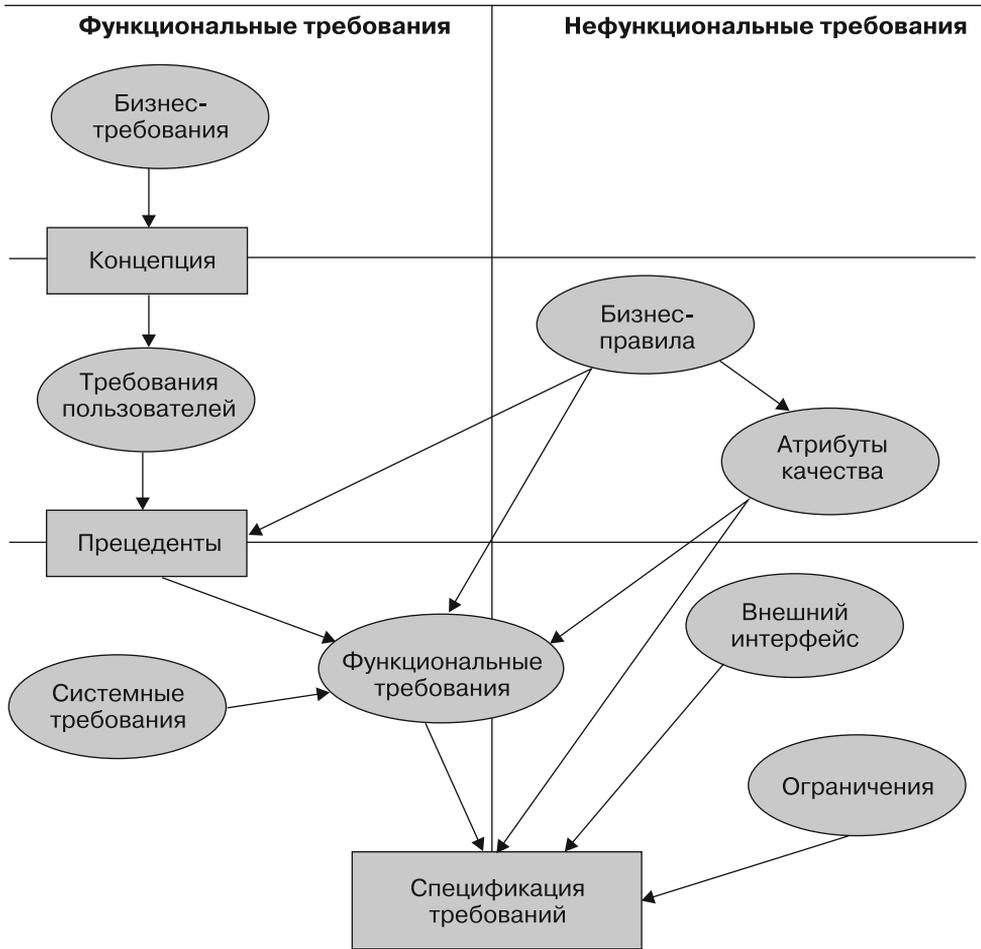


Рис. 2. Структура уровней требований по К. Вигерсу

методы структурной и функциональной декомпозиции. Шло время. ИТ-технологии активно развивались, появлялись новые платформы, парадигмы и языки программирования. Некоторыми западными институтами и организациями были предприняты попытки создания и описания процесса разработки программного обеспечения. В настоящее время многие из этих описаний стали де-факто стандартами в области разработки ПО (RUP, CMMI, Swebook и др.).

Одним из первых таких стандартов, с которыми я ознакомился, был RUP. Кажется, это было в 1997 году. В то время я занимался разработкой ПО на Java и структурных языках и не мог не оценить, насколько эффективно данная методология позволяла создавать приложения на языках ООП. Передо мной была поставлена задача по созданию новой системы. Решение этой задачи потребовало взаимодействия с более опытными сотрудниками, создания и согласования ТЗ к системе, программирования и обсуждения полученного ре-

## Traceability Overview

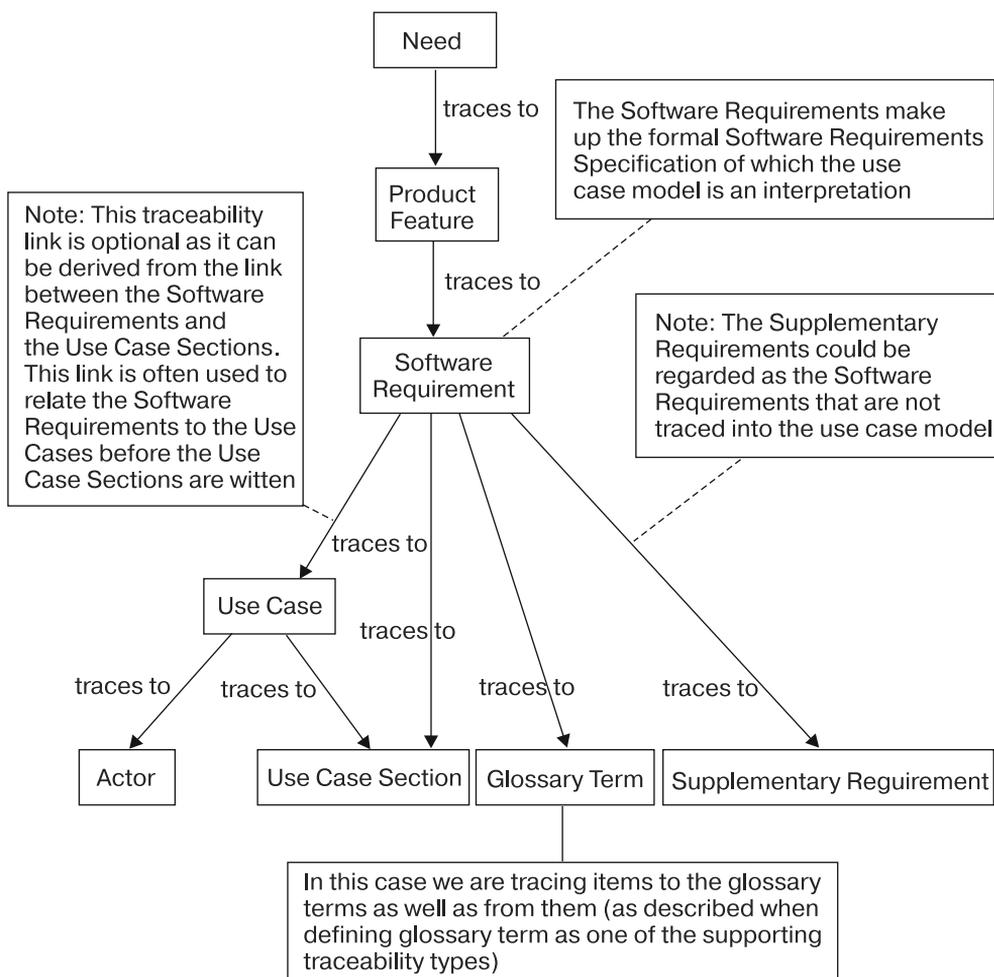


Рис. 3. Структура уровней требований в RUP (Rational Unified Process)

зультата с экспертами и коллегами. Специализированную литературу на русском языке тогда было практически невозможно найти. В стандарте RUP были описаны все роли в процессе разработки ПО и артефакты, посредством которых фиксируются все достигнутые договоренности с проектной командой. Мне стало понятно, какое количество ошибок я допустил, пытаясь выстроить взаимодействие и процесс разработки самостоятельно. Я осознал, почему у меня возникают сложности в согласовании ТЗ, в котором содержались требования к уровню дизайна (по-моему, там даже были какие-то фрагменты алгоритмов и структуры данных), с сотрудниками, выступающими от лица заказчика, но самое главное — я открыл для себя отличный инструмент для моделирования будущей системы и узнал о существовании языка UML.

Это открытие помогло мне построить процесс разработки системы максимально эффективно, а навыки моделирования поведения системы и объектно-ориентированного моделирования пригодились в общении с более опытными коллегами.



---

Сегодня существует несколько стандартов и методологий для разработки ПО: RUP, MFS, Iconix, спиральная разработка Boehm, Agile, XP и т. д. В книге вы найдете краткое описание и сравнительный анализ некоторых методологий. Не существует «серебряной пули» — методологии, максимально эффективной во всех случаях, — для всех видов проектов, систем любого типа и всех компаний. В каждом конкретном случае компания должна определить эффективную для себя методологию и применять ее при разработке ПО. Обычно за основу берется одна из «стандартных» методологий, которая затем адаптируется под специфику бизнеса организации — производителя ПО.

---

## 1.6. Почему бизнес- и системный анализ объединены в одной книге?

Строго говоря, бизнес- и системный анализ — разные дисциплины, и в идеале при разработке ПО должны выделяться две роли: бизнес-аналитик и системный аналитик, которые выполняют разные функции. Однако на практике эти роли часто совмещает один человек или одна из ролей может вообще отсутствовать. Уверен, что и вам в вашей профессиональной деятельности придется совмещать эти роли.

Я убежден в том, что профессиональный системный аналитик должен уметь проводить бизнес-анализ. Для разработки системы даже полезно, если системному аналитику удалось достаточно глубоко погрузиться в бизнес-область, поскольку хорошее понимание потребностей бизнеса приводит к созданию максимально полезных бизнесу продуктов.

В совмещении ролей системного и бизнес-аналитика нет ничего страшного, поскольку большинство методов и практик этих дисциплин пересекаются между собой. Конечно, в компетенции бизнес-аналитика останутся описание и поиск путей оптимизации бизнес-процессов, а в компетенции системного аналитика — моделирование поведения системы и выявление системных требований.

## 1.7. Что, кроме дисциплин анализа, должен знать хороший аналитик?

В современном мире разработки ПО сложность процессов и взаимодействий нарастает, при этом требования к скорости и качеству выполнения проектов по-

стоянно повышаются. Аналитик находится на переднем крае общения команды и заказчика, и от того, насколько точно и правильно он выявит и сформулирует требования к разрабатываемому продукту, зависит то, правильным путем пойдет вся команда или нет.

Аналитик работает с людьми, у которых есть свои привычки, убеждения, опыт. Для построения эффективных коммуникаций аналитик должен быть чуть-чуть психологом, обладать навыками предотвращения и разрешения конфликтов, уметь одинаково эффективно работать формально и неформально, самостоятельно и в команде. Развитие необходимых личностных факторов будет рассмотрено в главе «Младший аналитик».

Второй аспект работы аналитика — взаимодействие со всеми членами проектной команды: менеджером продукта, менеджером проекта, архитектором, тест-менеджером, разработчиками (программистами), тестировщиками. Здесь не обойтись без специальных технических навыков, иначе аналитику будет очень трудно найти общий язык с техническими специалистами. Аналитик должен не только владеть практиками анализа, но и постоянно расширять свой кругозор в предметной области разрабатываемого продукта и в технологиях, применяемых при производстве этого продукта. В главах «Аналитик» и «Старший/ведущий аналитик» мы поговорим о развитии необходимых профессиональных и специальных технических навыков.

Взаимодействуя с таким большим количеством людей, аналитик должен уметь планировать свою работу, управлять рисками и коммуникациями. Поэтому он должен обладать качествами лидера и немножко быть менеджером проекта. В принципе, работу аналитика можно назвать «проектом в проекте», результатом которого являются спецификации требований, потребителями результата — архитекторы, разработчики и тест-менеджеры, а заинтересованными лицами — практически все члены проектной команды. В главе «Начальник отдела анализа» мы подробнее поговорим о развитии необходимых лидерских и управленческих качеств.

Ну и самое главное: аналитик должен быть высоким профессионалом в технологиях и методах системного и бизнес-анализа, иначе цель его работы с точки зрения «проекта» может быть достигнута, но результат будет некачественным, что приведет к созданию некачественного и/или невостребованного продукта.

Если вас не испугал приведенный перечень всех «должен» — вы на правильном пути, в противном случае стоит подумать о других возможностях профессионального развития.

## 2. Профиль и квалификация аналитиков

---

---

В России силами организации АП КИТ была предпринята попытка создать отраслевой профессиональный стандарт «Системный аналитик» (<http://www.apkit.ru/files/analitik.doc>).

Однако до сегодняшнего дня данный стандарт не получил широкого признания и распространения, что неудивительно, так как достаточно сложно провести четкую грань между различными уровнями профессионализма.

Каждая компания предъявляет свои требования к кандидатам на позиции младшего аналитика и аналитика, ведущего аналитика и начальника отдела. Поэтому квалификационная шкала, которую я предлагаю в данной книге, — условная, ее нельзя считать единственно возможной. Шкала базируется на моем личном опыте работы в трех компаниях — я «усреднил» квалификационные требования к равнозначным позициям специалистов-аналитиков и постарался вывести некую наиболее вероятную градацию квалификаций и требований к ним.

Шкала квалификации в книге нужна главным образом для того, чтобы показать значимые вехи на Пути аналитика. Эти вехи характеризуются разным наполнением профиля аналитика.

Под профилем аналитика я понимаю следующие составляющие (рис. 4).

В основе профиля аналитика лежат личностные навыки, без развития которых вы не состоите в этой профессии. Будьте готовы к тому, что вам придется постоянно чему-то учиться. И по мере узнавания нового вы будете понимать, что знаете слишком мало, — этакая здравая энтропия профессии. Выбрав Путь аналитика, вы выбираете определенный стиль мышления, общения, развития и самой жизни в целом. Личностные навыки — это «красный свет светофора» на Пути аналитика. Без наличия и развития некоторых навыков лучше вообще уйти из этой профессии.

Первый внешний сегмент в профиле аналитика — профессиональные навыки. Желтый цвет сегмента — это деньги, которые вы тратите на книги, ваши инвестиции в самого себя, «желтый свет светофора». Получив нужные профессиональные знания и навыки, вы уже сможете продавать себя на рынке.



**Рис. 4.** Профиль аналитика

Но продавать себя выгодно вы сумеете, только получив специальные навыки, которые приобретаются в ходе практических работ, выполненных реальных проектов, зачастую путем проб и ошибок. Это ваш «зеленый свет» на Пути аналитика к дальнейшему развитию, и это цвет денег, которые вы сможете зарабатывать, уже не просто «продавая» себя на рынке, но делая это выгодно.

Давайте посмотрим, как изменяется профиль аналитика в зависимости от выделенных в книге квалификаций (рис. 5).

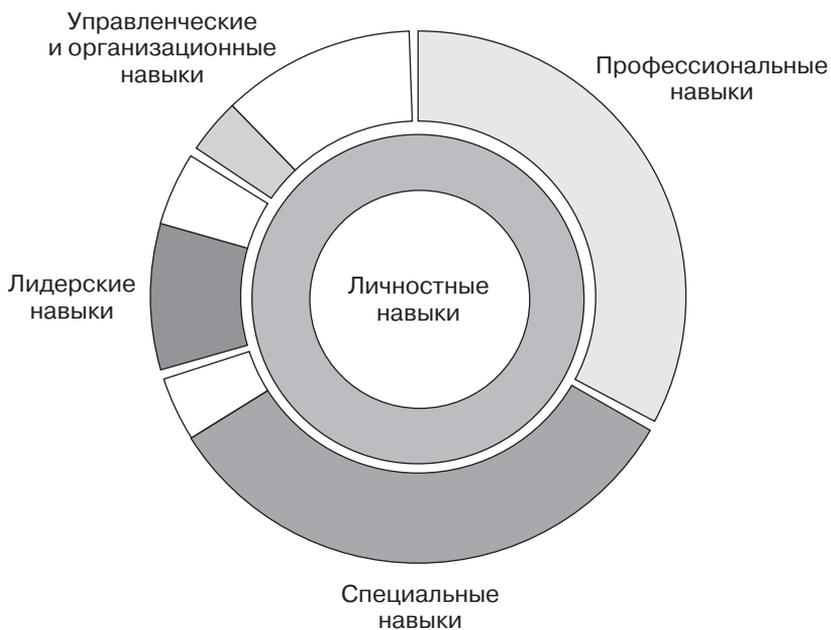
Несмотря на то что на диаграмме сегменты профессиональных и специальных навыков закрашены полностью, на самом деле всегда остаются «белые пятна», так как технологии и методологии не стоят на месте и постоянно развиваются. Поэтому аналитик должен непрерывно работать над всеми составляющими своего профиля, чтобы не вернуться на предыдущий уровень. Обратите также внимание на то, что профиль аналитика от квалификации к квалификации становится объемнее и «весомее». То есть происходит качественное и экстенсивное развитие.

Если взглянуть на вышесказанное с точки зрения карьерного роста, то Путь аналитика состоит из следующих шагов: младший аналитик → аналитик → старший аналитик → ведущий аналитик (лидер) → начальник отдела анализа. С этого пути можно «свернуть» как минимум дважды: на этапе «аналитик» можно уйти в архитектуру и стать архитектором или на этапе «ведущий аналитик» уйти в область управления проектами и стать менеджером проекта.

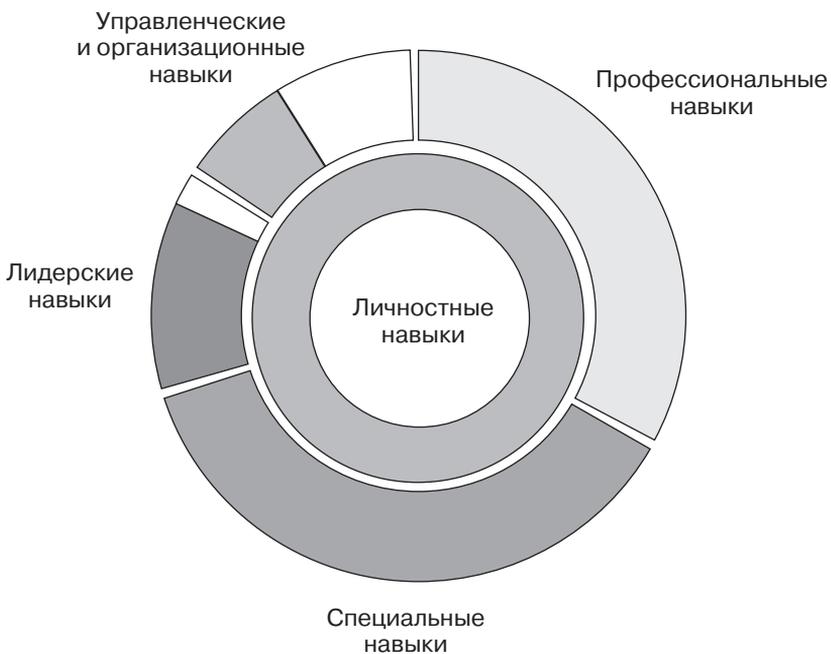
Какие же навыки и знания нужны на каждом этапе? Об этом — в следующем разделе.

**Младший аналитик****Аналитик****Рис. 5.** Профиль аналитиков в зависимости от квалификации (начало)

**Старший/ведущий аналитик**



**Начальник отдела**



**Рис. 5** (окончание)

## 2.1. Таблица квалификационных навыков



В табл. 1 представлена детальная информация о необходимых, на мой взгляд, компетенциях аналитика для каждой квалификации.

**Таблица 1.** Профессиональные и специальные навыки аналитика

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
<p><b>Младший аналитик</b></p>	<p>Книгу Дина Леффингвелла и Дона Видрига Managing Software Requirements я называю «библией аналитика». Помимо теоретической информации, вы найдете здесь ряд конкретных примеров и приемов проведения интервью, «мозговых штурмов» и т. д.</p> <p>Еще одна, более поздняя и не менее «евангелистская», книга в области анализа — «Разработка требований к программному обеспечению» Карла И. Вигерса.</p> <p>Иметь общее представление о различных методологиях разработки ПО.</p> <p>Знать основы RUP, а именно дисциплину Requirements.</p> <p>Знать и уметь применять UML, а именно Use Case-модель, уметь строить Domain Object Model.</p> <p>Понимать основные принципы объектно-ориентированного проектирования и моделирования.</p> <p>Рекомендую также почитать статьи по аналитике на сайте <a href="http://www.interface.ru">www.interface.ru</a></p>	<p>Выявлять заинтересованных лиц (ЗЛ).</p> <p>Управлять ожиданиями ЗЛ.</p> <p>Проводить собрания.</p> <p>Проводить интервьюирование.</p> <p>Проводить анкетирование.</p> <p>Проводить «мозговые штурмы».</p> <p>Уметь определять границы системы.</p> <p>Уметь выделять подсистемы и определять их границы.</p> <p>Уметь выявлять требования типов:</p> <ul style="list-style-type: none"> <li>• ответы и собранная информация;</li> <li>• запросы заинтересованных лиц;</li> <li>• глоссарий;</li> <li>• стандарты и ГОСТы;</li> <li>• характеристики аналогичных/наследуемых систем;</li> <li>• бизнес-требования;</li> <li>• бизнес-правила;</li> <li>• концепция создания и развития продукта (BVISION);</li> <li>• ограничения и допущения;</li> <li>• концепция системы (TVISION);</li> <li>• пользовательские требования;</li> <li>• функциональные требования;</li> <li>• функции системы/варианты; использования/прецеденты (Use Cases);</li> <li>• нефункциональные требования;</li> <li>• требования к пользовательскому интерфейсу;</li> </ul>

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
		<ul style="list-style-type: none"> <li>• требования к взаимодействию с внешними системами.</li> </ul> <p>Выявлять функции системы (Use Cases), моделировать поведение системы.</p> <p>Уметь строить трассировки/прослеживать требования.</p> <p>Понимать основные принципы тестирования.</p> <p>Знать английский язык на уровне, достаточном для чтения технической литературы</p>
<b>Аналитик</b>	<p>Все теоретические знания младшего аналитика, а также:</p> <ul style="list-style-type: none"> <li>• знать и уметь применять ГОСТ 34-й и 19-й серий;</li> <li>• знать и уметь применять нотацию IDEF0, диаграммы eEPC (extended Event Process Chain).</li> </ul> <p>Книги:</p> <ul style="list-style-type: none"> <li>• <i>Rosenberg Doug, Scott Kendall. Use Case Driven Object Modeling with UML: A Practical Approach;</i></li> <li>• <i>Rosenberg Doug. Agile Development with Iconix Process.</i></li> </ul> <p>Знать <b>основы RUP, а именно:</b> дисциплины Requirements, Analysis &amp; Design.</p> <p>Знать и уметь строить Robustness- и Sequence-диаграммы, Analysis Model.</p> <p>Расширять свой кругозор в методологиях. Познакомиться с гибкими методологиями, например Iconix.</p> <p>Рекомендую регулярно посещать сайты <a href="http://www.uml2.ru">www.uml2.ru</a>, <a href="http://www.agilerussia.ru/">http://www.agilerussia.ru/</a></p>	<p>Все навыки младшего аналитика, а также:</p> <ul style="list-style-type: none"> <li>• знать, что такое ПУТ, и уметь его разрабатывать;</li> <li>• понимать, какие модели существуют и где их место в разработке ПО;</li> <li>• уметь создавать модель анализа;</li> <li>• строить Robustness- и Sequence-диаграммы, понимать, зачем их вообще надо строить;</li> <li>• уметь читать программный код;</li> <li>• иметь навыки проведения презентаций</li> </ul>

(Продолжение)

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
<p><b>Старший/ведущий аналитик</b></p>	<p>Все теоретические знания аналитика, а также книги и статьи:</p> <ul style="list-style-type: none"> <li>• Данилин А., Слюсаренко А. Архитектура и Стратегия. «Инь» и «Янь» информационных технологий предприятия;</li> <li>• Boehm Barry W. A Spiral Model of Software Development and Enhancement;</li> <li>• A Guide to the Project Management Body of Knowledge. ANSI/PMI;</li> <li>• статья: Ebert C. Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques.</li> </ul> <p>Рекомендую регулярно посещать сайты <a href="http://msdn.microsoft.com/ru-ru/default.aspx">http://msdn.microsoft.com/ru-ru/default.aspx</a> и <a href="http://www-01.ibm.com/software/success/cssdb.nsf/topstoriesFM?OpenForm&amp;Site=rational&amp;cty=en_us">http://www-01.ibm.com/software/success/cssdb.nsf/topstoriesFM?OpenForm&amp;Site=rational&amp;cty=en_us</a></p>	<p>Все навыки аналитика, а также:</p> <ul style="list-style-type: none"> <li>• иметь детальное представление о жизненном цикле проекта и продукта;</li> <li>• знать, что такое ПУД, и уметь его создавать;</li> <li>• уметь создавать логическую модель и модель данных;</li> <li>• уметь создавать простой программный код (этот навык нужен для умения читать программный код — ведь никто так не поймет разработчика, как другой разработчик, и если вы хотите говорить с разработчиками на одном языке, вам придется освоить программирование в минимальном объеме);</li> <li>• уметь профессионально проводить презентации;</li> <li>• проводить выученные уроки по практикам разработки и управления требованиями;</li> <li>• быть наставником для аналитиков;</li> <li>• уметь предотвращать и разрешать конфликты в проектах;</li> <li>• уметь выявлять риски и управлять ими</li> </ul>
<p><b>Начальник отдела</b></p>	<p>Книги:</p> <ul style="list-style-type: none"> <li>• Capability Maturity Model for Software. SEI;</li> <li>• Mulcahy Rita/ PMP® Exam Prep. PMP;</li> <li>• De Carlo Doug. eXtreme Project Management;</li> <li>• Минцберг Г. Структура в кулаке: создание эффективной организации;</li> <li>• Адизис Ицхак. Идеальный руководитель;</li> </ul>	<p>Все навыки аналитика, а также:</p> <ul style="list-style-type: none"> <li>• знать модель зрелости процессов компании CMMI 1.2. в областях: RD, REQM, DAR, TS, PI, VER, RSKM, PP, PMC, IPM, VAL, QPM, SAM;</li> <li>• уметь анализировать эти области на предмет требуемых улучшений и несоответствий с моделью;</li> <li>• разрабатывать методологию системного анализа для компании;</li> </ul>

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
	<ul style="list-style-type: none"> <li>• <i>Фиорина Карли</i>. Трудный выбор. Уроки бескомпромиссного лидерства в сложных ситуациях от экс-главы Hewlett-Packard.</li> </ul> <p>Рекомендую регулярно посещать сайты: <a href="http://www.iteam.ru/">http://www.iteam.ru/</a>, <a href="http://msdn.microsoft.com/en-us/architecture/default.aspx">http://msdn.microsoft.com/en-us/architecture/default.aspx</a>; <a href="http://www.cecsi.ru/">http://www.cecsi.ru/</a>.</p> <p>Рекомендую зарегистрироваться в сети профессиональных контактов (например, <a href="http://moikrug.ru">moikrug.ru</a>, <a href="http://linkedin.com">linkedin.com</a>), заниматься просветительской деятельностью, участвовать в конференциях, форумах</p>	<ul style="list-style-type: none"> <li>• проводить тренинги и семинары;</li> <li>• иметь четкое представление об управлении проектом/программой проектов;</li> <li>• уметь строить и развивать команду аналитиков в проектах;</li> <li>• проводить выученные уроки по результатам выполненных работ/проектов;</li> <li>• участвовать в совершенствовании процессов;</li> <li>• разрабатывать процедуры, регламенты, рабочие инструкции;</li> <li>• создавать функциональную стратегию своего направления;</li> <li>• планировать развитие отдела, вовлекать топ-менеджеров в решение стратегических и тактических вопросов;</li> <li>• выстраивать эффективное взаимодействие с другими подразделениями;</li> <li>• разрешать конфликты на всех уровнях;</li> <li>• быть способным управлять проектом;</li> <li>• профессионально развивать подчиненных;</li> <li>• уметь проводить аттестацию сотрудника;</li> <li>• курировать создание базы знаний отдела;</li> <li>• управлять совершенствованием процессов в области анализа, разработки и управления требованиями;</li> <li>• управлять формализацией процессов и созданием системы менеджмента качества отдела</li> </ul>

# 3. Младший аналитик

---

---

В этой главе мы рассмотрим вопросы, касающиеся необходимых квалификационных навыков и знаний начинающего аналитика.

Прежде чем перейти к анализу таблицы квалификационных навыков, рассмотрим личностные навыки. Как я уже писал, выбирая Путь аналитика, вы выбираете определенный стиль мышления, общения, развития и самой жизни в целом. Личностные навыки — это «красный свет светофора» на Пути аналитика. Без наличия и развития некоторых навыков лучше вообще не вставать на этот путь.

## 3.1. Личностные навыки

В основе личностных навыков лежит ваша личность. Здесь важно все — воспитание, характер, образование, жизненный и профессиональный опыт, культура речи, ясность мышления. И если некоторым навыкам, таким как жизненный опыт и воспитание, нельзя научить — это данный вам базис, то остальные навыки вы можете и должны в себе развивать. Главное — определиться с тем, какие именно навыки будут полезны в профессии, а значит, и в жизни. Попробуемся найти ответ на этот вопрос.



---

Я считаю, что самыми главными навыками системного аналитика являются:

- ◆ аналитический стиль мышления;
  - ◆ умение сконцентрироваться и проследить главную нить мысли;
  - ◆ умение конспектировать;
  - ◆ умение выстраивать эффективные коммуникации;
  - ◆ умение слушать;
  - ◆ терпимость к чужой точке зрения, не совпадающей с собственной;
  - ◆ умение оставаться спокойным и отделять факты от эмоций;
  - ◆ умение подыскивать краткие и ясные иллюстрации;
  - ◆ умение помнить не только озвученные ожидания заинтересованных лиц, но и историю изменения ожиданий и требований (отличная память).
-

Что же такое аналитический стиль мышления? Я расскажу о своем видении этого понятия. Определение будет длинным, поэтому постарайтесь сконцентрироваться и не потерять фокус и нить моей мысли, чтобы воспринять определение как целостную, объемную картину. Итак, приготовились?

### 3.1.1. Аналитический стиль мышления



**Аналитический стиль мышления** — умение обрабатывать и анализировать информацию, выявлять понятия, сущности и их атрибуты, устанавливать взаимосвязи и отношения «целое — часть» между этими сущностями, выявлять и отслеживать причинно-следственные связи между событиями и их воздействием на выявленные сущности, а также их состояние; умение структурировать информацию; способность разобраться, кто и в какой информации нуждается и как он ее использует в своей деятельности/жизни; умение выделять главное; способность выделить из всей собранной информации только нужную и важную часть для каждого конкретного заинтересованного лица для обсуждения, уточнения и согласования; умение предвидеть, какая информация подвержена изменениям со стороны заинтересованных лиц и/или под влиянием объективной реальности бизнеса.

Это то, к чему, по моему мнению, должен стремиться любой системный и бизнес-аналитик. Почти все из перечисленных навыков можно развить. Я расскажу о некоторых методах, помогающих в этом.

Начнем с умения обрабатывать информацию.

#### Процесс обработки и анализа информации

Процесс обработки и анализа информации можно представить в виде следующих шагов:

- ◆ определить группы/категории информации;
- ◆ определить атрибуты для дополнительной категоризации информации;
- ◆ разбить информацию на определенные ранее группы;
- ◆ внутри каждой группы выделить главные информационные составляющие и построить от них иерархию;
- ◆ присвоить значения дополнительным атрибутам, определенным ранее;
- ◆ определить перечень возможных состояний информационных составляющих;
- ◆ присвоить состояния информационным составляющим;
- ◆ провести трассировки/обеспечить прослеживаемость одной информационной составляющей из другой.

Конечно, это не значит, что аналитик каждый раз выполняет все эти действия, четко следуя алгоритму, но если вы присмотритесь к работе профессионального

аналитика и прислушаетесь к его высказываниям в течение какого-то времени, например, при попытке решить совместно какую-то проблему, где требуется обработать значительные объемы информации, то заметите, как аналитик выполняет эти шаги — в уме, на бумаге или с помощью специальных средств и компьютерных программ.



Давайте рассмотрим процесс обработки информации «в голове» аналитика на несерьезном примере всем известной поговорки «Шла Саша по шоссе и сосала сушку». Сама по себе фраза не несет четко определенного смысла, но, поверьте моему опыту, вы очень часто будете сталкиваться с подобными высказываниями заказчиков.

Шаги алгоритма обработки информации	Информация
	Шла Саша по шоссе и сосала сушку
Определить группы/категории информации	Саша. Цели Саши. Действия Саши
Определить атрибуты для дополнительной категоризации информации	Источник информации
Разбить информацию на определенные ранее группы	<p>Саша.</p> <p>Недостаточно информации. Вопросы: возраст, вес, рост, размер шага?</p> <p>Цели Саши:</p> <p><i>глобальная цель</i></p> <p>Недостаточно информации. Вопросы: куда шла? Откуда? Зачем? Что она несла с собой?</p> <p><i>локальная цель</i></p> <p>Недостаточно информации. Вопросы: шла, чтобы достичь глобальной цели, или просто решила погулять? Почему она сосала сушку? Она голодная, у нее болят зубы или она просто любит сушки? Это оптимальный способ достижения локальной цели?</p> <p>Действия Саши:</p> <p><i>шла по шоссе</i></p> <p>Недостаточно информации. Вопросы: в течение какого времени она шла (на настоящий момент)? С какой скоростью? Шла просто так или вприпрыжку? С кем шла?</p> <p><i>сосала сушку</i></p> <p>Недостаточно информации. Вопросы: почему именно сушку? Это последняя сушка? Почему она просто не съела сушку?</p>

Шаги алгоритма обработки информации	Информация
Внутри каждой группы выделить главные информационные составляющие и построить от них иерархию	На текущий момент невозможно
Присвоить значения дополнительным атрибутам, определенным ранее	Источник информации: Сидоров
Определить перечень возможных состояний информационных составляющих	Для всех информационных составляющих всех групп (цели, действия, факты о Саше) состояния одинаковые: выявлено; для прояснения; согласовано; утверждено
Присвоить состояния информационным составляющим	«Для прояснения»
Провести трассировки/обеспечить прослеживаемость одной информационной составляющей из другой	Почему она просто не съела сушку? → Это оптимальный способ достижения локальной цели? В течение какого времени Саша шла (на настоящий момент)? → Куда шла, откуда, зачем?

Как показывает приведенный пример, в ходе обработки информации у аналитика появляется масса вопросов, цель которых — как можно более тщательно прояснить изначальное высказывание, чтобы иметь максимально полную и исчерпывающую картину.

В этом примере невозможно понять, почему заданы именно эти вопросы, так как не определен контекст общения. Если бы контекст общения был задан как: «разрабатывается система по вычислению/нахождению оптимального маршрута пешего пути между пунктами А и Б для людей любого пола и возраста», то многие вопросы из списка бы исчезли, а их место заняли контекстно-зависимые вопросы, например: какую среднюю скорость Саша развивает в зависимости от погоды и вида покрытия шоссе, как зависит скорость Саши от собственного веса и физического состояния, веса ручной клади, вида ручной клади (сумка/чемодан с колесиками). Целью данного примера является иллюстрация навыка обработки и анализа информации. Этот навык надо развивать, его не получится «включать» на работе и «выключать» в обычной жизни — такой стиль мышления должен со временем стать второй натурой.

Теперь давайте рассмотрим реальный пример из профессиональной области.



Попробуйте по аналогии выполнить обработку и анализ ожидания заказчика от создаваемой системы обмена файлами: «Бизнес компании требует обеспечения гарантированной доставки файлов как внутри корпоративной сети компании, так и при пересылке через публичные сети между ее офисами».

Я бы выделил такие информационные группы: корпоративная сеть, публичная сеть и общие требования, поскольку требования к поведению системы в этих сетях, скорее всего, отличаются друг от друга, но существуют и общие требования. С другой стороны, группы можно было выбрать и другие, например: передача файлов, архивирование файлов, просмотр и редактирование файлов, администрирование, а типы сетей отмечать в виде атрибутов. Дополнительными атрибутами здесь я бы выделил «ФИО заинтересованного лица», так как при работе с утверждениями заказчика это крайне важно, и добавил бы атрибут «Отдел» со значением, скажем, «Административный офис», и атрибут «Источник» со значением, например, «Протокол встречи с генеральным директором от 1.01.2009». Из анализа информации при распределении ее по группам у меня появляются вопросы: что такое гарантированная доставка файла? Должен ли отправитель получать уведомление о результате отправки файла? В каком виде? Где физически будет храниться файл при отправке внутри сети? В центральном хранилище данных (БД, сеть?) или в персональных папках пользователей? Все ли могут послать файлы всем? Как быть с огромными файлами? Как бороться с возникающими дублями? Нужна ли версионность отправляемых файлов? А история — кто, кому, когда и что отправил? Необходимо ли обеспечить защиту от несанкционированного доступа к содержимому файла и его изменения при передаче? По каким алгоритмам? Под какими ОС должна работать система? Нужна ли конвертация файлов из разных форматов? Как система будет взаимодействовать с другими информационными системами (антивирусное ПО, защита трафика и т. п.)? Есть ли в компании утвержденная политика информационной безопасности? Какая? И т. д.

Согласитесь, первоначальное утверждение, мягко говоря, не отражает полной картины, и для того, чтобы мы могли из этого утверждения получить тестируемые, реализуемые, ясные для понимания, законченные и полные требования, нам потребуется проводить еще не один цикл интервью. Учитывая, что автор этого ожидания — генеральный директор компании, для ответов на вопросы надо будет привлечь других сотрудников, потому что генеральный директор вряд ли сможет ответить на все вопросы, а даже если и сможет, то остается риск разработки системы, которая не будет помогать сотрудникам в выполнении их ежедневных бизнес-функций. И ценность такой системы, несмотря на то что формально все пожелания генерального директора будут учтены, весьма сомнительна.

Очень важную роль в обработке и анализе информации играют следующие методы.

## Образное восприятие информации

Этот метод базируется на приемах образного мышления и особенно хорош для понимания взаимодействия пользователя с системой.



Практика образного восприятия является составной частью техники быстрого чтения. Я рекомендую вам посмотреть первоисточники и материалы по технике быстрого чтения в Интернете, поскольку это позволит вам убить двух зайцев — научиться образно мыслить и быстро читать, что даст вам возможность, в свою очередь, эффективно перерабатывать огромные объемы информации в сжатые сроки. Я лично занимался в свое время в школе быстрого чтения Олега Андреева заочно, по почте. Это действительно работает.

В примере, рассмотренном выше, можно образно представить себе процесс выбора и отправки файла. Я попытаюсь объяснить, что это такое, на примере своей цепочки образов. При прочтении нижеследующего текста у вас должны возникать соответствующие картинки перед глазами.

Итак, процесс выбора и отправки файла. Когда я начинаю размышлять об этом процессе, то визуализируются примерно такие образы и сами собой рождаются вопросы. Представляю себя в качестве пользователя, сидящего на своем рабочем месте. Пользователь (регистрируется в системе? Или система использовала Windows-авторизацию?) находит в Explorer файл для отправки, щелкает на нем правой кнопкой мыши, выбирает в меню «Отправить файл через систему обмена файлами» (пока неважно, как конкретно будет выглядеть меню, но возникает вопрос локализации для разноязычных Windows). Открывается окно. Представляю себе свой монитор с этим открывшимся окном. «Смотрю» на это окно (или все-таки это подчиненное меню? Размеры окна? Расположение по центру? В точке, где находится указатель мыши?) с внутренними и внешними адресатами (кто этих адресатов заносит? Есть ли интеграция с **Active Directory**? **А если получатель может выступать и как внутренний, и как внешний адресат?**). Адресаты сгруппированы в группы. Можно развернуть и свернуть адресатов группы. Система запоминает состояние свернутости или развернутости групп, запоминает пять наиболее часто используемых адресатов и имеет возможность выбрать их сразу, без поиска в группах, выбрать адресатов по алфавиту (отдельная вкладка), система имеет строку поиска. Пользователь выбирает адресатов, отмечая чек-боксы. Вводит какой-то свой комментарий или выполняет операцию drag & drop из Outlook, при этом текст письма Outlook вставляется в поле комментария (длина поля комментария? Поддерживает ли HTML? Функция автоподписи таких сообщений?). Пользователь нажимает кнопку «Отправить». Система проверяет наличие файла в базе данных — представляю конвейер с движущейся по нему коробкой (а если другая коробка уже застряла впереди?), заполненной цветными деталями, по мере движения происходит сравнение с другими коробками (в каком порядке? Со всеми коробками сравнивается?). «Коробка» нашлась. Лампочка совпадения. Специальный сканер сканирует бар-код коробки (путь) и отправляет адресатам только распечатку этого бар-кода (квитанцию). Если «коробка» не нашлась,

система распознает (?) тип файла (а может, будут какие-то атрибуты файла, чтобы их могли задать при отправке, — например, «Название проекта») и ставит коробку на полку, где стоят такие же коробки. Специальный стикер маркирует коробку бар-кодом. Внутренним адресатам отправляется распечатка этого бар-кода (квитанция). Все квитанции проходят через специальный черный ящик — «контроль безопасности», который проверяет, может ли квитанция быть отправлена всем указанным адресатам. Если возникает проблема, то квитанция отправляется назад отправителю с пометкой «адресат не имеет права получать файлы этого типа/этого проекта/этого уровня конфиденциальности» и т. д.

Надеюсь, вы поняли основную идею этого метода. Кстати, по сути, мы с вами описывали сценарий взаимодействия пользователя с системой. Следующий шаг после того, как выполнена визуализация такого сценария, — выявить варианты использования системы/Use Cases.

Как вы уже смогли убедиться, образное восприятие помогает не только органично представить себе процесс, но и найти массу дополнительных вопросов для уточнения поведения системы.

Недавно я посмотрел видеofilm «Чудо в клетке» о сложных биологических процессах в клетках человеческого тела, о механизме работы человеческого зрения. Если вы найдете полтора часа свободного времени, рекомендую вам посмотреть этот фильм. Он наглядно иллюстрирует приемы образного представления сложной информации и сам по себе познавателен и интересен.

## «Что такое солнце?»

Этот метод наиболее полезен при выявлении понятий и их взаимосвязей. Даже если вам кажется, что вы знаете смысл понятия, обязательно проясняйте, правильно ли вы его понимаете, с экспертами предметной области.

Конечно, большинство пользователей и заказчиков живут на той же планете, на которой живем и мы с вами, но иногда попадают и «инопланетяне», которые на данный вопрос могут ответить: «это звезда класса Y в солнечной галактике, что чуть правее туманности Z» или «Солнце — это источник магнитных бурь на Земле», и нам, с нашим понятием «Солнце — источник жизни и энергии», будет очень сложно строить адекватный диалог, пока мы не договоримся о терминах.

Говорят, что Ленин с Троцким подолгу договаривались о терминах, чтобы затем в течение часа разрешать сложнейшие вопросы. Это действительно работает.

Уточняйте термины и понятия с самого начала.



В нашем примере этот метод применим следующим образом.

- ◆ Что такое гарантированная доставка?
- ◆ Что такое файл?
- ◆ Что такое публичные сети?

## Абстрагирование

Данный метод заключается в попытке посмотреть на каждый конкретный вопрос с более общей точки зрения.



В рассматриваемом примере применение этого метода позволит нам задать много новых вопросов, на которые надо искать ответы.

- ♦ Только файл или любой выделенный кусок текста в редакторе можно так послать?
- ♦ Файл и группа файлов — не одно ли это и то же?
- ♦ Адресат, сотрудник компании, клиент — это не одна и та же абстрактная сущность «контакт»? Не потребуется ли интеграция с CRM-системой? А может, лучше пересмотреть архитектуру ИТ так, чтобы данные обо всех пользователях хранились в AD, а CRM использовала бы их? И т. д.

Как вы уже поняли, дотошность и педантизм, которые могут показаться со стороны занудством, относятся отнюдь не к негативным чертам характера аналитика, а скорее к важным навыкам, необходимым для успешного выявления требований к системе.

### 3.1.2. Умение конспектировать



В ходе многочисленных контактов аналитик получает огромное количество информации. Я не рекомендую надеяться только на свою память. Как говорится, самый тупой карандаш лучше, чем самая острая память.

Для того чтобы развить в себе этот навык, рекомендую прочитать книгу Л. Ф. Штернберга «Скоростное конспектирование». Приведу короткую выдержку из этой книги.



Давайте проведем эксперимент. Возьмите карточку, показанную на рис. 6, а, покажите ее вашему товарищу и спросите, что на ней записано. Ответ будет моментальным: «Теорема Пифагора». Теперь сделайте то же самое с карточкой рис. 6, б. Ответ вы получите такой же, но вам придется подождать секунд десять для того, чтобы прочитать и осознать текст. Аналогичный опыт с карточкой рис. 6, в можно проводить только с тем, кто знает стенографию, а ответ вы получите секунд через тридцать — запись надо не просто прочитать, а расшифровать.

Почему же такая разница? Ведь записано одно и то же. Все дело в том, как записано. Исходным вариантом является текст рис. 6, б, но так писать довольно долго. Вариант рис. 6, в — зашифрованный текст. Пишется такой текст быстрее, но читается хуже. Вариант рис. 6, а — это уже обработанный для наилучшего восприятия текст, который читается моментально, да и пишется быстро. ©

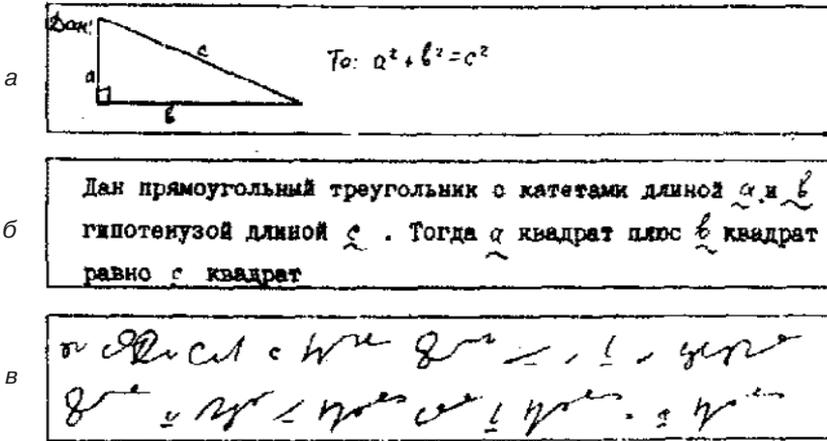


Рис. 6. Техника скоростного конспектирования (пример)

В этом основная суть навыка — уметь перерабатывать информацию на лету и фиксировать не текст, а образы. Как вы видите, этот навык очень тесно связан с образным мышлением.

В книге вы найдете много конкретных рекомендаций и приемов, которые вам помогут в профессии и в жизни. Приведу еще одну краткую выдержку.



В любом вычислительном центре можно найти подсоединенную к ЭВМ электрифицированную пишущую машинку. На ее клавиатуре в нижнем ряду клавиш находится интересный символ @ — «коммерческое at». По-видимому, его изобрели коммерсанты, которым просто некогда дописывать палочку буквы «t». На лекции избытка времени тоже нет, и этот странный символ нам очень поможет.

Возьмем самый длинный и наиболее часто употребляемый термин, оставим от него первую букву и, не отрывая пера от бумаги, «обернем» его линией. Например, в программировании для ЭВМ таким термином является «идентификатор». Сократим его до  $\text{И}$ . Обратите внимание: пишется быстрее, чем произносится, и ни с чем не спутаешь. Понятно, что выбор терминов для таких сокращений зависит от тех текстов, с которыми приходится иметь дело автору конспекта. Так, например, психолог тем же сокращением  $\text{И}$  воспользуется, скорее всего, для термина «интровертированность».

А что делать, если несколько терминов начинаются с одной и той же буквы? В этом случае есть несколько вариантов.

Использовать большие и малые буквы, например: алгоритм —  $\text{А}$ ; анализ —  $\text{@}$ .

Взять несколько начальных букв: информация —  $\text{ИНФ}$ .

Использовать латинский алфавит: реакция —  $\text{Р}$ ; раствор  $\text{С}$  (от англ. или франц. *solution*).

Почти для всех видов текстов можно рекомендовать сокращения: © — система, ☞ — элемент.

Такое сокращение легко читается. Психологи утверждают, что из минуты, затраченной на чтение, мы 58 секунд считываем промежутки между символами, поэтому сокращение ☞ читается быстрее, чем «и.», «ид-р» («идентификатор»). ©

Со временем вы выработаете свои уникальные приемы скоростного конспектирования и сами не заметите, насколько быстро они станут вашей второй натурой. Потому что это действительно очень эффективно и удобно.

### 3.1.3. Коммуникационные способности

Три кита коммуникационных способностей аналитика таковы:



1. «Мнение не бывает ни истинным, ни ложным, а лишь полезным в жизни или бесполезным; ибо оно творение Времени и с течением времени утрачивает свое влияние и значение. Так подымись же выше мнения и ищи мудрость непреходящую» © (Шри Ауробиндо).

Это одна из главных мыслей, которую, на мой взгляд, лучше сделать частью своего «я» как можно раньше. Не всегда, конечно, будет получаться следовать этой мудрости и «держать в уме» эту отстраненность. Человек эмоционален по определению, и иногда эмоции берут верх. Но я стараюсь почаще напоминать себе о том, что «мнение не бывает ни истинным, ни ложным», а лишь полезным для дела или нет. Это помогает мне оставаться спокойным и отделять факты от эмоций.

2. Вторая важная цитата, которую аналитик должен знать как «Отче наш» и выполнять: «Критикуя — критикуй мнение, а не его автора». © (Леонардо да Винчи). Созвучен этой мудрости еще один не менее важный принцип: «Критикуя — предлагай».
3. Для каждого человека самым приятным является звук своего голоса. Аналитик должен забыть на работе об этом. Его основная работа — слушать. Слушать, задавая вопросы, уточняя ответы, выуживая и конспектируя информацию.

Кроме вышесказанного, рекомендую поинтересоваться предметом «менталитет и речевой этикет наций» и узнать больше про культурные отличия разных наций. Даже если вам не придется в скором времени общаться с иностранными партнерами, вы сможете многое почерпнуть для себя, размышляя над особенностями культуры разных народов, над их способами ведения бизнеса, типичными коммуникациями и деловыми приемами. Полезно взять для себя самое лучшее из разных культур и постоянно критически пересматривать свои навыки. Кроме того, информация, которую вы почерпнете, может быть спроецирована вами на разные по темпераменту типы людей, независимо от национальной принадлежности.



Аналитик должен владеть искусством ведения переговоров. Есть очень хорошие тренинги, которые не просто помогут вам приобрести навыки переговоров, но и позволят разобраться в самом себе, выявить, над какими сторонами вашей личности вам следует поработать. В свое время я проходил интенсивный тренинг «Коммерческие переговоры» у независимого консультанта Павла Букова. Думаю, вы сможете найти информацию о нем и его тренингах в Интернете.

Еще одной важной составляющей являются навыки невербального общения. Это умение использовать язык тела, жестов, глаз. Сейчас нетрудно найти и ознакомиться с соответствующей литературой по данной теме. Настоятельно рекомендую сделать это. Такие навыки вам обязательно пригодятся.

Не открою большого секрета, если скажу, что все люди делятся на аудиалов, кинестетиков и визуалов. Первые лучше всего воспринимают информацию на слух, вторые — через чувства и ощущения, последние — зрением. Замечено, что для каждой из этих категорий характерны ключевые слова и высказывания, без которых они не могут обходиться.

- ◆ Кинестетик: «мне кажется», «я чувствую», «я в шоке!», «это ужас!» и т. п.
- ◆ Аудиал: «я думаю», «послушайте», «говорят, что...» и т. п.
- ◆ Визуал: «по-видимому», «представьте себе», «давайте рассмотрим» и т. п.

Определить принадлежность человека к этим категориям можно по взгляду и жестам.

При ответе на любой вопрос визуал обычно направляет свой взгляд вверх-влево (зрительно вспоминает при намерении сказать правду) либо вверх-вправо (когда собирается сконструировать полуправду или солгать); аудиал смотрит по горизонтали влево (при слуховом воспоминании правды) или по горизонтали вправо (намереваясь ответить так, как ему выгодно); кинестетик же смотрит вниз-вправо (правда) или вниз-влево (внутренний диалог и контроль речи). Жестикулируют они тоже по-разному: визуал обычно поворачивает ладони снизу вверх и выводит их на уровень глаз, аудиал водит руками по горизонтали на уровне груди, кинестетик использует «рубящие» жесты — сверху вниз.

Чтобы добиться максимального успеха во время ведения переговоров или интервью, необходимо разговаривать с собеседником на его языке: визуал предмет беседы лучше «видит», аудиал — слышит, кинестетик — переживает. Старайтесь понять, к какой категории он относится, и используйте соответствующие ключевые слова и высказывания.

Я рекомендую также прочитать две книги Э. Берна по психологии: «Люди, которые играют в игры» и «Игры, в которые играют люди». В них нет конкретных приемов, которые вы сможете использовать в своей профессиональной деятельности, но они позволят вам лучше понять себя и окружающих, что в конечном итоге поможет в профессиональных коммуникациях.



Всегда используйте в работе принцип планирования коммуникаций еще до начала выполнения работ. Это планирование определяет успешность и своевременность информирования «правильных» людей в «правильное» время о «правильных» вещах в «правильной» форме с целью наиболее быстрого и аргументированного принятия решений по вопросам в рамках выполнения работ. Существуют пять «золотых» правил коммуникации: 1 — «кто», 2 — «кого», 3 — «о чем», 4 — «когда» и 5 — «в какой форме» уведомляет. При планировании коммуникаций важно договориться и сформировать эти правила.

### 3.1.4. Управление эмоциональным фоном

В нейролингвистическом программировании существует практика дистанцирования, которая помогает справиться со стрессами и тревожными ощущениями и позволяет управлять своим эмоциональным фоном. Если вкратце, то методика заключается в следующем:

- ◆ создайте в воображении картинку ситуации/переживаний и удалите ее от себя. Следите за вашими эмоциями и ощущениями, старайтесь отдалить себя от переживаний. Сделайте картинку негативных переживаний маленькой, тусклой и далекой, а позитивных, наоборот, большой, яркой, близкой к вам;
- ◆ попробуйте сделать картинку плоской, объемной, черно-белой, цветной, обратите внимание на композицию, где вы в этой картинке. Переместите себя в другое место. Отдайте остальное от себя самого на картинке. Есть ли у картинки рамка? Измените ее цвет, поместите всю картинку справа от себя, а теперь слева.

Главное — в течение всех этих действий следить за своими ощущениями и чувствами. Почаще напоминайте себе, что «мнение не бывает ни истинным, ни ложным, а лишь полезным для дела или нет».

### 3.1.5. Полезные мелочи

Всегда носите с собой блокнот и ручку — когда вы погружены в решение какой-то проблемы, мозг постоянно думает над ее решением, даже если вы заняты чем-то другим. Хорошая мысль может прийти неожиданно, и будет очень обидно ее упустить. Записывайте свои идеи, какими бы бредовыми они вам ни казались на первый взгляд.



Всегда записывайте информацию при проведении интервью. После окончания интервью обязательно переводите записанное в электронный вид и получайте подтверждение от интервьюируемого сотрудника.

В идеале это должно быть формальное подтверждение по почте в виде утверждения результатов интервью, например, в форме протокола.



Заведите собственную базу выученных уроков, в которую заносите все сложные ситуации, возникающие в вашей жизни. Анализируйте эти ситуации и выработайте персональные лучшие практики. Я веду такую базу в течение нескольких лет с помощью инструмента управления требованиями — Rational Requisite PRO. В ней есть два типа требования: проблема и хорошая практика. Происхождение любой хорошей практики (рекомендации к самому себе на будущее) можно проследить до проблемы, из анализа которой эта практика возникла. Хорошие практики имеют атрибут «Область знаний» со значениями: «Личное развитие», «Управление», «Коммуникации», «Требования», «Моделирование».

Английский язык — ваши ворота в мир новейших изысканий в области информационных технологий. Уровень знаний должен позволить вам читать специальную литературу без словаря.



Планируйте собственное профессиональное и личностное развитие. Создайте план с четкими контрольными сроками (например, с помощью Microsoft Office продуктов: Project, Excel, в Outlook — неважно) и приучите себя выполнять поставленные задачи к этим срокам. Создайте свое собственное хранилище знаний. Продумайте его структуру и заполняйте книгами, которые вы будете читать, и интересными материалами ваших производственных проектов. Раз в полгода пересматривайте это хранилище, удаляйте лишнее и отмечайте собственный прогресс и рост.

При изучении книг активно используйте рукописные значки на полях книг, цветные карандаши и маркеры для того, чтобы отметить важные и интересные мысли.

Познакомьтесь с программным инструментом **Mind Map** компании **Mind Technologies** (<http://www.visual-mind.com>). Он позволяет строить карты размышлений и принятия решений. Я обращаюсь к этому инструменту редко — предпочитаю использовать сразу инструмент для управления требованиями, чтобы иметь больше возможностей по управлению информацией.

Инструмент для создания презентаций **Microsoft Power Point** — неплохой способ первичной группировки информации для обсуждения с нетехническими специалистами. Используйте цвет фона, значки в углу презентации, чтобы категоризировать информацию.

Всегда делитесь полученными знаниями с вашими коллегами. Делайте это бескорыстно, как в пословице: «Делай добро и бросай его в воду...»

## 3.2. Профессиональные и специальные навыки

Разные компании выбирают разные способы организации собственной работы. В большинстве современных компаний используется матричная структура организации, когда существуют функциональные отделы, в которых сотрудник

числится и профессионально развивается, и проектный офис, отвечающий за управление портфелем проектов компании. При старте нового проекта запрашиваются требуемые человеческие ресурсы, которые попадают в распоряжение менеджера проекта. Так аналитик попадает в проект. Дальнейшее повествование ведется с точки зрения аналитика, попавшего и работающего в проекте. Для успешного начала работы в производственных проектах компании-разработчика ПО вы должны обладать следующими навыками (табл. 2).

**Таблица 2.** Профессиональные и специальные навыки младшего аналитика

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
<p><b>Младший аналитик</b></p>	<p>Книгу Дина Леффингвелла и Дона Видрига Managing Software Requirements я называю «библией аналитика». Помимо теоретической информации, вы найдете в ней ряд конкретных примеров и приемов проведения интервью, «мозговых штурмов» и т. д.</p> <p>Еще одна, более поздняя и не менее «евангелистская», книга в области анализа — «Разработка требований к программному обеспечению» Карла И. Вигерса.</p> <p>Иметь общее представление о различных методологиях разработки ПО.</p> <p>Знать основы RUP — дисциплину Requirements.</p> <p>Знать и уметь применять UML — Use Case-модель, уметь строить Domain Object Model.</p> <p>Понимать основные принципы объектно-ориентированного проектирования и моделирования.</p> <p>Рекомендую также почитать статьи по анализу на сайте <a href="http://www.interface.ru">www.interface.ru</a></p>	<p>Выявлять заинтересованных лиц (ЗЛ).</p> <p>Управлять ожиданиями ЗЛ.</p> <p>Проводить совещания.</p> <p>Проводить интервьюирование.</p> <p>Проводить анкетирование.</p> <p>Проводить «мозговые штурмы».</p> <p>Уметь определять границы системы.</p> <p>Уметь выделять подсистемы и определять их границы.</p> <p>Уметь выявлять требования типов:</p> <ul style="list-style-type: none"> <li>• ответы и собранная информация;</li> <li>• запросы заинтересованных лиц;</li> <li>• глоссарий;</li> <li>• стандарты и ГОСТы;</li> <li>• характеристики аналогичных/наследуемых систем;</li> <li>• бизнес-требования;</li> <li>• бизнес-правила;</li> <li>• концепция создания и развития продукта (BVISION);</li> <li>• ограничения и допущения;</li> <li>• концепция системы (TVISION);</li> <li>• пользовательские требования;</li> <li>• функциональные требования;</li> <li>• функции системы/варианты использования/прецеденты (Use Cases);</li> <li>• нефункциональные требования;</li> <li>• требования к пользовательскому интерфейсу;</li> </ul>

Продолжение ⇨

(Продолжение)

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
		<ul style="list-style-type: none"> <li>• требования к взаимодействию с внешними системами.</li> </ul> Выявлять функции системы (Use Cases), моделировать поведение системы. Уметь строить трассировки/прослеживать требования. Понимать основные принципы тестирования. Знать английский язык на уровне, достаточном для чтения технической литературы

### 3.2.1. Использовать специальные инструменты

Для управления требованиями и моделирования существуют специальные программные инструменты. Я в своей практике использую в основном инструменты линейки компании Rational (<http://www.rational.com>), для управления требованиями — **Requisite PRO**, для моделирования — **Rational Rose**. Эти инструменты интегрируются.

Вокруг инструментария всегда слишком много ненужных споров и шумихи. Кому-то нравится одно, кому-то — другое. Выбор того или иного инструмента непринципиален. Если есть понимание, что именно надо сделать с требованиями или с моделью, то не имеет значения, как будет выглядеть инструмент. Главное, чтобы он имел все нужные функции, которые позволят выполнять работу — проводить системный анализ и разработку требований. Кроме продуктов Rational, мне, например, нравится инструмент Enterprise Architect компании Sparx (<http://www.sparx.com>). Да и новые инструменты от IBM (компания Rational вошла в состав IBM) — Rational Software Architect и др. — тоже вполне интересные и удобные.

Обычно в компании уже выбраны соответствующие инструменты, и вам надо будет их изучить, если вы не сталкивались с ними раньше. Но, повторяю, не надо этого бояться. Главное — не инструмент, а идеи. А идеи после прочтения этой книги у вас, я надеюсь, будут.



Идеи, которые будут описаны в этой книге, могут быть реализованы практически на любом инструменте управления требованиями и моделирования. Поэтому вместо употребляемого мною RequisitePRO можете подставлять название того инструмента управления требованиями, который используете вы. Повторюсь, обычно выбор инструмента не играет никакой роли, кроме случаев описания конкретной функциональности RequisitePRO, которой может не быть в других инструментах управления требованиями.

### 3.2.2. Понимать методологии разработки ПО

Каждый аналитик должен иметь общее представление о существующих методологиях разработки ПО. В этом разделе представлен краткий сравнительный анализ некоторых основных методологий разработки ПО с моими выводами.

Все методологии условно можно разделить на «легкие» и «тяжелые». «Тяжелая» методология, как правило, достаточно жестко регламентирует роли, их ответственность, состав и даже структуру артефактов, последовательность выполняемых действий. К таким методологиям можно отнести RUP [5], MSF [1]. «Легкая» методология имеет короткий, по сравнению с «тяжелой», перечень активностей, характеризуется ранним прототипированием, наличием значительно упрощенных спецификаций требований и «живыми» коммуникациями. К «легким»/«гибким» методологиям можно отнести Agile, XP. Между ними я бы поместил спиральную разработку по Boehm и Isonix [14].



---

Конечно, все эти деления условны: любая «тяжелая» методология может быть адаптирована для простых проектов и небольших команд и оказаться в разряде «легких», так же «легкую» методологию можно привести к «тяжелой», если заказчику потребуется, например, создать формальные спецификации по ГОСТу или если коммуникации в проекте затруднены из-за объективных причин (распределенная команда, удаленный заказчик, система разрабатывается несколькими командами из разных компаний).

---

Сегодня в среде разработчиков ПО есть ярые приверженцы и сторонники каждой из этих методологий, готовые отстаивать «правильность и эффективность» именно «своей» методологии. В принципе, любой проект можно выполнить по любой из методологий с определенной долей вероятности успеха выполнения проекта. Но, на мой взгляд, каждая из методологий имеет свои отличительные черты и условия, в которых она принесет максимальную выгоду компании.

Например, многие участники сообщества разработчиков Agile в России (<http://www.agilerussia.ru/>) считают, что методология Agile может успешно применяться на крупных проектах — при разработке сложных систем с большой проектной командой. Но я бы не рекомендовал разрабатывать «коробочный» продукт по методологии Agile, как не стал бы использовать RUP для разработки in-house-приложения по автоматизации учета трудозатрат. Не стоит надеяться и на Isonix, если речь идет о системе сводной банковской отчетности, лучше подумать об использовании RUP [5]. И наоборот — я не стал бы применять RUP при создании подсистемы «личный кабинет» корпоративного портала, скорее всего, здесь целесообразно применить методологию Isonix.



---

Я рекомендую всегда вдумчиво подходить к выбору методологии разработки ПО, понимать, почему процесс разработки построен именно так. Оптимально ли это? Не надо ли его изменить? И если ИТ-стратегия и ИТ-архитектура должны служить целям бизнес-стратегии и обеспечивать потребность бизнес-модели в информации, то методология разработки ПО должна учитывать следующие аспекты:

- ♦ какой продукт мы разрабатываем — «коробочный», тиражируемый или разово поставляемое решение;
- ♦ есть ли возможность построить **Scrum Team** или у нас сложные коммуникации и требуется серьезное управление рисками;
- ♦ достаточная ли квалификация у проектной команды, чтобы выполнять работы без создания формальных спецификаций, не требует ли заказчик исчерпывающих спецификаций для будущего развития системы, не понадобится ли сертифицировать систему в дальнейшем;
- ♦ разрабатываемая система ориентирована на взаимодействие с пользователем либо с другими системами и т. д.

Только после ответа на все эти вопросы стоит приступать к выбору методологии разработки ПО. За основу можно взять одну методологию и дополнить ее практиками из других.

Я рекомендую вносить гибкость в «тяжелые» методологии за счет использования практик управления, эффективных коммуникационных приемов и правил групповой работы «легких» методологий и «придавать солидности» «легким» методологиям путем применения некоторых «тяжелых» спецификаций требований и практик управления рисками и разработкой архитектуры в тех случаях, когда это действительно необходимо.

Итак, рассмотрим вкратце некоторые из упомянутых методологий.

## Rational Unified Process (RUP)

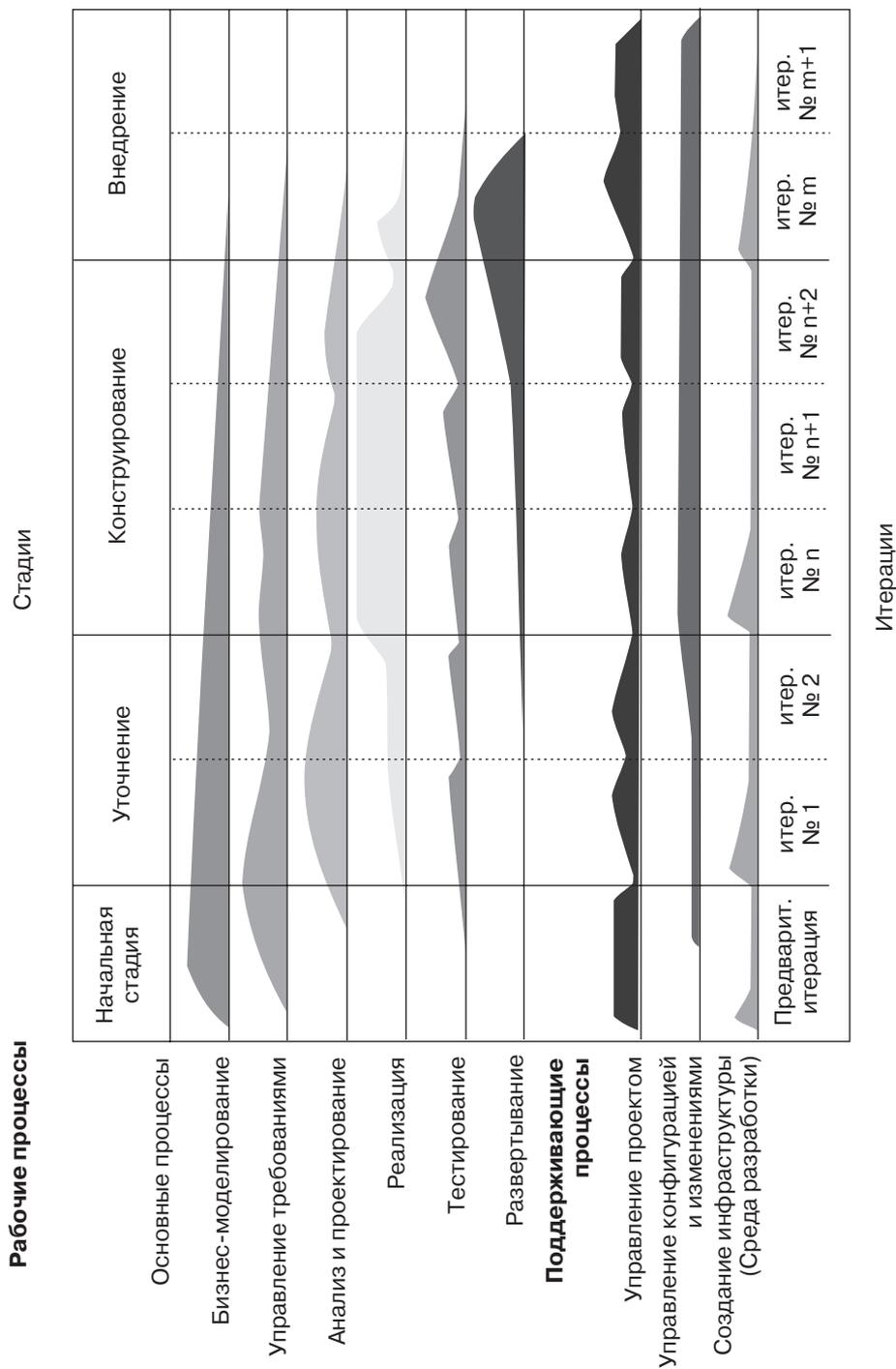


**Rational Unified Process — унифицированный процесс разработки ПО компании Rational** с однозначно выраженными рекомендациями по разработке ПО, которые включают в себя перечень всех необходимых деятельности, выполняемых проектными ролями на каждой итерации, и шаблонов артефактов.

Согласно RUP, **жизненный цикл разработки ПО состоит из нескольких итераций**. Каждая итерация включает в себя организованный набор последовательных действий по бизнес-моделированию, разработке требований, анализу и дизайну, разработке, тестированию и развертыванию, в значительной степени зависящих от того, на каком этапе (фазе) находится разработка.

Итерации в фазе задумки и оценки фокусируются на работах по управлению, требованиям и дизайну. В фазе конструирования фокус смещается на дизайн, разработку и тестирование, в фазе передачи — на тестирование работоспособной версии системы. Итерации должны управляться, с жестким соблюдением и контролем границ проекта и расписания работ.

На каждой фазе, от инициации до передачи, выполняется цикл работ с последовательным расширением деятельности по мере узнавания информации и со смещением акцентов на области, специфичные для каждой конкретной фазы [5] (рис. 7).



**Рис. 7.** Итерации в Rational Unified Process

Методология RUP [5] (рис. 8) требует выполнения большого объема работ, создания и поддержки множества артефактов значительным количеством проектных ролей.

Методология RUP эффективна для применения в больших неоднородных и распределенных командах, выполняющих длительные по срокам сложные проекты, особенно если предполагается постоянное развитие продукта, которое может выполняться разными компаниями-разработчиками или разными командами в рамках одной компании-разработчика ПО.

## Microsoft Solution Framework (MSF)



Microsoft Solution Framework (MSF) [1] не является методологией разработки ПО. Она предлагает подходы, основанные на определенной совокупности принципов, моделей, дисциплин, руководств и методик для проектов различной степени сложности, ориентированных на поставку решений. В каждом конкретном проекте предлагаемый подход нужно адаптировать в соответствии с процессом поставки решения в этом проекте.

В настоящее время (2010 год) Microsoft предлагает два подхода к разработке приложений: MSF для гибкой разработки приложений (Agile Software Development) и MSF для совершенствования CMMI-процессов (CMMI Process Improvement).

Ниже приводится краткое описание ключевых элементов MSF. Более подробно о MSF можно прочитать в книге М. Тернера «Основы Microsoft Solution Framework».

- ◆ **Основные принципы** — базовые элементы MSF. Описывают значения и стандарты, общие для всех элементов MSF. Определяют, каким образом члены проектной команды должны совместно действовать, чтобы успешно выпустить решение. Определяют взаимодействие команды с заинтересованными лицами. Основные принципы нацеливают команду на максимальный успех.

В MSF определены девять ключевых принципов:

- ◆ открытый обмен информацией;
  - ◆ согласованное в команде общее видение проекта;
  - ◆ наделение членов команды полномочиями;
  - ◆ строгая подотчетность и общая ответственность;
  - ◆ постепенное повышение отдачи;
  - ◆ гибкость, готовность к изменениям;
  - ◆ вложение ресурсов в качество;
  - ◆ обучение на основе опыта;
  - ◆ партнерские отношения с заказчиками.
- ◆ **Установки** — правила, которые должны быть приняты членами проектной команды в качестве норм их личного поведения. Определяют подход

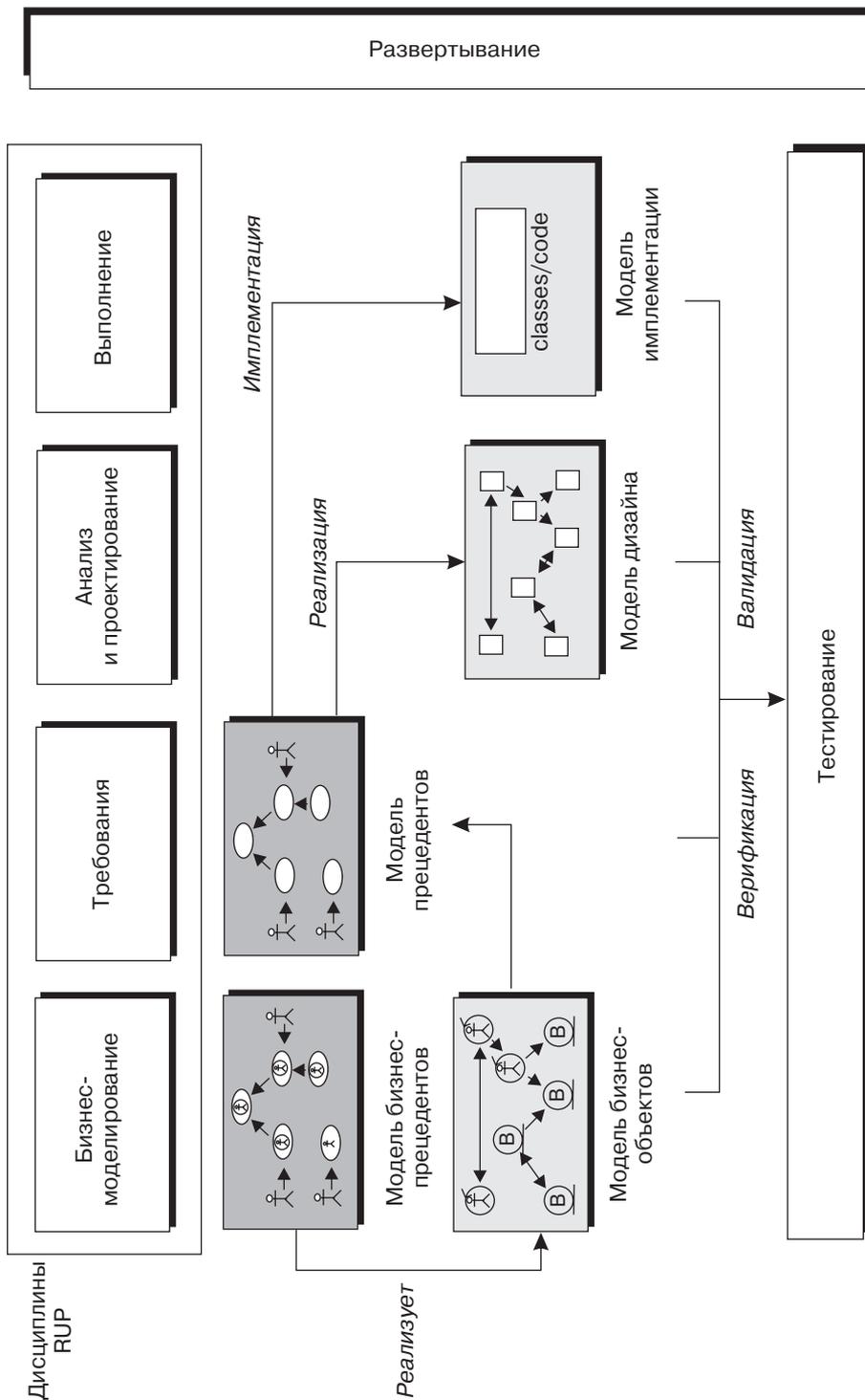


Рис. 8. Дисциплины и модели, используемые в RUP

каждого члена команды к поставке решения, нацеливают участника команды на успех.

Примеры установок:

- ◆ способствовать равенству в команде;
  - ◆ концентрироваться на экономической отдаче;
  - ◆ видеть перспективу решения;
  - ◆ постоянно учиться;
  - ◆ работать над качеством обслуживания;
  - ◆ выполнять обязательства;
  - ◆ поощрять профессионализм.
- ◆ **Модели** – схематические описания структур проектной команды и процессов. В MSF используются две модели – команды и процессов.

В модели команды определяются состав команды и активность каждого ее участника.

В модели процессов описываются основные проектные активности. Модель процессов делит ЖЦ проекта на фазы, описывает для каждой фазы создаваемые артефакты и определяет прохождение контрольных точек.

На рис. 9 показана модель команды MSF [2].



**Рис. 9.** Модель команды MSF

- ◆ **Дисциплины** – области использования подхода (управление проектом, управление рисками и т. п.).
- ◆ **Проверенные методики** – это практики, работоспособность которых подтверждена в реальных условиях. Связаны с определенной практической областью.

Примеры проверенных методик:

- ◆ предоставление командам возможности работать в одном месте;
  - ◆ определение и проектирование решения с участием всех ролей;
  - ◆ мотивирование команд для повышения производительности.
- ◆ **Рекомендации** — набор рекомендуемых практик. MSF предлагает использовать итеративный процесс. В конце каждой итерации должен появляться стабильный релиз. На рис. 10 отражена последовательность выполнения фаз.

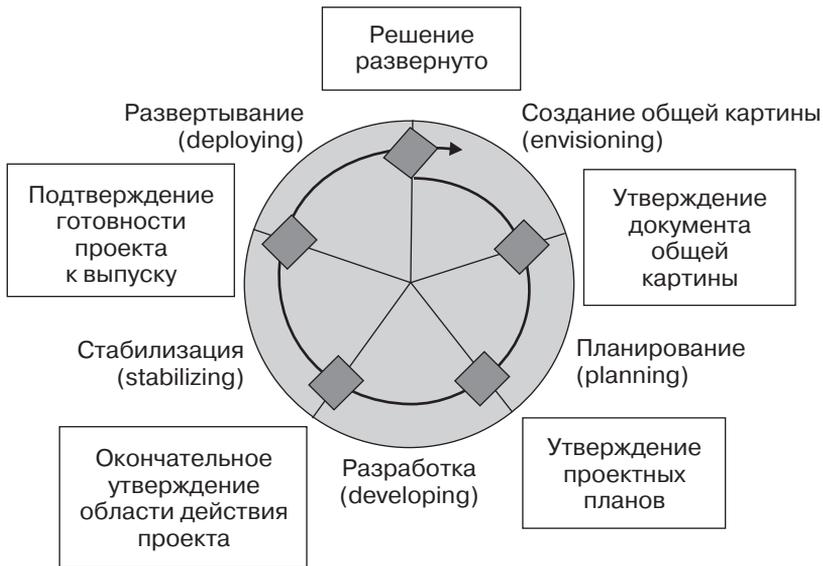


Рис. 10. Модель процесса MSF

## Iconix



Iconix — методология, близкая по смыслу к методологии RUP. Очень часто ее называют «упрощенный RUP» [21].

Девиз Iconix — *Agile without being Extreme* © («Гибкость без экстремальности»).

Процесс Iconix (рис. 11) — управляемый моделью гибкий процесс объектного моделирования. Он фокусируется на областях, которые находятся между Use Cases и конечным программным кодом. Эта методология обращает особое внимание на то, какие действия необходимы для качественного анализа и дизайна системы после выделения Use Cases.

Итеративность в Iconix — это последовательность действий, направленных на получение и поставку прототипа продукта/системы пользователю.

Все диаграммы взяты с оригинального сайта <http://www.iconixsw.com>.

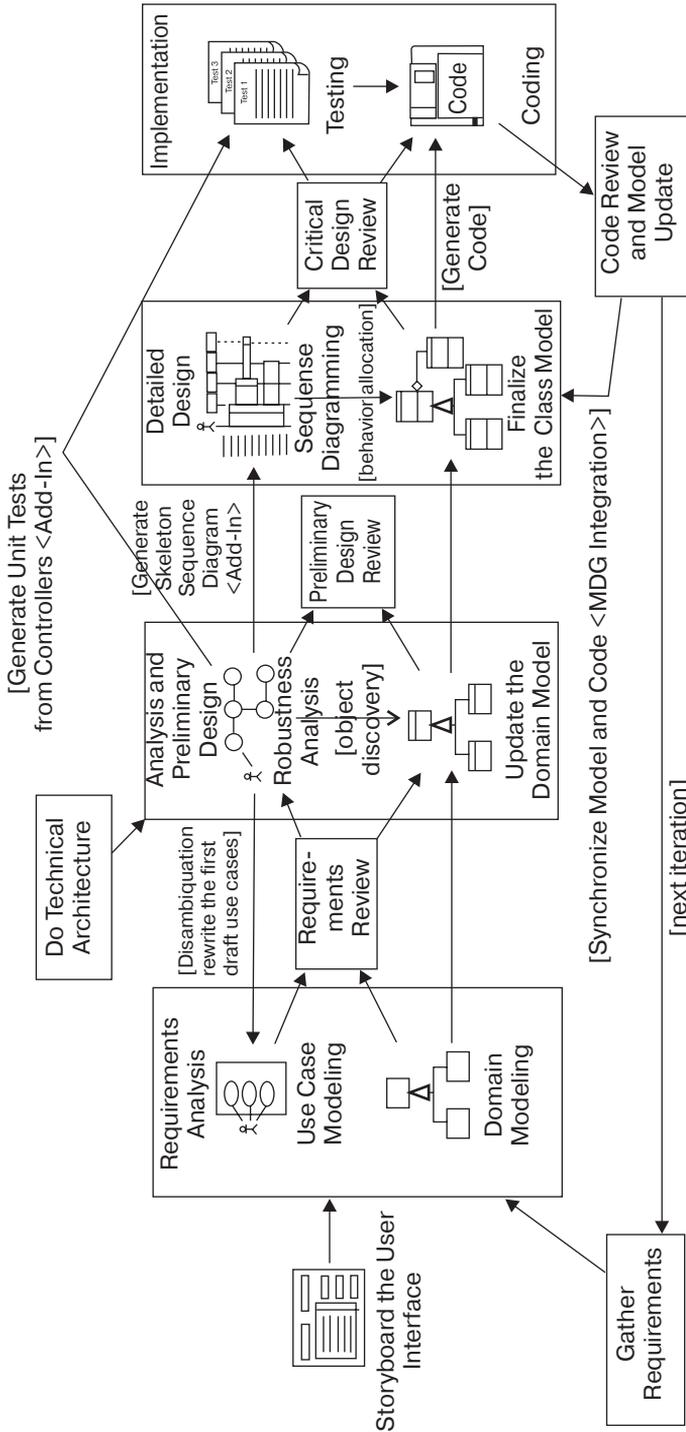


Рис. 11. Основные элементы процесса по IcoSix

В отличие от RUP [5] процесс разработки ПО по Iconix [21] ограничивается минимальным набором артефактов, необходимых для понимания динамической и статической составляющих системы (рис. 12).

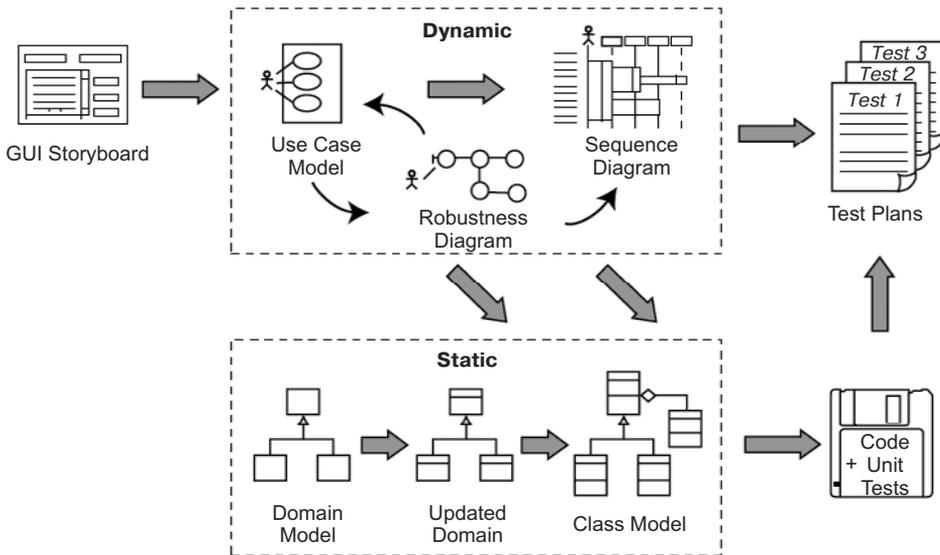


Рис. 12. Артефакты процесса разработки ПО в Iconix

К тому же Iconix акцентируется на продукте с точки зрения пользователя системы путем разработки прототипов интерфейсов и использования этих прототипов в качестве исходной точки для функционального моделирования и построения модели предметной области и ориентируется на поставку выполняемого прототипа продукта/системы.

Благодаря использованию разработки управляемой моделью при минимально необходимом документировании обеспечивается максимальная гибкость, в то же время на базе разрабатываемых моделей всегда можно построить детальную спецификацию.

С другой стороны, данный подход не исключает формирования и при необходимости согласования таких документов, как спецификации требований, ТЗ и задания на разработку.

Еще одно отличие от Agile — Iconix в большей степени реализует подход к разработке управляемой моделью и, как следствие, объектно-ориентированное моделирование, что не мешает, впрочем, разрабатывать Agile проекты с использованием Iconix (<http://iconixprocess.com/books/agile-development/>).

Методология Iconix может применяться для проектов любого уровня сложности при условии, что развитие продукта не распределяется между разными компаниями-разработчиками. Для проектов, больших по объему и сложных одновременно, имеет смысл дополнять методологию Iconix [21] практиками и артефактами RUP [5].

# Agile



Гибкая методология по экстремальной разработке ПО (рис. 13, 14). Девиз методологии — *It's about common sense* © («Это вопрос здравого смысла»). Основывается на понятии stories (истории) в качестве основного механизма выяснения и контроля требований проекта. Под историями понимается все, что может быть обсуждено с заказчиком.

Суть понятия «история» состоит в том, что каждая история как основной механизм управления требованиями проекта служит не для документации требований, а скорее для напоминания заказчику о наличии некоторых моментов (историй) для дальнейших обсуждений продукта с командой. Акцент на вовлеченности заказчика не только в процесс сбора требований, но и в процесс разработки.

Методология не основывается на формальном подходе к организации требований, практически не использует объектно-ориентированное моделирование, хотя и не запрещает этого. Существует несколько разновидностей методологии, которые не рассматриваются в настоящем документе.

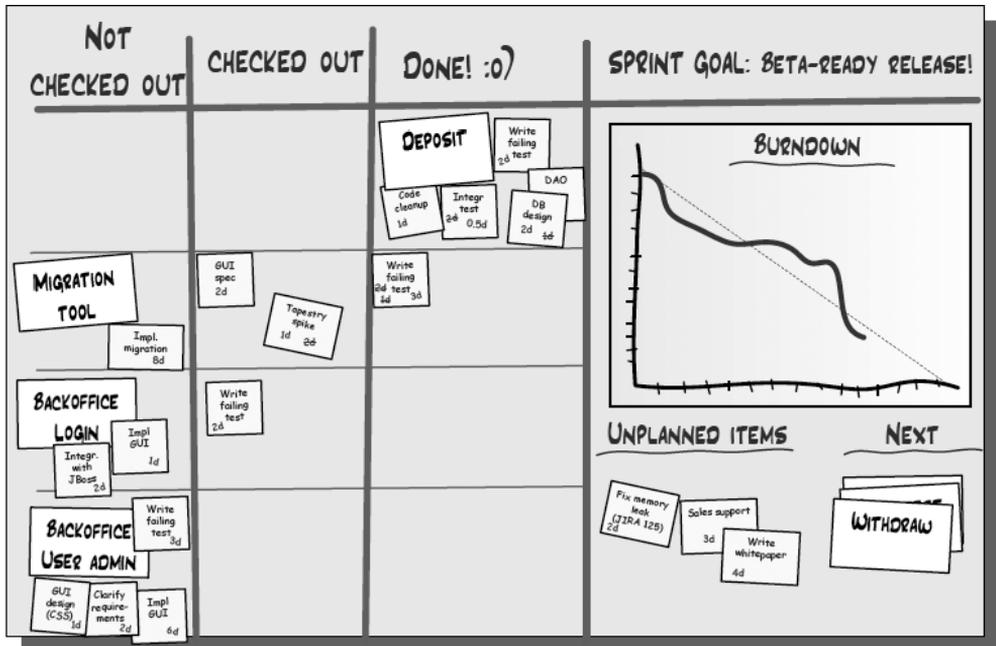


Рис. 13. Agile Dash Board

Итеративность в Agile построена по принципу максимальной частоты поставки прототипов системы (от нескольких недель до нескольких месяцев) с предпочтением наиболее короткого времени между поставками.

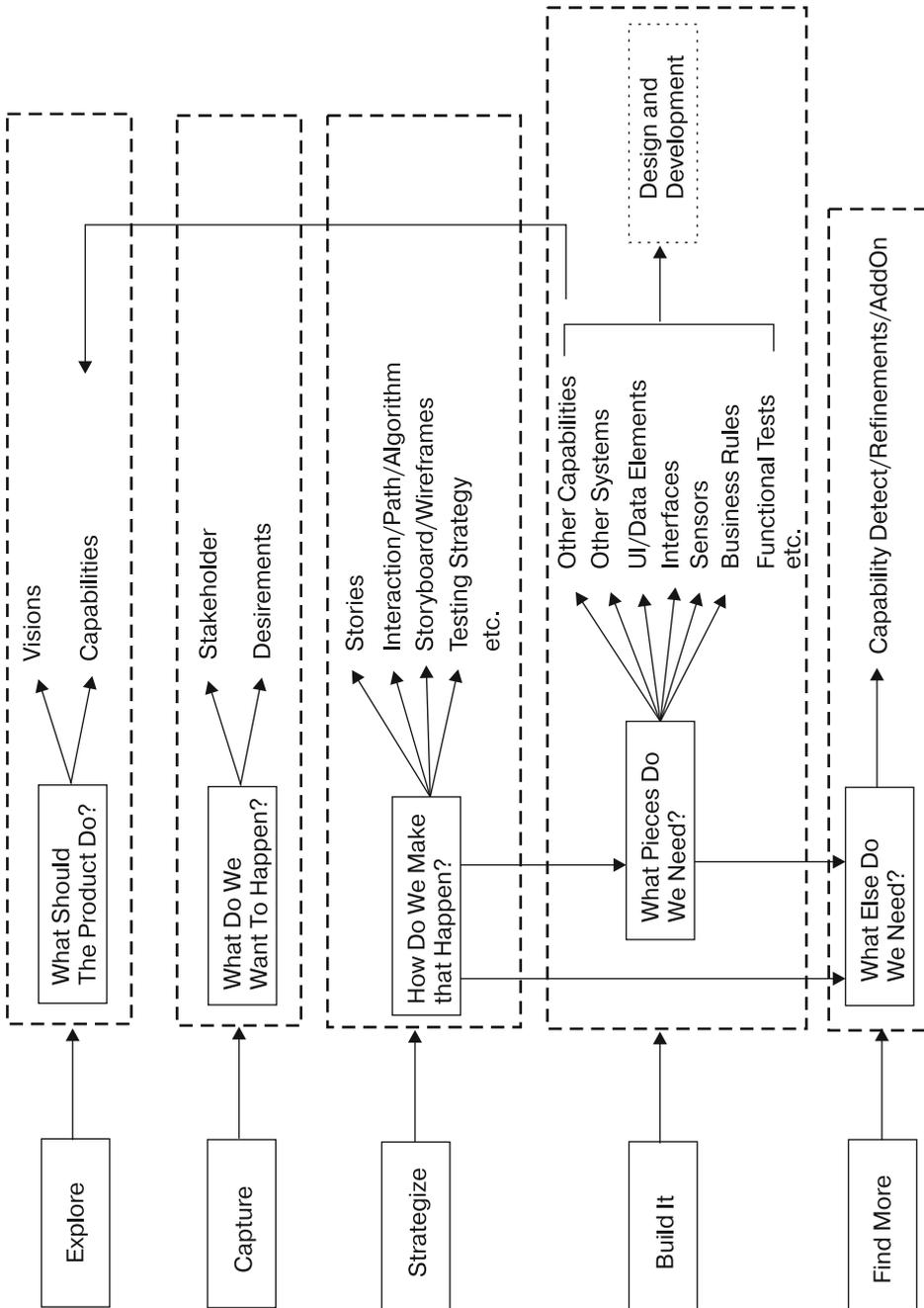
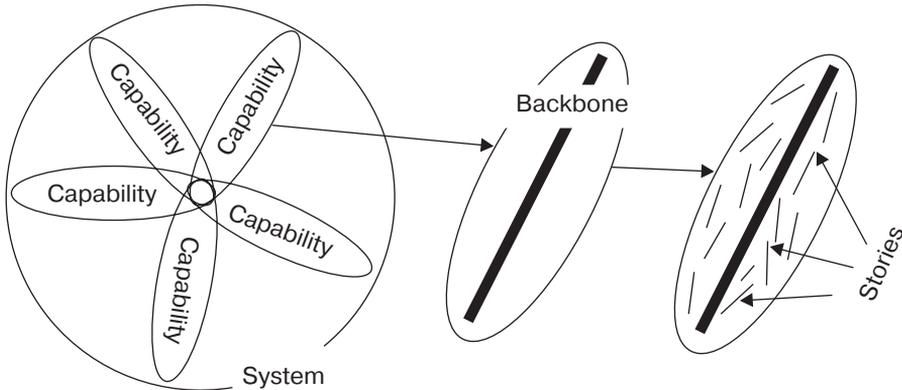


Рис. 14. Основные элементы процесса по Agile

В отличие от RUP и Iconix процесс разработки ПО по Agile не формализует набор артефактов, необходимых для понимания динамической и статической составляющих системы, а только рекомендует некоторые основные информационные сущности, вокруг которых строится разработка (рис. 15).



**Рис. 15.** Информационные сущности процесса разработки ПО в Agile

Артефакты как таковые не выделяются в виде спецификаций требований или других спецификаций, вместо этого создается заранее структурированный раздел обсуждений (например, в wiki), где в процессе анализа историй фиксируются знания о предметной области, которые и являются основой для дальнейшей разработки системы.

Методология Agile идеально подходит для проектов разработки, где не требуется дальнейшее развитие продукта, а поддержкой продукта будет заниматься компания-разработчик.

Методология Agile также требует наличия высокопрофессиональной команды открытых и инициативных людей (scrum team) с ярко выраженным лидером (scrum master).

## Extreme Programming (XP)

Одна из разновидностей Agile-методологий.




---

Экстремальное программирование предполагает разработку по мере поступления информации.

Методология не основывается на формальном подходе к организации требований, равно как не базируется на формально определенных подходах к разработке архитектуры и/или функциональности, хотя и не запрещает этого. Часто все требования к системе находятся только в голове (в переписке) разработчика, который учитывает их по мере разработки кода.

---

Методология не предполагает объектно-ориентированного моделирования — часто модель классов находится непосредственно в «голове» разработчика и сразу реализуется исходным кодом.

Итеративность построена по принципу максимальной частоты поставки прототипов системы — по мере выяснения новых требований к системе сразу и непосредственно правится исходный код продукта/системы. По факту готовности и минимальной проверки код передается заказчику для тестирования и выяснения дальнейшей функциональности.

Артефакты XP: переписка с заказчиком, другие неформальные артефакты или неформализованная документация, исходный код и рабочий прототип системы/продукта.

Методология XP может применяться для небольших проектов разработки, где достаточно понятна и прозрачна функциональность, не требуется дальнейшее развитие продукта сторонними компаниями-разработчиками, а поддержкой продукта также будет заниматься компания-разработчик.

Методология XP требует наличия высокопрофессиональной и сработанной команды, способной решать любые задачи на уровне создания кода без предварительного моделирования поведения и структуры системы с параллельным его обсуждением. Продукты, выполненные по такой методологии, часто зависят от конкретных разработчиков, что приводит к появлению незаменимых экспертов в компании.

## Спиралевидная разработка Воеhm

Основные принципы методологии:

- ◆ не детализировать спецификации до того, как будут стабилизированы высокорискованные элементы системного дизайна и архитектуры;
- ◆ прототипировать систему на каждом этапе — основное мероприятие по предотвращению рисков;
- ◆ выделять главное — детали не важны до тех пор, пока их незнание не становится критическим для дальнейшей разработки проекта.



---

Иллюстрация методологии по Воеhm представлена на рис. 16 (<http://it4business.ru/img/swebok/slm/5.gif>).

---

## Общие выводы

Для маленьких по объему проектов с очевидной функциональностью, которые разрабатываются для одного клиента или для собственных потребностей компании, которые не будут дорабатываться или будут дорабатываться по минимуму одной и той же командой, имеет смысл рассматривать методологию XP.

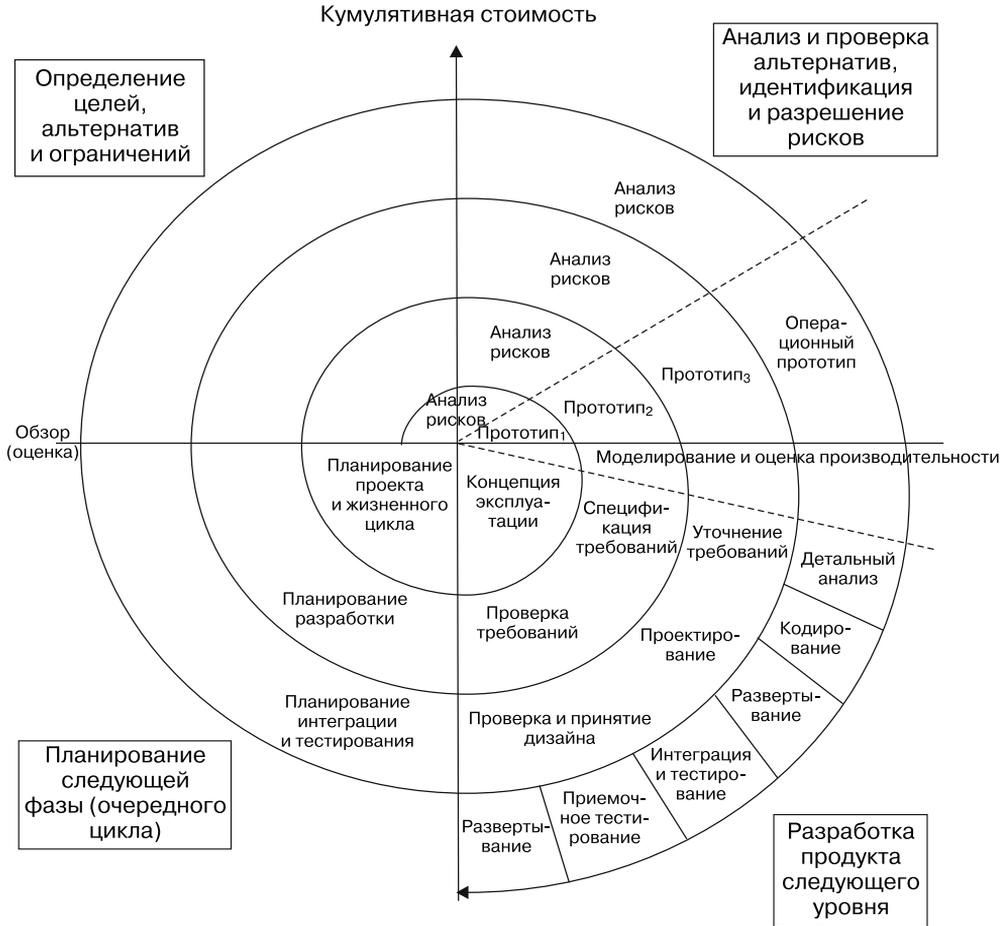


Рис. 16. Спиралевидная модель Боэма

Для более сложных по функциональности и долгосрочных проектов, которые разрабатываются для нескольких клиентов и/или на длительную перспективу, но которые не будут дорабатываться или будут дорабатываться по минимуму одной и той же небольшой командой, организованной как scrum team, подходит методология Agile.

Для больших по объему, длительности и сложных проектов, «коробочных» продуктов, которые разрабатываются на длительную перспективу, целесообразно применение методологии Iconix [21] или RUP [5].



Спиралевидная модель разработки Boehm снимает значительное количество рисков и при этом является максимально гибкой и быстрой с точки зрения скорости разработки ПО, поэтому я рекомендую выбирать ее в качестве каркаса методологий, располагая «вдоль» спирали деятельности Iconix [21] и RUP [5] и дополняя их хорошими практиками Agile.

### 3.2.3. Выявлять ожидания ЗЛ и управлять ими

Когда аналитик попадает в проект, менеджер проекта совместно со спонсором, как правило, уже создал устав проекта. Наиболее интересующая информация из устава проекта для аналитика — обоснование бизнес-потребности в выполнении проекта, бизнес-требования к продукту, ограничения и допущения, перечень заинтересованных лиц. В этом разделе мы рассмотрим практики работы с ЗЛ.

#### Выявлять ЗЛ и их ожидания

Первое, что должен сделать системный аналитик в проекте, — понять бизнес-потребность/бизнес-обоснование выполнения проекта и совместно с менеджером проекта определить тех заинтересованных лиц, которые напрямую ожидают получения измеримой выгоды от конечного результата проекта.

Рекомендую аналитику уже на этом этапе занять активную позицию и постараться провести краткое интервьюирование заинтересованных лиц на предмет правильности и полноты сформулированной в уставе проекта информации об обосновании бизнес-потребности в выполнении проекта. Здесь стоит вспомнить про типы различных архитектур, ИТ-архитектуру заказчика как конечную цель для встраивания разрабатываемого в проекте решения. Очень важно попытаться понять, куда и как заказчик планирует развивать ИТ-архитектуру, — весьма вероятно, что среди ЗЛ вы не найдете ИТ-директора компании заказчика. Стоит поговорить с менеджером проекта (МП) и убедить его в необходимости войти в состав ЗЛ. Если разрабатываемая система должна интегрироваться с какими-то системами собственной разработки заказчика, полезно включить в список заинтересованных лиц архитектора этих систем со стороны заказчика.

Часто в уставе проекта фиксируется следующая информация о заинтересованных лицах: ФИО, должность, причина заинтересованности в проектах. Я советую аналитику попытаться убедить менеджера проекта расширить эту информацию как минимум полями: приоритет удовлетворения ожиданий (1... 5), по каким вопросам принимает решение, кто замещает в случае недоступности, телефон, e-mail.

После выявления заинтересованных лиц следует начать предварительную подготовку к интервьюированию ЗЛ. Шаблон «Запросы заинтересованного лица» вы можете скачать на сайте книги, а также найти в приложении. В нем подробно описано назначение разделов и приведены вопросы, которые рекомендуется задавать аналитику для заполнения каждого раздела.



---

Настойчиво рекомендую вам обращаться к шаблонам, на которые я буду ссылаться, так как в описании их разделов часто содержатся методологическая информация и конкретные практики по заполнению.

---



Здесь приведу только возможную структуру этого документа.

*Описание заинтересованного лица или пользователя.*

*Исследование проблем заинтересованного лица.*

*Понимание рабочего окружения пользователя.*

*Подтверждение понимания проблем.*

*Вклад аналитика в определение проблем заинтересованных лиц (подтвердить или опровергнуть предположения).*

*Исследование (оценка) предлагаемого решения (если применимо).*

*Исследование (оценка) перспектив.*

*Исследование (оценка) надежности, производительности и поддержки.*

*Другие требования.*

*Резюме.*

*Сводка аналитика.*

При создании этого шаблона за основу были взяты рекомендации RUP. Вы можете упростить шаблон и оставить только наиболее важные разделы. Не нужно стремиться к тому, чтобы заполнить все разделы для каждого заинтересованного лица. Например, раздел «Исследование (оценка) надежности, производительности и поддержки» не всегда может быть заполнен при интервьюировании представителей бизнес-дивизиона. В данном случае более полезно интервью технических специалистов заказчика.

Основной идеей этого документа является описание проблем заинтересованного лица для дальнейшего анализа и выявления требований к разрабатываемой системе. Вы должны фиксировать всю информацию с позиции ЗЛ, не искажая ее и не производя никаких предварительных обработок. Не бойтесь записать «лишнее». Постарайтесь отождествить себя с ЗЛ в момент интервью и сопереживать ему. Вы должны почувствовать, что ему мешает в работе, почему и в чем создание системы ему поможет.

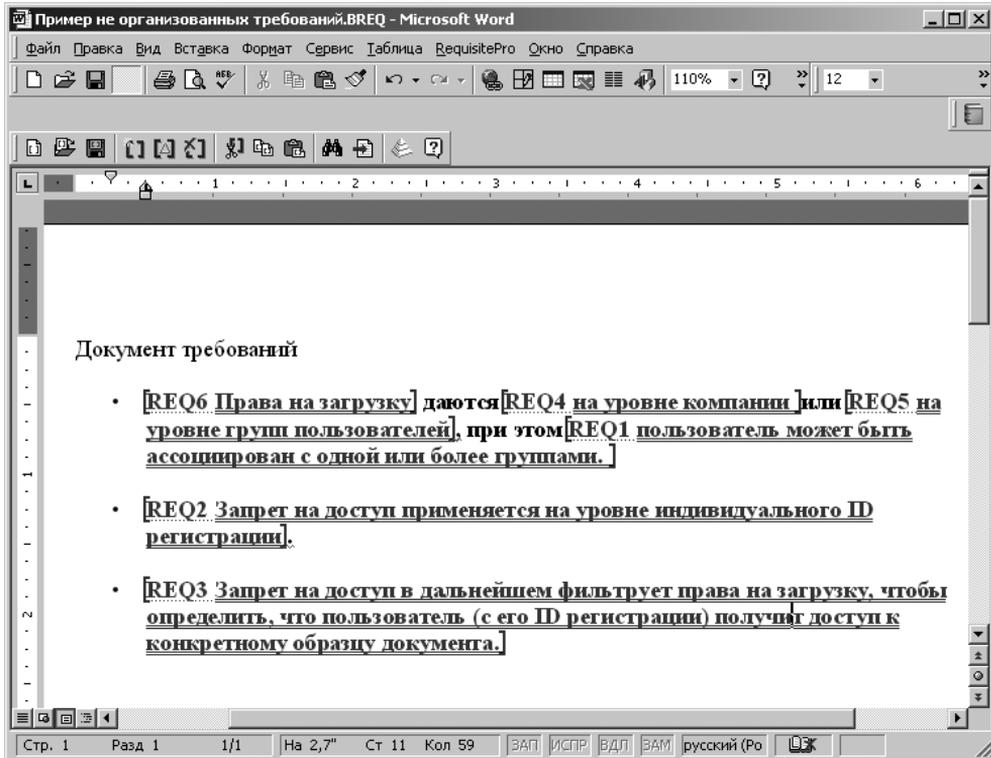
Если вы качественно проведете эту работу, проектная команда получит много новой информации, которой нет в уставе проекта. Возможно даже, что после интервьюирования заинтересованных лиц цели выполнения проекта изменятся.

## Управлять ожиданиями ЗЛ

Для того чтобы быть уверенными в том, что все ожидания заинтересованных лиц будут учтены в разработке системы, все противоречия разрешены, ожидания приоритизированы и реализованы в соответствии с их приоритетом и важностью, необходимо управлять ожиданиями.

В инструменте Rational Requisite PRO существует возможность импорта документов с последующей разметкой текста в виде требований (рис. 17).

При этом REQ1–REQ6 — требования заранее определенного типа в проекте требований Requisite PRO.



**Рис. 17.** Импорт документа MS Word в Rational Requisite PRO с разметкой требований

Для эффективного управления собранными требованиями документы, полученные в результате интервьюирования ЗЛ, импортируются в Rational Requisite PRO, в них выделяются требования типа «ожидания заказчика». Структурирование и иерархия требований остаются на усмотрение аналитика (рис. 18).



Я рекомендую в документах вида «Запросы заинтересованного лица (Сидоров В. А.)» выделять тип требования — «Ответ» (Answer — ANSW) в качестве кандидатов в ожидания ЗЛ. На этом этапе не стоит беспокоиться о поиске оптимальной формулировки требований, повторяемости требований и противоречиях в требованиях. Просто формируется полный иерархический список всех требований ANSW.

После этого аналитик последовательно проходит сформированный иерархический список требований, выявляя из него требования типа «запросы ЗЛ» (Stakeholder request — STKR). На данном этапе аналитик старается без потери смысла переформулировать каждое рассматриваемое требование-кандидат в ожидание — возможно, из одного такого требования сформируется несколько ожиданий, может быть, они будут объединены в какую-то иерархию. Здесь важно разрешать противоречия и конфликты выявленных ожиданий и исключить

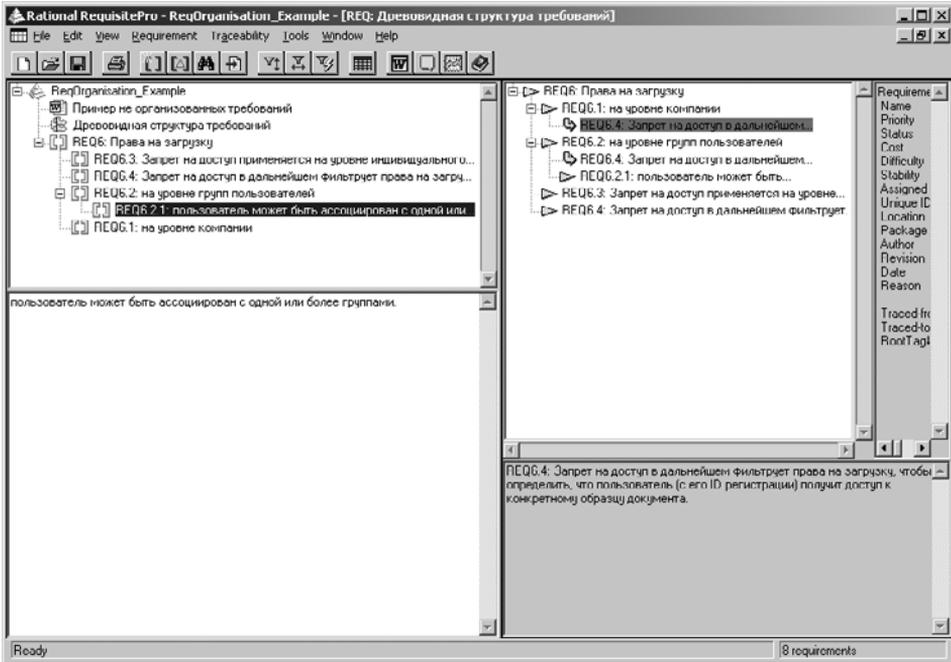


Рис. 18. Структура и иерархия требований

повторяемость требований. Каждое требование ANSW должно быть трассировано на требование STKR или иметь состояние «отвергнуто». После окончания анализа результатов интервьюирования все требования типа ANSW должны быть в состоянии «учтено в ожиданиях», «отложено» или «отвергнуто». Каждое требование типа STKR должно соответствовать четырем характеристикам: тестируемое, реализуемое, ясное для понимания, законченное и полное.

На языке UML можно отразить отношение между типами требований ANSW и STKR следующим образом (рис. 19).



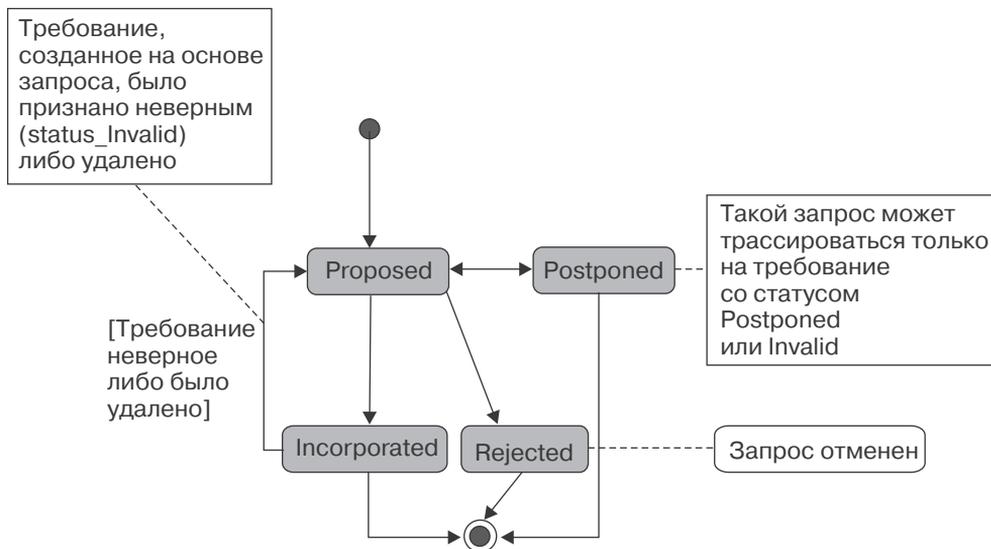
Рис. 19. UML-нотация связи требований

Требованиями STKR управляет только системный аналитик. Рекомендуемый перечень состояний для этого типа требований следующий.

Значение статуса	Краткое описание
Proposed	«В работе». Запрос заинтересованного лица (STKR) принят в разработку
Incorporated	«Учтено в ожиданиях». На основе запроса ЗЛ было создано одно или несколько требований
Postponed	«Отложено». Запрос ЗЛ отложен и при разработке требований учитываться не будет. Такой запрос может трассироваться только на требование со статусом Postponed или Invalid
Rejected	«Отвергнуто». Запрос ЗЛ отменен и при разработке требований учитываться не будет. Может не иметь трассировок и трассироваться на требование со статусом Invalid. В этот статус требование может быть переведено менеджером проекта только по решению ССВ <sup>1</sup> проекта
Invalid	Требование, созданное на основе запроса заинтересованного лица, было признано неверным

Следующая диаграмма представляет собой машину состояний и возможные переходы из состояния в состояние (рис. 20).

#### Статус требования типа «Запросы заинтересованных лиц»



**Рис. 20.** Иллюстрация статусов требования STKR

<sup>1</sup> ССВ – Change Control Board, комитет по управлению изменениями, [http://en.wikipedia.org/wiki/Change\\_control\\_board](http://en.wikipedia.org/wiki/Change_control_board).

## Проводить интервьюирование

Для выявления запросов заинтересованных лиц аналитик должен уметь проводить интервью. Рассмотрим этот вопрос более подробно.



---

Любое интервью начинается с определения цели интервью, доведения ее до интервьюируемого и назначения встречи. Часто все это приходится делать по почте.

---

Поэтому прежде, чем приступить к анализу практик интервью, приведу некоторые **хорошие практики деловой переписки**:

- ◆ если не вы начинаете переписку, а отвечаете на письмо, не отрезайте контекст писем. Сохраняйте нить обсуждения или обязательно пишите вводную часть письма, чтобы корреспондент не терял контекст переписки;
- ◆ выбирайте стиль обращения, адекватный роли системного аналитика, — уважительный, деловой и формальный;
- ◆ выдерживайте выбранный стиль письма от начала до конца в одном тоне;
- ◆ структурируйте письмо. В идеале оно должно в самом начале содержать конкретные вопросы, на которые вы хотите получить ответы (только самое главное). Затем должна быть вводная часть, напоминающая общий контекст данного конкретного письма. После этого должна идти основная часть, где приводятся ваши рассуждения и полный список вопросов для прояснения. В заключительной части рекомендуется выразить готовность к сотрудничеству и надежду на получение ответа. Не стесняйтесь писать «спасибо».

Теперь рассмотрим **хорошие практики интервью**.

### Подготовка к интервью

Выявите и составьте четкое понимание зоны ответственности каждого интервьюируемого (ЗЛ) и уровня его компетентности — с кем и о чем можно говорить, чье мнение в каких вопросах является компетентным.

Определите цель интервью. Подготовьте письменный перечень вопросов и вышлите их заранее — как приложение к приглашению на встречу через Outlook, например.

Убедитесь в том, что список вопросов совпадает с компетенцией интервьюируемого, что он сможет вам дать ответы и полномочен давать ответы по этим вопросам. Так, например, к начальнику ИТ лучше всего обращаться с техническими вопросами, а с вопросами «что это такое» — к тому, кто «этого» хочет, то есть к сотрудникам бизнес-дивизионов заказчика.

Прежде чем планировать встречу с заказчиком и спонсором, следует ознакомиться с предметной областью и по возможности выявить основные риски, которые могут повлиять на работы, сроки проекта и пр. Выявленные риски необходимо вынести на обсуждение.

На некоторые встречи с заказчиком и спонсором целесообразно приглашать системного архитектора, так как он может быть более компетентен в технических вопросах.

Если вы участвуете в проекте по развитию функциональности уже разработанной системы, нужно объяснять заказчику, спонсору, менеджеру проекта необходимость в проведении анализа влияния запросов заинтересованных лиц (импакт-анализа). Импакт-анализ — это работа аналитика, системного архитектора и разработчиков по выявлению необходимых изменений в требованиях, архитектуре и коде для реализации новых запросов ЗЛ. По результатам импакт-анализа должен измениться WBS (**Work Breakdown Structure**) — иерархическая структура работ проекта.

Цените время интервьюируемого. Планируйте встречу поминутно и следите за выполнением этого плана.

### Проведение интервью

В начале встречи всегда требуется «войти в контакт» — после приветствия можно сделать комплимент офису собеседника (если вы на его территории), сделать пару нейтрально-вежливых замечаний о погоде, об отличном внешнем виде собеседника, если вы видите в первый раз, можно сказать, что вы слышаны о высоком уровне квалификации собеседника и его профессионализме. Эта фаза интервью должна быть легкой, непринужденной и короткой — не более 1–3 минут. Но если собеседник захочет поговорить больше о погоде, спорте или собственном костюме, не перебивайте. Слушайте. Слушайте активно, поддерживая беседу, пока не почувствуете, что можно перейти к сути вашего визита.

В начале деловой части встречи надо пояснить цели встречи и повестку встречи. Представьтесь, объясните роль, которую вы выполняете в проекте, донесите цель встречи и расскажите, как будет проходить встреча. Обязательно уточните, может ли интервьюируемый помочь вам в достижении целей встречи или их нужно скорректировать. Спросите, нет ли у собеседника желания изменить повестку встречи и порядок обсуждения. В конце встречи обязательно планируйте дальнейшие шаги совместно с интервьюируемым.

Аналитик должен больше слушать, чем говорить. Задавайте вопросы и внимательно слушайте ответы. Ищите в ответах новые вопросы, используя навыки анализа и обработки информации. Старайтесь вовлечь интервьюируемого в беседу, задавать открытые вопросы, на которые нельзя ответить только «да» или «нет». Конспектируйте всю информацию, которую удастся «выудить».

Вопросы не должны строиться в «стиле благодетеля» («Какие у вас проблемы?»), более уместен «стиль слуги» («Как я еще могу быть вам полезен? Как разработываемое решение может помочь в вашей повседневной работе? Чем оно может быть вам полезным?»). Поощряйте интервьюируемого вопросами «Что еще вы можете об этом сказать? Что еще, на ваш взгляд, должна сделать система? Как еще можно облегчить выполнение этой операции?» и т. п. Обязательно уточняйте, кто еще

может рассказать что-то интересное по обсуждаемой теме с точки зрения собеседника. Помните, что вам нужно получить доступ к артефактам (рабочим инструкциям, распоряжениям и т. п.). Согласуйте эти вопросы с самого начала и узнайте, к кому обратиться за такими доступами.

## Завершение интервью

В конце встречи требуется «выйти из контакта» — поблагодарите за потраченное время, отметьте полезность и важность встречи, выразите надежду на дальнейшее сотрудничество и напомните, что просите ответить на ваше письмо с кратким отчетом (протоколом) по проведенной встрече, чтобы быть уверенными, что вы не исказили информацию. Этот протокол должен появиться и быть отправлен интервьюируемому не позднее 3–5 часов с момента встречи. Иначе вы рискуете не получить никакого фидбека (обратной связи). Поэтому при планировании проведения интервью в вашем личном графике планируйте время на обработку результатов интервью и создание протокола встречи.

## Проводить анкетирование

Вместо интервью допускается анкетирование ЗЛ, но если есть возможность очной встречи с ЗЛ, лучше всегда ею воспользоваться, а анкетирование рассматривать как дополнительный механизм, который позволяет прояснить то, что было неочевидно на интервью.

К анкетированию нужно готовиться не менее тщательно, чем к интервью. Анкеты для разных ЗЛ готовятся разные, здесь, как и при подготовке к интервью, следует хорошо продумать, на какие вопросы вам смогут компетентно ответить, и включать в анкету только их.

Старайтесь строить вопросы анкеты в порядке убывания важности и критичности выполнения ожиданий заказчика, при этом давайте ему возможность оценить важность и критичность самостоятельно.

Анкета должна позволять анкетированному на любой вопрос написать ответ в свободной форме. Не ограничивайте возможность ответа на ваши вопросы выбором из предложенных вами вариантов. Однако это не означает, что вы не должны такие варианты предлагать.

Очень плохая практика — формулировать вопрос таким образом, чтобы он содержал целый список вопросов. Разбивайте такие вопросы на несколько.



---

Многие топ-менеджеры заняты и не смогут найти время на заполнение анкет в электронном виде. Будьте готовы предоставить анкету в бумажном виде, чтобы анкетированный мог заполнить ее дома, по пути на встречу, в кафе. Обязательно переводите такие анкеты в электронный вид. Планируйте на это свое время.

---

Выделяйте время на обработку анкет в своем личном графике.

## Проводить совещания



В ходе работы аналитику придется проводить самые разнообразные совещания как с проектной командой, так и с представителями заказчика. Ниже я представлю свои рекомендации по подготовке и проведению любого совещания, которые вы можете взять за основу в своей работе.

### Подготовка совещания

1. Совещание должно иметь конкретные, достижимые и понятные цели.
2. Совещание должно иметь четкую повестку — как правило, это структурированный список вопросов, которые планируется обсуждать на совещании.
3. Определите длительность каждого пункта повестки.
4. Если это не первое совещание, в повестку совещания имеет смысл включить вводную часть, поясняющую контекст данного совещания.
5. Если инициатор совещания ожидает, что некоторые из участников будут выступать с докладами, то должно быть явно указано, кто и о чем будет докладывать.
6. Если для подготовки к совещанию нужно прочитать какие-то документы, например спецификации требований, то эти документы должны быть приложены к приглашению на совещание.
7. Рекомендуется рассылать приглашение на совещание за 1–3 дня до его проведения.
8. При определении даты и времени совещания нужно учитывать рабочее расписание и график предполагаемых участников.
9. Совещание должно иметь регламент — ограничение по времени.
10. Длительность совещания не должна превышать 1,5 часа. Более продолжительные совещания неэффективны.
11. Следует приглашать минимально необходимое количество участников. Идеальный вариант —  $7 \pm 2$ . Совещание, на котором более девяти участников, с очень большой долей вероятности будет непродуктивным.
12. Определите перечень участников совещания, чье присутствие на совещании обязательно.
13. Определите перечень участников совещания, которые могут прийти на совещание по желанию.
14. Каждый обязательный участник должен явно подтвердить или отклонить свое участие в совещании. Если обязательный участник отклоняет свое участие, выясните причину.
15. У каждого совещания должны быть ведущий, модератор и секретарь. Ведущий ведет совещание в соответствии с повесткой, модератор имеет право прерывать прения, вмешиваться в ход проведения совещания. Секретарь

фиксирует максимально полную картину совещания: вопросы, ответы, реплики, не заявленные в повестке выступления с мест, принятые решения и дальнейшие шаги. Эти роли совещания должны быть явно указаны в повестке.

16. Старайтесь заранее просчитывать дальнейшие шаги, которые вы предпримете после этого собрания.
17. По итогам совещания должен быть составлен его протокол. Желательное время подготовки и отправки протокола совещания — 3–5 часов.
18. При изменении времени, длительности или повестки совещания инициатор должен разослать уведомления всем участникам.
19. Инициатор совещания (как правило, это ведущий совещания) обязан заранее заказать переговорную комнату и все необходимое оборудование для проведения совещания. За 30 минут до начала совещания следует убедиться, что для проведения совещания все готово, и устранить мелкие недостатки (отсутствие бумаги, карандашей, маркеров для флипчарта).



---

Пример. Шаблон протокола совещания

**Тема совещания:** [тема собрания].

**Дата совещания:** [дд.мм.гггг].

**Начало:** [чч:мм].

**Окончание:** [чч:мм].

**Общее время совещания:** [x] мин.

Регламент не превышен/превышен на [y] мин.

**Докладчик:**

[ФИО докладчика/-ков через запятую].

**Ведущий:**

[ФИО ведущего].

**Секретарь:**

[ФИО секретаря].

**Присутствовали:**

[ФИО присутствовавших через запятую].

**Отсутствовали:**

[ФИО отсутствовавших (уважительная причина отсутствия) через запятую].

**Обсуждавшиеся вопросы:**

1. [Реально обсуждавшийся на собрании вопрос 1].
2. [Реально обсуждавшийся на собрании вопрос 2].

**Принятые решения:**

1. [Название группы решений 1].
  - 1.1. [Решение 1.1].
  - 1.2. [Решение 1.2].

2. [Название группы решений 2].
  - 2.1. [Решение 2.1].
  - 2.2. [Решение 2.2].

**Дальнейшие шаги:**

1. [Описание дальнейшего шага 1], ответственный — **[ФИО]**, срок — **[дата]**.
2. [Описание дальнейшего шага 2], ответственный — **[ФИО]**, срок — **[дата]**.

*Отсутствующие из числа участников, приглашенных «по желанию», не фиксируются. В разделе «Отсутствовали» фиксируются только приглашенные как обязательные участники собрания. ФИО ответственных за дальнейшие шаги и срок предполагаемого выполнения работ выделяются полужирным шрифтом.*

Шаблон протокола совещания вы можете скачать на сайте книги.

## **Проведение совещания**

1. Если совещание проходит в частично незнакомой друг другу целевой аудитории, ведущий представляется и просит представиться участников собрания либо представляет их сам.
2. Ведущий открывает совещание, оглашает цели и повестку совещания, представляет модератора совещания, секретаря и докладчиков. Здесь, как и при проведении интервью, рекомендуется спросить, всем ли понятны цели и повестка, нет ли дополнений и/или изменений повестки, обсудить предложения по изменению повестки (если есть) и утвердить окончательную повестку совещания. При этом может потребоваться вмешательство модератора совещания.
3. В первую очередь я рекомендую определить наличие кворума из состава обязательных участников и в случае его отсутствия, возможно, отменить проведение совещания.
4. Каждый участник должен понимать, зачем он приглашен на совещание, и быть подготовлен к нему. Я рекомендую ведущему явно спрашивать у аудитории, кому удалось ознакомиться с материалами к совещанию, всем ли понятно, зачем их пригласили на совещание и что от них ожидается.
5. Вводная часть совещания на этом завершается, и начинается совещание. Совещание проводится ведущим в соответствии с повесткой. Ведущий и модератор строго следят за соблюдением регламента выступлений докладчиков.
6. Докладчик (или ведущий) перед началом доклада должен определить форму проведения дискуссий — можно ли задавать уточняющие вопросы сразу, или будет отдельное время для того, чтобы высказать свои мнения. Я также рекомендую докладчику обратить внимание секретаря, что именно из вопросов и обсуждений для него важнее всего, чтобы секретарь совещания обратил внимание на эти моменты при конспектировании совещания.

7. Докладчик делает доклад. Модератор следит за тем, чтобы выступал всегда один участник совещания. Выступающего участника может прерывать только ведущий или модератор. Модератор также следит за тем, чтобы спор не переходил на личностный уровень, жестко и сразу пресекает подобные попытки.
8. В соответствии с определенными докладчиком правилами начинаются обсуждения. Ведущий совместно с модератором в мягкой мотивирующей форме старается обеспечить конструктивный климат совещания, чтобы, критикуя чужие предложения, каждый участник совещания предлагал свой вариант. Ведущий или модератор могут остановить бесперспективные и затянувшиеся по времени обсуждения.
9. Ведущий должен стараться, чтобы обсуждения заканчивались выявлением дальнейших шагов и действий. Для всех намеченных действий должны быть назначены ответственные и сроки. Допускается не назначать ответственных и сроки на самом совещании, но это должно быть взвешенное и озвученное решение.

## Проводить «мозговые штурмы»

«Мозговой штурм» — это совещание особого вида, целью которого является генерация максимального количества идей относительно обсуждаемой проблемы.

Такие совещания рекомендуется проводить в комнате, оборудованной флипчартом с цветными фломастерами, телефоном для телеконференций (если надо) и ПК, подключенным к локальной сети, со всем возможным ПО для рассмотрения проектных материалов, то есть со всем необходимым для достижения целей «мозгового штурма».



---

«Мозговые штурмы» должны проходить с соблюдением следующих принципов: «Нет» критике! «Нет» оценке! «Нет» обсуждениям идей!

---

«Мозговой штурм» — это «идеегенерирующее» событие. Все идеи, несмотря на их кажущуюся абсурдность, должны фиксироваться секретарем. Идеи должны обсуждаться на других совещаниях в формате обычного проблемного совещания.

За более подробной информацией о методах и практиках проведения «мозговых штурмов» рекомендую обратиться к книге Леффингвелла *Managing Software Requirements* [21].

### 3.2.4. Эффективно взаимодействовать с командой и ЗЛ по почте

Аналитик в определенном смысле является центральной (но ни в коем случае не главной) фигурой на коммуникационном поле проекта. Он активно контактирует, с одной стороны, с заинтересованными лицами проекта, включая пред-

ставителей заказчика, с другой — с проектной командой: менеджером проекта, архитектором, разработчиками, тест-дизайнером, тестировщиками.

Конечно, самым эффективным способом коммуникации является личное общение, но в реальной жизни, как правило, невозможно встречаться с проектной командой и заинтересованными лицами в любое время, когда аналитику нужно прояснить требования. Главным образом потому, что в момент подготовки требований другие проектные роли обычно заняты на разработке предыдущей версии системы/разработке другой системы, а представители заказчика, хотя и заинтересованы в успехе проекта, должны выполнять свои непосредственные обязанности и у них просто может не оказаться времени на встречу.

Именно поэтому для аналитика важны навыки ведения деловой переписки.

### Правила формирования заголовка писем

Каждое, без исключений, письмо должно иметь тему. Письма без темы являются одним из наиболее распространенных и в то же время самых грубых нарушений правил использования почты.

Тема должна позволить получателю понять контекст письма без его открытия.

### Правила ведения переписки

Переписка ведется уважительным тоном, в деловой и дружелюбной форме.

Основной принцип ведения переписки — информативность и легкость восприятия.

Ответственность за восприятие информации получателем лежит на авторе письма. В связи с этим для уверенности в правильном и одинаковом понимании предмета обсуждения следует формулировать и задавать вопросы обратной связи (так называемые открытые вопросы: «Правильно ли я понял, что...»).



---

Если в переписке участвует более двух человек, то ответ на письмо лучше выполнять через команду почтового клиента «Ответить всем». Исключение составляют опросы, когда ответ пишется только инициатору опроса.

---

При ответе на письмо заголовок письма никогда не изменяется (не учитывая автоматически добавленные почтовым клиентом Re:, Fw: и т. п.).

При обращении к конкретному сотруднику в теле письма следует явно выделять его имя. Например: «**Иван**, по твоему (или Вашему — как принято в компании) вопросу отвечаю...»

При просьбе выполнить ту или иную работу (дать информацию) обязательно явно указывать дату и время, к которым эта информация необходима. Например: «**Иван**, прошу тебя (или Вас) прислать мне документ... с комментариями сегодня — **01.01.11 к 16:00**».

Если сотрудник, к которому обращаются с просьбой, не может выполнить просьбу в указанное время, он сразу сообщает об этом автору с указанием срока, когда сможет выполнить просьбу.

При большой текущей загрузке и невозможности ответить на письмо в сроки, указанные в письме, адресат обязательно пишет, что он ознакомился с письмом, и указывает дату и время, когда сможет детально ответить на данное письмо.

При длительном обсуждении какой-либо темы по почте в режиме дискуссии в начало письма следует обязательно включать раздел «Резюме/ краткие выводы», в котором кратко приводить основные предложения/ замечания/ уточнения.

При инициации обсуждения проблемы в форме переписки рекомендуется формировать структуру первого письма в соответствии с нижеследующей структурой.

- ◆ Общее описание проблемы.
- ◆ Существующее решение (тезисно).
- ◆ Предлагаемое решение (резюме).
- ◆ Существующее решение (детально, по необходимости).
- ◆ Предлагаемое решение (детально).

Это не означает, что в письме следует явно выделять название этих разделов, однако информация должна структурироваться в соответствии с приведенным планом. Для структурирования информации рекомендую использовать американский стиль письма, когда новая мысль начинается с нового абзаца. По такому принципу структурирована и наша книга.



---

Во избежание спама в переписке рекомендуется обмениваться ответами не более чем 3–5 раз. Если после третьего ответа инициатору переписки неясны какие-либо принципиальные моменты, целесообразно провести совещание, чтобы обговорить все вопросы, поднятые в переписке. Совещание — самая эффективная, но и самая дорогостоящая форма коммуникации. Если после третьего ответа инициатору переписки необходимо уточнить незначительные моменты, допускается проведение еще двух раундов переписки.

---

Несмотря на навыки деловой переписки, не стоит забывать, что приоритетным способом коммуникации по горячим вопросам являются телефонные переговоры или встреча. В случае решения срочных вопросов не надо надеяться на почту, лучше связаться с нужными людьми по телефону или провести личную встречу.



---

Я рекомендую всегда закреплять любую устную договоренность с заинтересованными лицами и с проектной командой либо формальным письмом, либо протоколом собрания, либо согласованием и доведением измененной информации в артефакте (документе).

---

### 3.2.5. Определять границы системы и создавать концепции

Для определения границ системы обычно используется самая простая диаграмма — контекстная.

Контекстная диаграмма — модель анализа, отображающая место новой системы в соответствующей среде. Она определяет границы и интерфейсы между разрабатываемой системой и сущностями, внешними для этой системы, например пользователями, устройствами и прочими информационными системами, — акторами системы.

Невозможно корректно установить границы системы без определения бизнес-требований и бизнес-правил.

Существует достаточно много определений этих типов требований. Я предпочитаю следующее: требования бизнеса к системе (Business Requirements — BREQ) — это то, что нужно бизнесу от системы, иначе говоря, какие выгоды получит бизнес заказчика от системы, а бизнес-правила (Business Rules — BRULE) — «правила игры» этого бизнеса, важные с точки зрения системы.

Бизнес (упрощенно) — это «прибыль, капитализация и репутация», то есть бизнес-требование вполне может звучать как «минимизировать расчет и составление спецификаций для производства такой-то детали». Но здесь важно заметить, что бизнес-требованием его делает не ожидание мастера цеха производства «таких-то деталей» по улучшению/облегчению собственной деятельности, а то, что в результате выполнения этого требования бизнес в конечном итоге увеличит свою прибыль. С другой стороны, бизнес-потребность может звучать и более обобщенно, например: «обеспечить эффективное распределение готовой продукции по складам». Но пока это утверждение не является требованием, поскольку не соответствует характеристикам требования, — его нужно анализировать и уточнять. Что такое эффективное распределение? Что такое готовая продукция? И т. д. Как это делать, мы с вами уже обсуждали.

Что же такое бизнес-правила? Это «правила игры», принятые в бизнесе, важные с точки зрения системы. Например, требование «при получении денежного перевода сумма зачисляется на счет клиента в течение 15 минут и доступна к дальнейшим операциям не позднее 1 часа с момента получения перевода» критически важно, если мы разрабатываем систему денежных переводов для банка. Второй важной категорией бизнес-правил являются законы, государственные регламенты и распоряжения по отраслям, например: «информация о входящих и исходящих звонках абонента должна храниться в течение N месяцев».



Бизнес-требования и бизнес-правила должны обязательно выполняться системой. Первые — потому что это и есть выгода бизнеса от системы и, если эти требования не будут выполняться, такая система просто никому не будет нужна. Вторые — потому что в противном случае система может нарушить производственный процесс или будет противоречить требованиям закона.

На понимание и определение границ системы влияют также собранные ожидания заинтересованных лиц и результаты их обсуждений с командой, в процессе которых у участников команды формируется первое представление о системе. Выявив бизнес-требования и бизнес-правила, аналитик должен разрешить противоречия между ожиданиями заинтересованных лиц и потребностями и правилами бизнеса. Ожидания ЗЛ имеют более низкий приоритет, чем бизнес-требования и бизнес-правила, и в случае выявления противоречий ожидания ЗЛ отклоняются с объяснением причин отклонения инициатору этого ожидания.

Для того чтобы управлять и обосновывать отклонения запросов ЗЛ, строятся следующие трассировки (рис. 21).



**Рис. 21.** Трассировки для обоснования отклонения запросов ЗЛ

Но и это еще не все. Если вы разрабатываете продукт, который в перспективе будет сертифицироваться, или если надо учитывать конкретные стандарты и ГОСТы на программный продукт либо его компоненты, вы должны учесть и другие типы требований.

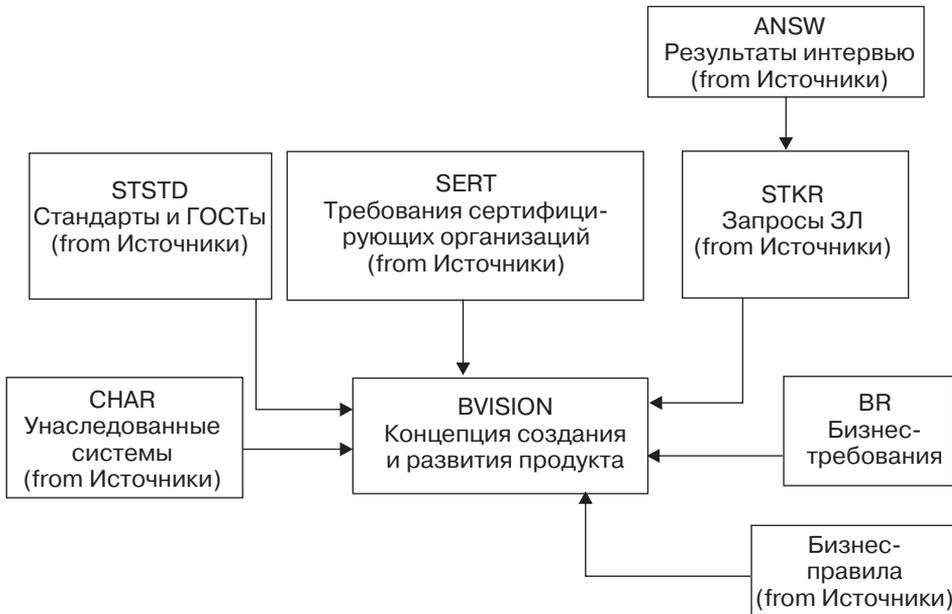


Для упрощения работы с требованиями к системе я рекомендую вводить требование типа «концепция создания и развития продукта» (**Business Vision — BVISION**), в котором **аккумулировать информацию из всех упомянутых типов требований** (рис. 22).

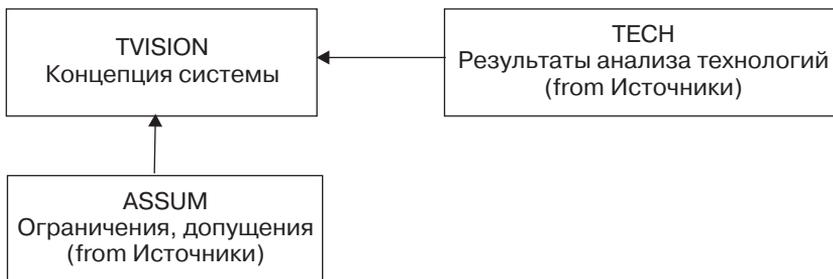
Создание такого типа требований возможно, когда нет задачи проследить судьбу каждого исходного типа требования. В отдельных компаниях существует роль «менеджер продукта», в обязанности которого включена работа по созданию документа «Концепция создания и развития продукта». Строго говоря, это идеальный вариант. В данном случае аналитик должен взять такой документ за основу и выявить из него и исходных требований требования типа BVISION, чтобы иметь возможность управлять ими и строить трассировки на требования более низкого уровня.



Второй аспект, необходимый для определения границ и концепции системы, — учесть передовые технологии, выявленные ограничения и допущения технического характера. Эта информация аккумулируется в требованиях типа «концепция системы» (**Technical Vision — TVISION**) (рис. 23).



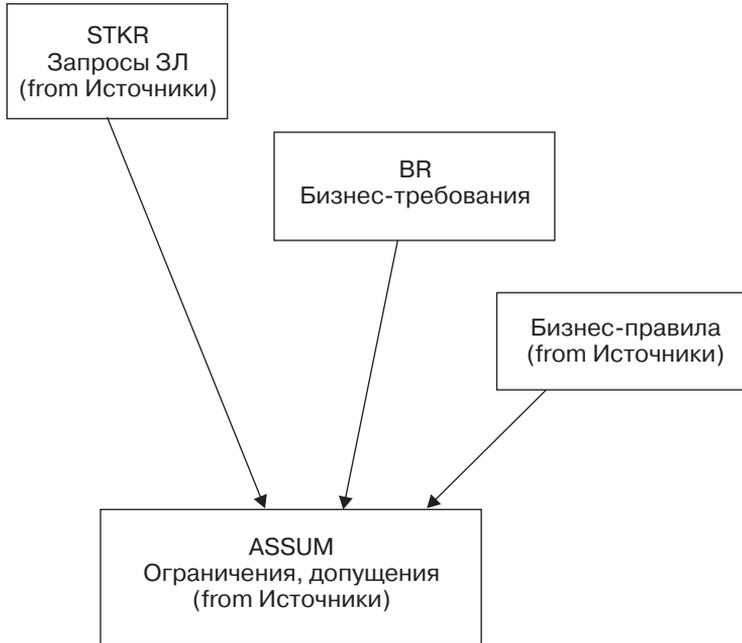
**Рис. 22.** Концепция объединяющего требования Business Vision



**Рис. 23.** Концепция объединяющего требования Technical Vision

Такой анализ обычно выполняет системный архитектор, от аналитика же, возможно, понадобится помощь в создании требований в специализированном инструменте управления требованиями. Архитектор активно пользуется результатами работы аналитика на этапе выявления ожиданий заинтересованных лиц, бизнес-требований и бизнес-правил (рис. 24).

Зачастую архитектор не управляет требованиями типа ASSUM (Assumption) или TECH (Technology), а сразу создает спецификацию «Концепция системы». В данном случае аналитик не должен полагаться на то, что архитектор ничего не упустил, а обязан провести ревью этого документа, прослеживая, как учтены в концепции запросы ЗЛ, бизнес-требования и бизнес-правила. Если какие-то требования не учтены, аналитик совместно с архитектором должен поправить концепцию системы.



**Рис. 24.** Происхождение требований типа «ограничения и допущения»

На сайте книги вы можете скачать шаблон спецификации «Концепция системы».

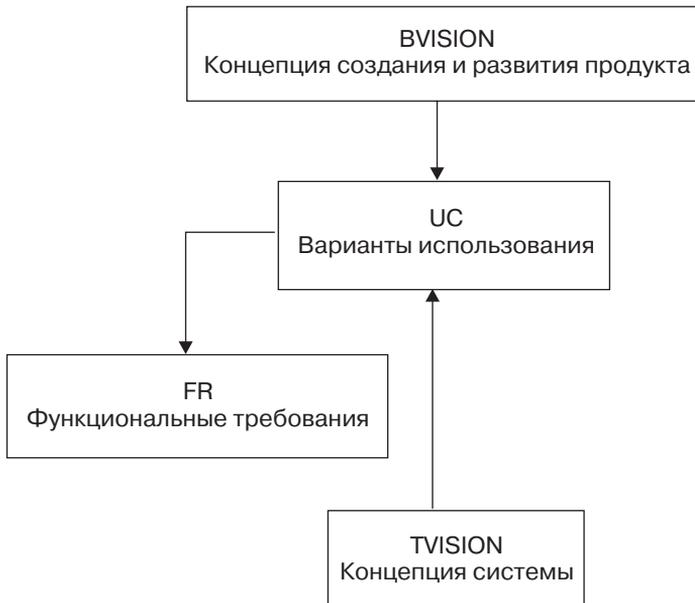
### 3.2.6. Выделять подсистемы и определять их функции

После описания системы в целом проводится разбиение ее на крупные фрагменты. Для этого выявляются функции системы. Здесь возможны два варианта развития событий.

Первый — аналитик, определив совместно с архитектором границы и концепцию системы, начинает моделировать поведение системы. Как вы уже знаете, в разных методологиях процесс моделирования поведения системы реализуется по-разному. Рассмотрим случай моделирования поведения в виде вариантов использования системы (*Use Case*) для каждого выявленного актора. По определению «*Use Case* — функция системы, которая приносит ощутимый и значимый результат актору» и описывается как сценарий взаимодействия актора и системы. После того как аналитик выявил все возможные варианты использования системы (*Use Cases*) со стороны каждого актора, выявленные варианты группируются по схожим признакам. Варианты использования системы — это функции системы. Таким образом, выполняя группировку, мы, по сути, осуществляем функциональную декомпозицию, объединяя «родственные» функции системы в подсистемы. В результате появляются:

«Подсистема распознавания речи», «Подсистема логирования», «Подсистема взаимодействия с внешними устройствами» и т. д. На протяжении всей работы с моделью вариантов использования аналитик тесно взаимодействует с архитектором и разработчиками. После утверждения модели вариантов использования все они описываются на уровне, достаточном для разработки архитектуры и разработки кода (рис. 25).

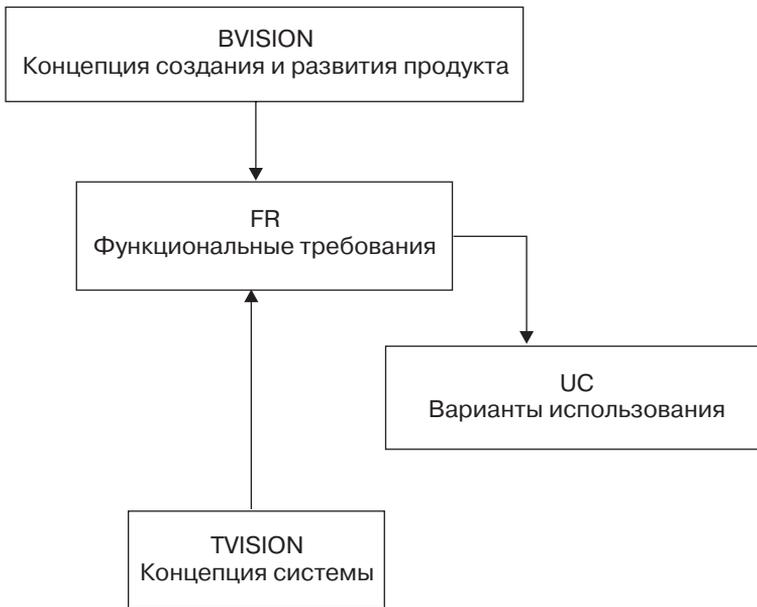
Преимущество данного подхода в том, что проектируемая таким образом система с самого начала ориентирована только на взаимодействие системы с акторами, и дальнейшая разработка может строиться через приоритизацию вариантов использования. Еще одно его преимущество: не обязательно тратить время на выявление функциональных требований — это имеет смысл, если необходимо создать и утвердить у заказчика формальную спецификацию, например, в виде ТЗ. Для разработки функциональные требования неактуальны, поскольку она будет выполняться на основании сценариев вариантов использования.



**Рис. 25.** Описание поведения системы (варианты использования) как основа описания требуемой функциональности ИС

Другой вариант развития событий — когда на основании требований BVISION и TVISION проектная команда определяет функциональные требования к системе в виде утверждений «система должна делать то-то и то-то» (рис. 26). Эти утверждения организуются в иерархии с главными требованиями в виде родительских узлов, например: «Система должна позволять речевой ввод информации», «Система должна логировать все операции», «Система должна обеспечивать работу с принтерами», «Система должна обеспечивать работу

со сканерами». Затем требования группируются по схожим признакам и выявляются те же подсистемы: распознавания речи, логирования и взаимодействия с внешними устройствами. Далее для выделенных подсистем создаются варианты использования, которые реализуют выявленные требования. Преимущество данного метода в том, что благодаря выявлению функциональных требований всей командой очертания и назначение системы становятся более четкими без трудозатрат на построение модели вариантов использования, анализ функций системы и ее взаимодействия с акторами. Иногда модель вариантов использования не строится, а реализация поведения системы оставляется на усмотрение разработчика. На мой взгляд, такой вариант допустим только для простых систем с очевидной функциональностью или для «домашних» систем, если требования к качеству этих систем невысокие.



**Рис. 26.** Функциональные требования как основа описания требуемой функциональности ИС и поведения системы (вариантов использования)

На мой взгляд, истина где-то посередине. Выбор того или иного пути зависит от сложности системы, от степени новизны предметной области для команды и опыта разработки систем такого класса, от квалификации проектной команды и владения современными средствами моделирования.



Я рекомендую гибридный вариант, когда определяются только высокоуровневые функциональные требования к системе до, скажем, третьего уровня максимум, а детальная проработка функциональности достигается за счет Use Case-моделирования. Такой подход позволяет сочетать в себе плюсы обоих вариантов.

Процесс выявления подсистем, определения их функций и границ называется функциональной декомпозицией.

Для автоматизации работы с моделями и требованиями я использую инструменты **Rational Rose** и **Rational Requisite PRO**. С их помощью можно легко построить единую среду разработки (рис. 27).



**Рис. 27.** Единая среда разработки

Эта среда разработки за счет интеграции средств моделирования и управления требованиями обеспечивает:

- ◆ возможность быстрого перехода от требования в проекте требований к соответствующему варианту использования в модели вариантов использования;
- ◆ возможность быстрого перехода от варианта использования в модели вариантов использования к соответствующему требованию в проекте требований.

Как я уже писал, это не единственные инструменты, которые можно использовать в своей работе. Рекомендую обратить внимание на DOORS Telelogic, Sparx Enterprise Architect.

### 3.2.7. Выявлять пользовательские требования

Параллельно с проведением моделирования Use Case аналитик выявляет требования будущих пользователей к системе (UREQ).

Отличие пользовательских требований от ожиданий ЗЛ в том, что ожидания ЗЛ — это ожидания от использования системы **с точки зрения выполнения бизнес-операций ЗЛ**, иными словами, это бизнес-заинтересованность ЗЛ в результатах деятельности системы. Пользовательские требования (UREQ) — это ожидания потенциальных пользователей системы **от работы с самой системой**.

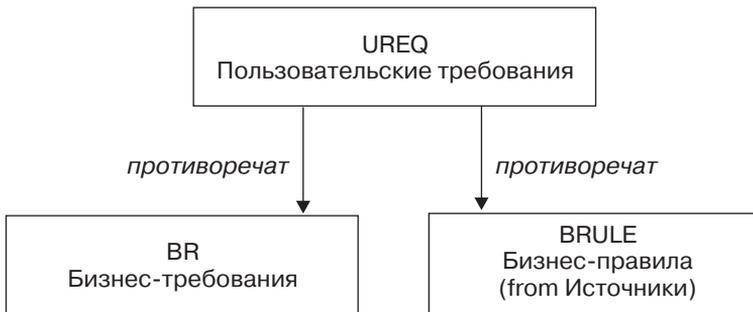
Следует понимать, что не все ЗЛ являются пользователями и не все пользователи рассматриваются как ЗЛ. Например, если мы проектируем систему учета рабочего времени, то ЗЛ с большой вероятностью станут директор компании, менеджеры проектов, начальники отделов, отдел кадров, бухгалтерия, а пользователями системы — все обычные сотрудники, сотрудники ИТ-отдела (в качестве администраторов), менеджеры проектов, начальники отделов и, возможно, директор компании.

Пользовательские требования выявляются путем проведения интервью и анкетирования будущих пользователей системы и представляют ожидания пользователей по работе с системой. Выполнение этих требований делает систему удобной для работы конечных пользователей, что является одним из слагаемых компонентов успеха внедрения и использования системы.



По мере выявления пользовательских требований аналитик должен постоянно проводить проверки противоречивости пользовательских требований бизнес-требованиям и бизнес-правилам. Очевидно, что бизнес-требования и правила более приоритетны, и если есть такое противоречие, пользовательское требование отклоняется, а пользователь, выдвинувший такое требование, уведомляется о причинах отклонения его ожиданий.

Для эффективного управления этим процессом строят следующие трассировки (рис. 28).



**Рис. 28.** Управление противоречиями пользовательских и бизнес-требований и правил

Утвержденные пользовательские требования учитываются аналитиком в вариантах использования системы.

Наиболее распространенная ошибка — когда пользовательские требования путают с требованиями к пользовательскому интерфейсу. Не повторяйте эту ошибку. Требования к пользовательскому интерфейсу касаются внешнего вида приложения, концепции и дизайна графического интерфейса системы. Пользовательские требования — это требования по удобству работы с системой. Например: «должна быть возможность выбора нескольких файлов для отправки одному и тому же адресату». Здесь не идет речь о том, что конкретно должно быть реализовано в пользовательском интерфейсе. Анализируя это требование, аналитик с большой долей вероятности получит следующие требования к пользовательскому интерфейсу: «при создании сообщения адресату должна быть обеспечена возможность добавления нескольких файлов путем выполнения операции *drag & drop* из Explorer»; «должна быть возможность выбрать несколько файлов в Explorer и выбрать из выпадающего меню команду “Отправить адресату...”». Эти требования может сформулировать тот же самый пользователь, но если в первом случае речь шла о принципиальной возможности системы, то во втором — уже о конкретных предпочтениях по работе с графическим интерфейсом.

Пользовательские требования очень похожи на функциональные требования. Если перед вами не стоит задача отчитаться перед каждым будущим пользователем о судьбе высказанных им ожиданий, можете не выделять в проекте требований отдельный тип UREQ, а фиксировать их сразу как функциональные требования или требования других типов. Например, часто пользовательские требования касаются формата данных: «зарплата должна округляться до четвертого знака после запятой».

### 3.2.8. Выявлять нефункциональные требования



Нефункциональное требование — это:

- ♦ описание ограничений на функционирование системы, отвечающее на вопрос «В каких внешних условиях должна работать система?»;
- ♦ описание ограничений на функционирование системы, отвечающее на вопрос «В каких внутренних условиях должна работать система?»;
- ♦ описание ожидаемых качественных параметров работы системы, отвечающее на вопрос «Какие измеримые количественные характеристики должна иметь система?»;
- ♦ описание ограничений на реализацию функций системы, отвечающее на вопрос «Как (с использованием чего) система должна что-то делать?»;
- ♦ требования к данным, с которыми должна работать система;

- ♦ требования к протоколам, с использованием которых должна работать система;
- ♦ требования к интерфейсам, по которым система должна взаимодействовать с другими системами.

Нефункциональные требования иногда называют дополнительными требованиями (supplementary requirements).

На рис. 29 представлена UML-диаграмма, поясняющая состав нефункциональных требований.

Нефункциональные требования выявляются всей проектной командой. Хотя требования GUI, ICE, DOC, DATA, CERT и относятся к нефункциональным, их лучше выделять в проекте требований как отдельные типы, поскольку с ними работают разные проектные роли. Например, с требованиями типа GUI работают в основном системный аналитик и дизайнер графических интерфейсов, с требованиями ICE и DATA — системный аналитик, архитектор и проектировщик базы данных, с требованиями DOC и CERT — специалисты отдела документирования и сертификации (технические писатели и менеджеры по сертификации).

Источник нефункциональных требований — BVISION и TVISION. Архитектор должен анализировать эти требования, задавая себе вопросы, приведенные в определении нефункционального требования, и стараться выявить нефункциональные.

Первое, с чего следует начинать работу с нефункциональными требованиями, — посмотреть на систему с точки зрения установки и развертывания. На каких версиях ОС может работать система, какие дополнительные компоненты необходимы для ее работы, какие версии этих компонентов гарантированно поддерживаются?

Второе — описать ожидаемые качественные параметры работы системы, отвечающие на вопрос «Какие измеримые количественные характеристики должна иметь система?». Особенно это касается функций системы, использующих каналы связи. Какая пропускная способность этих каналов необходима? С какой скоростью должна передаваться информация? Есть ли требования к защищенности информации? Какой протокол передачи данных используется? Какой формат данных? На втором месте стоит функциональность многопользовательских систем. Каково максимально допустимое время отклика у системы? Какое время отклика оптимальное? Сколько пользователей могут работать с системой одновременно? Какова скорость выполнения расчетов?



Хорошая практика выявления нефункциональных требований — критический анализ описаний вариантов использования, который также выполняется аналитиком и архитектором совместно. В описании находятся речевые конструкции со смыслом «система взаимодействует с...», «передает информацию», «сообщает», «пользователь вводит данные», «система рассчитывает/вычисляет», «система шифрует информацию», которые анализируются с помощью вопросов: есть ли какие-то ограничения на реализацию этой

## Нефункциональные требования

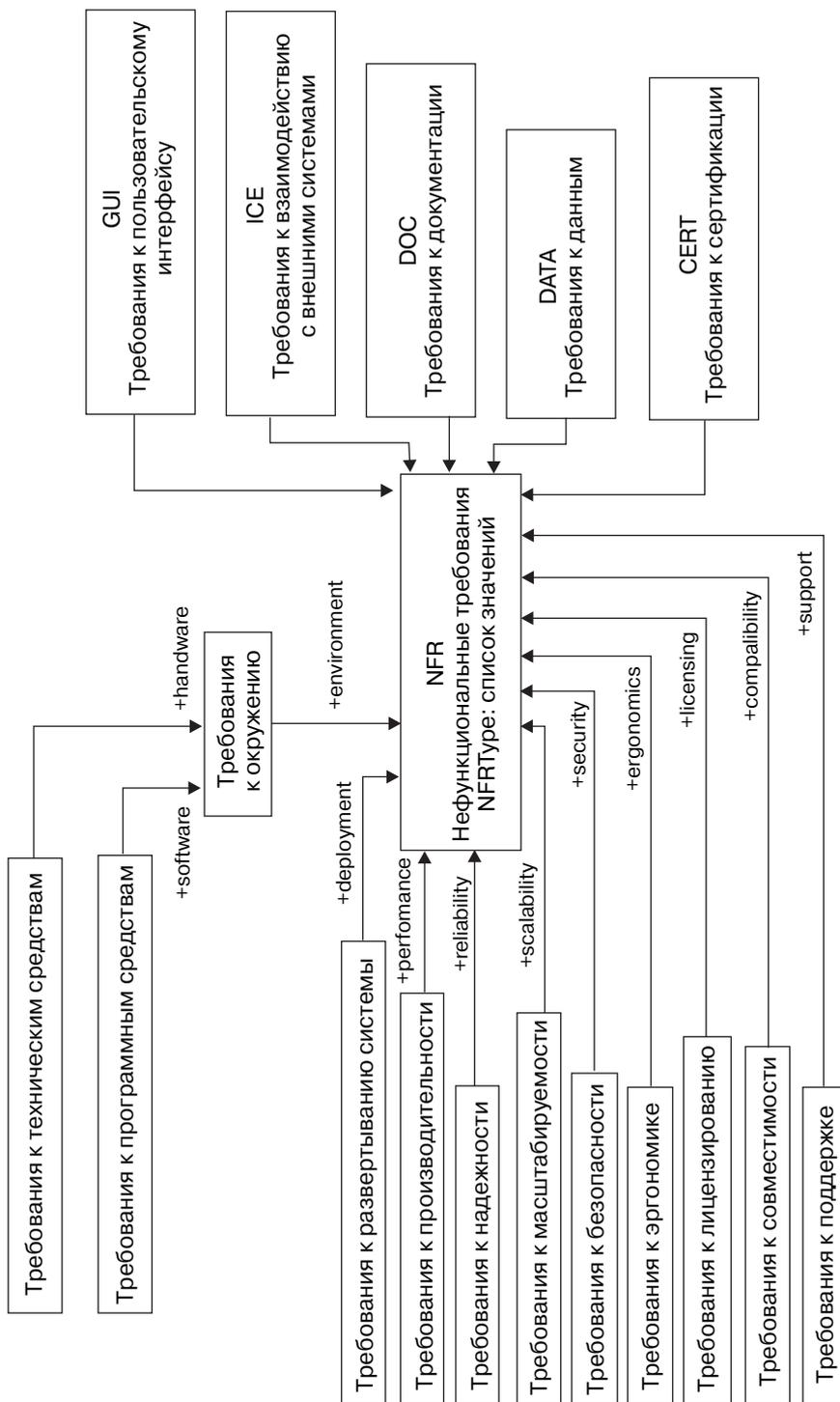


Рис. 29. Состав нефункциональных требований

функциональности? Определены ли все внешние и внутренние условия для работы этой функциональности? Как (с использованием чего) система должна что-то делать? Кто, что, кому, когда, в какой форме, с какой периодичностью, по какому каналу предоставляет?

Нефункциональные требования крайне критичны при реализации системы. Их выполнение — залог достижения желаемой производительности системы и ее удовлетворительных скоростных характеристик.

Некоторым разработчикам бывает трудно понять разницу между функциональными и нефункциональными требованиями, поскольку понимание функциональности системы у разработчика системы и аналитика разное. Для разработчика функциональность — это все функции/методы всех классов системы, даже если эти функции не приносят никакого полезного результата какому-то актору. Аналитик же должен думать о функциях системы как о вариантах использования.

Действительно, с точки зрения разработчика, требования «система должна использовать SSL для обеспечения безопасности» и «система должна поддерживать одновременную работу с десятью пользователями» скорее функциональные, чем нефункциональные, ибо для того, чтобы реализовать требование об одновременной работе десяти пользователей, например, он должен будет придумать и воплотить определенную функциональность классов системы.

Я рекомендую более внимательно относиться к формулированию требований. Вышеприведенные требования лучше разделить на два типа — функциональные и нефункциональные. Например, требование «система должна использовать SSL для обеспечения безопасности» разбивается на функциональное требование «система должна обеспечивать безопасную передачу данных по сети» и нефункциональное — «безопасность передачи данных должна обеспечиваться с помощью SSL». Таким образом, нефункциональные требования «дополняют» функциональные, именно поэтому они и называются часто *supplementary* (дополняющие) requirements. Это помогает найти общий язык с разработчиками.

### 3.2.9. Выявлять требования к пользовательскому интерфейсу

Требования к пользовательскому интерфейсу касаются внешнего вида приложения, концепции и дизайна графического интерфейса системы. Основным источником выявления таких требований — интервьюирование будущих пользователей системы. Важно понимать, что не все заинтересованные лица будут выступать как будущие пользователи, и следует ограничивать их в стремлении повлиять на дизайн пользовательского интерфейса.



Я рекомендую фиксировать в виде требований к пользовательскому интерфейсу общие, концептуальные требования, например такие, как: «любая выполняемая операция должна иметь возможность отмены» или «все экранные формы системы должны позволять перейти на главную страницу портала по

одному щелчку». Остальные требования к пользовательскому интерфейсу рекомендуем сразу отображать в форме прототипов интерфейсных форм.

Подобные прототипы могут разрабатываться в Microsoft Visio, удобную функциональность имеет Sparx Enterprise Architect, также можно создавать фреймворк (экранные формы без какой-то реальной функциональности) на любом языке программирования, например на C#.

Следует четко понимать, что аналитик не способен и не должен разрабатывать конечный внешний вид интерфейсов, его задача — создать и согласовать с пользователями системы прототип GUI. Окончательный дизайн экранных форм выполняет дизайнер графических интерфейсов.

На сайте книги вы можете скачать специальный скрипт для Microsoft Visio, который позволяет создавать прототипы GUI и специфицировать GUI элементы этих прототипов непосредственно в Visio, сохраняя комментарий/требование в базе данных Microsoft Access. Этот инструмент будет особенно полезен, если выбранная методология разработки требований — Iconix [21].

### 3.2.10. Управлять состояниями требований

Выявление требований в проекте требований имеет смысл тогда и только тогда, когда будут выполняться трассировки, задаваться атрибуты требований и когда этими требованиями планируется управлять (рис. 30). Если этого делать не планируется, достаточно обычного текстового документа. Об управлении требованиями мы подробнее поговорим в отдельной главе.

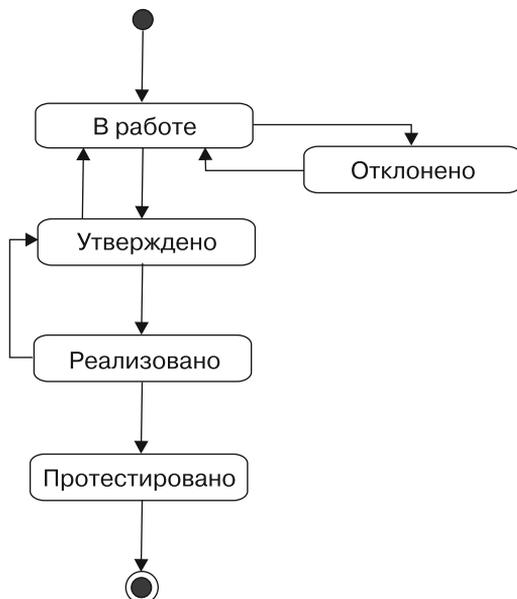


Рис. 30. Простейшая диаграмма состояний для управления требованиями

Здесь переход из состояния «Реализовано» к «Утверждено» возможен вследствие неуспешного тестирования.

### 3.2.11. Формировать спецификации требований

Спецификации требований упрощают процесс согласования требований с заинтересованными лицами и командой проекта. С другой стороны, если вы используете современные средства моделирования и управления требованиями и способны убедить заинтересованных лиц работать с этими артефактами, то спецификации требований не нужны. Однако такое случается достаточно редко.



Единственное, что здесь можно посоветовать, — постараться свести к минимуму трудозатраты на создание спецификаций требований. Если вы используете инструмент **Rational Requisite PRO**, то для него существует специальный инструмент **Rational SoDA for Word**. Он имеет заранее подготовленный набор стандартных шаблонов спецификаций, который вы можете применять, чтобы получать спецификации в формате **Microsoft Word**. Однако вы должны придерживаться определенных правил создания моделей и работы с проектом требований, иначе эти шаблоны не смогут корректно прочитать ваши данные и сгенерировать отчет.

Наряду с использованием предопределенных шаблонов вы можете создавать собственные отчеты для **Rational SoDA for Word**. Инструмент имеет достаточно понятный интерфейс, который позволит быстро научиться строить отчеты любой сложности. Кроме того, вы можете открыть любой из предустановленных шаблонов отчетов, изменить их и сохранить уже как свой собственный шаблон. На сайте книги вы найдете презентацию «Возможности **Rational SoDA** и полезные шаблоны», а также некоторые готовые шаблоны для генерации спецификаций требований.

### 3.2.12. Понимать основные принципы тестирования

На этом этапе вы должны понимать, что тестируется не только законченный программный продукт, но и требования. Это происходит в форме ревью спецификаций требований со стороны архитектора системы и разработчиков, а также тест-дизайнеров и тест-менеджера.

Тест-менеджер — специальная проектная роль. Он отвечает за качество продукта в целом. В его обязанности входит создание стратегии обеспечения качества продукта и стратегии тестирования. Данная стратегия, в частности, определяет виды используемого в проекте тестирования (функциональное, нагрузочное, интеграционное, ретроспективное, приемочные испытания) и методику прове-

дения каждого из них. Аналитик должен ознакомиться с документами, чтобы понимать, какие результаты от него потребуются для реализации этих стратегий. Кроме того, тест-менеджер отвечает за объемы тестирования в проекте, и именно он, совместно с менеджером проекта, определяет сценарии тестирования и список создаваемых тест-кейсов.

Тест-дизайнер — проектная роль. Он отвечает за создание сценариев тестирования и тест-кейсов. Основная задача создания сценария тестирования и тест-кейсов — обеспечить возможность для тестирующего проверить реализованное поведение системы на соответствие требованиям и вариантам использования системы. Как правило, один вариант использования проверяется несколькими сценариями тестирования. В идеале аналитик должен выявить все возможные разовые проходы по сценарию варианта использования без ветвления и альтернативных потоков для каждого Use Case (Use Case Realization). В этом случае тест-дизайнер создает тестовые сценарии, базируясь на соответствующих Use Case Realization. **Однако в реальной жизни такие трудозатраты в проект стараются не включать, и тест-менеджер или тест-дизайнер создает сценарии тестирования, базируясь на собственном понимании вариантов использования.**



---

Отсюда можно сделать следующие выводы:

- ◆ желательно создавать как можно более простые и однозначно трактуемые сценарии вариантов использования;
  - ◆ необходимо проводить передачу знаний (рассказывать) о разработанных требованиях и вариантах использования тест-менеджерам и тест-дизайнерам;
  - ◆ тест-менеджер/тест-дизайнер должен участвовать в согласовании вариантов использования и требований и проверять их на возможность тестирования;
  - ◆ аналитик должен участвовать в согласовании разработанных сценариев тестирования и тест-кейсов.
- 

На практике это выливается в то, что вы будете тратить свое время на незначительные правки требований, чтобы тест-менеджер признал их тестируемыми, а также на консультирование тест-дизайнеров и на ревью спецификаций тестирования даже после того, как выполнили основную работу с требованиями. Учитывайте это при планировании своего рабочего времени. И помните: разработав требования, вы не заканчиваете работу в проекте.

### 3.2.13. Полезные мелочи

Не бойтесь экспериментировать с нотациями. Если вы не знаете какой-то общепринятой мировой нотации, подходящей для конкретного случая, придумайте совместно с командой свою и используйте ее.

Не пренебрегайте инструментом Microsoft Excel. Вы не раз столкнетесь с тем, что люди будут бояться работать с Rational Requisite PRO или с другим инструментом

управления требованиями. Microsoft Excel позволяет организовать «плоское» управление требованиями вполне эффективно. Он может применяться для выявления и согласования запросов заинтересованных лиц, обсуждения архитектурных проблем и рисков, ведения всевозможных реестров (проблем, открытых вопросов). Для управления требованиями лучше использовать специализированные инструменты.

Будьте готовы к тому, что вам придется для себя и команды управлять требованиями в **Rational Requisite PRO**, а для отдельных заинтересованных лиц проекта и представителей заказчика готовить спецификации требований или другие представления требований (например, в формате Microsoft Excel). Это потребует от вас дополнительных затрат времени и сил. Посему...



Упрощайте себе жизнь. Создавайте маленькие программы для собственных нужд. Например, программу, которая из MS Excel может создать требования в Rational Requisite PRO, программу, которая найдет все изменения, сделанные кем-то в требованиях, и т. д. Для этого достаточно знать язык программирования Microsoft Visual Basic for Applications, понимать принципы объектно-ориентированного программирования и изучать API библиотеки продуктов, для которых вы будете создавать такие вспомогательные программы. Для инструмента Rational Requisite PRO можно найти много полезных скриптов на сайте <http://www.ibm.com/developerworks/rational/library/3784.html>. А на сайте <http://www.ibm.com/developerworks/rational/library/445.html> приведено описание библиотеки расширения от Rational Requisite PRO и примеры создания собственных скриптов.



Не бойтесь использовать флипчарты, электронную доску или бумагу и карандаш при обсуждении и «мозговых штурмах». Стройте модели от руки, дискутируйте — это всегда быстрее, чем отрисовывать модели в специализированных программах (например, в Rational Rose). Но имейте в виду, что все такие «зарисовки» — проектные материалы. Подшивайте и храните их. Это тоже история изменения требований.

На сайте книги вы можете скачать мой конспект UML и Rational Unified Process, который я делал, когда начинал изучение RUP [5] и использование практик моделирования в системном анализе. Этот конспект не содержит какой-то принципиально новой информации, но поможет лучше понять, как связаны между собой различные диаграммы UML.



### 3.3. Типичные проблемы и вопросы

В этом разделе приведены наиболее типичные проблемы и вопросы, с которыми сталкивается начинающий аналитик.

Вопрос / проблема	Рекомендация / комментарий
Они не приходят на встречу!	<p>Это нормально. Отставить панику. Вежливо, но настойчиво добивайтесь проведения встречи. Подстраивайтесь под заинтересованное лицо. Попросите предложить удобное время и место встречи.</p> <p>Если проблема не решается — обратитесь к менеджеру проекта за помощью</p>
Они сами не знают, чего хотят!	<p>Я понимаю ваше негодование, но позволю себе напомнить о том, что любая разрабатываемая ИС существует не сама по себе, а в какой-то конкретной конечной ИТ-архитектуре, регламентируемой ИТ-стратегией, согласованной с ней и зависящей от бизнес-архитектуры, которая, в свою очередь, динамично и постоянно изменяется под влиянием рынка и целей бизнеса. Следовательно, это абсолютно нормальная ситуация.</p> <p>Информируйте менеджера проекта о том, что заинтересованное лицо опять хочет изменить требования, — вы должны провести импакт-анализ. Менеджер проекта, возможно, инициирует переоценку проекта, и, если заказчик согласится платить за дополнительные работы, придется выполнить очередной его «каприз»</p>
Непонятно, кто с кем контактирует и кто за что отвечает.	<p>Если в проекте есть план управления коммуникациями, ознакомьтесь с ним, и вы найдете ответы на свои вопросы.</p> <p>Если такого плана нет, делегируйте вопрос менеджеру проекта — это его зона ответственности</p>
А за что отвечаю я?	<p>Руководствуйтесь должностной инструкцией, базовыми правилами работы отдела, процедурами и рабочими инструкциями системы менеджмента качества компании.</p> <p>Если этих документов нет, делегируйте вопрос начальнику аналитического отдела — это его зона ответственности</p>
Почему процесс разработки строится не по RUP (подставьте наименование любой другой методологии, которой вас учили и которую вы считаете эффективной)?	<p>Это нормально. Нельзя взять какую-то «чистую» методологию и начать ее использовать в компании.</p> <p>При выборе методологии разработки ПО надо учитывать следующие аспекты:</p> <ul style="list-style-type: none"> <li>• какой продукт мы разрабатываем: «коробочный», тиражируемый или разово поставляемое приложение;</li> <li>• есть ли возможность построить scrum team, или у нас сложные коммуникации и требуется серьезное управление рисками;</li> <li>• достаточная ли квалификация у проектной команды, чтобы выполнять работы без создания формальных спецификаций, не требует ли заказчик исчерпывающих спецификаций для будущего развития системы, не понадобится ли сертифицировать систему в дальнейшем;</li> </ul>

(Продолжение)

Вопрос / проблема	Рекомендация / комментарий
	<ul style="list-style-type: none"> <li>разрабатываемая система скорее ориентирована на взаимодействие с пользователем или с другими системами и т. д.</li> </ul> <p>Предлагайте улучшения методологии и применяемых практик в тех проектах, в которых участвуете, и представляйте ваши предложения начальнику аналитического отдела</p>
<p>Что за странные спецификации требований?</p>	<p>Первое время старайтесь не анализировать состав и структуру спецификаций требований. Обращайте внимание на суть, на то, какие типы требований есть в спецификациях. Возможно, именно в таких спецификациях нет никакого глубинного смысла и, как это часто бывает, они просто «исторически сложились» в таком виде.</p> <p>В проектах, где вы будете участвовать, предлагайте структурировать информацию спецификаций, которые будете изменять</p>
<p>Да кому нужны эти ваши требования? Мы писали спецификации в Microsoft Word в течение пяти лет, и все было нормально</p>	<p>Если не планируется управлять требованиями и трассировать их друг на друга ни в этом конкретном проекте, ни на уровне всего продукта, то и выделять их как требования не имеет смысла. В данном случае можно обойтись обычными спецификациями в формате Word.</p> <p>Иначе говоря, необходимость создания требований диктуется методологией разработки.</p> <p>Принятие таких решений не в компетенции аналитика. Эти решения принимает продуктовый комитет в соответствии с утвержденными в компании процессами, разработанными Software Engineering Process Group</p>
<p>Что дальше с точки зрения Пути аналитика?</p>	<p>На этом этапе вашей карьеры вариантов немного: вы можете либо перешагнуть в следующую категорию — «Аналитик» и остаться на Пути аналитика, либо уйти в область бизнес-анализа и стать бизнес-аналитиком, либо податься в разработчики</p>

## 3.4. Заключение

В этой главе мы проанализировали основополагающие методы и приемы работы аналитика, необходимые ему личностные навыки, рассмотрели несколько поясняющих примеров.

Кроме того, последовательно прошли этапы разработки требований аналитиком и выяснили, какие специальные и профессиональные навыки помогут ему выполнять эту работу максимально быстро, качественно и эффективно.

Как и обещал, в вводной части я не рассматривал подробно теорию, но настоятельно рекомендую вам продолжить чтение книги только после того, как вы ознакомитесь с теоретическими источниками (табл. 3).

**Таблица 3.** Профессиональные и специальные навыки младшего аналитика

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
<p><b>Младший аналитик</b></p>	<p>Книгу Дина Леффингвелла и Дона Видрига <i>Managing Software Requirements</i> я называю «библией аналитика». Помимо теоретической информации, вы найдете здесь ряд конкретных примеров и приемов проведения интервью, «мозговых штурмов» и т. д.</p> <p>Еще одна, более поздняя и не менее «евангелистская», книга в области анализа – «Разработка требований к программному обеспечению» Карла И. Вигерса.</p> <p>Иметь общее представление о различных методологиях разработки ПО.</p> <p>Знать основы RUP – дисциплину Requirements.</p> <p>Знать и уметь применять UML – Use Case-модель, уметь строить Domain Object Model.</p> <p>Понимать основные принципы объектно-ориентированного проектирования и моделирования.</p> <p>Рекомендую также почитать статьи по вопросам анализа на сайте <a href="http://www.interface.ru">www.interface.ru</a></p>	<p>Выявлять ЗЛ.</p> <p>Управлять ожиданиями ЗЛ.</p> <p>Проводить совещания.</p> <p>Проводить интервьюирование.</p> <p>Проводить анкетирование.</p> <p>Проводить «мозговые штурмы».</p> <p>Уметь определять границы системы.</p> <p>Уметь выделять подсистемы и определять их границы.</p> <p>Уметь выявлять требования типов:</p> <ul style="list-style-type: none"> <li>• ответы и собранная информация;</li> <li>• запросы заинтересованных лиц;</li> <li>• глоссарий;</li> <li>• стандарты и ГОСТы;</li> <li>• характеристики аналогичных/наследуемых систем;</li> <li>• бизнес-требования;</li> <li>• бизнес-правила;</li> <li>• концепция создания и развития продукта (BVISION);</li> <li>• ограничения и допущения;</li> <li>• концепция системы (TVISION);</li> <li>• пользовательские требования;</li> <li>• функциональные требования;</li> <li>• функции системы/варианты использования/прецеденты (Use Cases);</li> <li>• нефункциональные требования;</li> <li>• требования к пользовательскому интерфейсу;</li> </ul>

Продолжение ⇨

(Продолжение)

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
		<ul style="list-style-type: none"><li>• требования к взаимодействию с внешними системами.</li></ul> Выявлять функции системы (Use Cases), моделировать поведение системы. Уметь строить трассировки/прослеживать требования. Понимать основные принципы тестирования. Знать английский язык на уровне, достаточном для чтения технической литературы

# 4. Аналитик

---

---

## 4.1. Личностные навыки

В этом и последующих разделах книги, посвященных личностным навыкам, приведены выученные уроки из моей производственной практики и выводы из книги «Трудный выбор. Уроки бескомпромиссного лидерства в сложных ситуациях» от экс-главы Hewlett-Packard Карли Фиорины [17].

### 4.1.1. Самомотивация

Любой человек может научить вас очень многому, лишь бы вы сами хотели учиться. Не отвергайте ничей опыт, не осмыслив идеи, которые в нем скрыты, и не подвергнув критическому анализу свои методы работы. Обращайте внимание не только на профессиональные навыки ваших коллег, но и на их манеру говорить, жестикулировать, вести себя в сложных ситуациях. Замечайте лучшее в других и ставьте цели по развитию этих навыков у себя.

Не думайте о следующей работе, сосредоточьтесь на том, чтобы выполнять свои обязанности как можно лучше на текущей работе. Цели в жизни важны, но это еще не все. В жизни важнее дорога, а не прибытие в конечный пункт. Не будьте карьеристами. Старайтесь развиваться гармонично, даже если для этого вам придется делать шаги в сторону от намеченных целей, ведь именно способность сделать шаг в сторону от дороги делает нас теми, кто мы есть.

Развивайте в себе чувство «драйва», умения загораться идеей и активно воплощать ее в жизнь, стремитесь «двигать дело».

### 4.1.2. Самоорганизация

#### Планируйте

Планируйте не только свою ежедневную деятельность, но и профессиональное и личностное развитие. Не нужно составлять детальные планы с большим количеством деятельностей и взаимосвязей, с горизонтом планирования на 2–3 месяца.

Подобные планы, как правило, не реализуются и не приносят большой пользы. Планируйте на уровне целей. А детализацию целей на уровне деятельности выполняйте по мере того, как вы начнете работать над достижением этих целей. Это позволит вам увидеть всю картину целиком и не тратить много времени на собственно создание плана вместо того, чтобы заниматься его реализацией. Такое планирование называется планированием методом набегающей волны.



Я не рекомендую планировать выполнение какой-то одной деятельности в течение всего рабочего дня. Гораздо эффективнее выполнять одновременно две-три задачи. Более приоритетной или самой сложной задаче имеет смысл уделить 4–5 часов рабочего времени, а оставшиеся 3–4 часа — другим, менее приоритетным/более легким задачам.

## Выполняйте планы

Иногда бывает трудно заставить себя что-то делать по плану, потому что есть другие, более интересные, задачи. И вот уже сработало напоминание, что надо переключиться на другую задачу, а соблазн «сейчас, вот-вот закончу» очень велик. И это «вот-вот» незаметно превращается еще в три часа, после чего на вопрос начальства, почему не выполнена «скучная» работа, ответить нечего.



Я настойчиво рекомендую вам придерживаться вашего же ежедневного плана. Если напоминание о переключении сработало в момент «полета мысли», остановитесь на две минуты и определите время, достаточное, на ваш взгляд, для того, чтобы тезисно записать то, что у вас на тот момент будет на уме, в качестве напоминаний на будущее, когда вы вернетесь к выполнению текущей работы. Не выходите за рамки этого времени. Даже если вы не успели записать все, заставьте себя остановиться и переключиться на другую задачу. Для более легкого переключения рекомендую встать и пройтись, посмотреть в окно, дав отдохнуть глазам, сделать пару физических упражнений.

Развивайте способность быстро переключаться между задачами. Это очень важный навык не только для аналитика, но и для любого современного специалиста.

Недельный план аналитика, который стремится к развитию и профессиональному росту, может быть построен примерно так.

День недели	Рабочее время	Оставшееся время
Понедельник	1 час утром на изучение новостей и статей по ИТ и методологиям разработки ПО в Интернете. 1 час до обеда на планирование своей рабочей недели. 6 часов работы по рабочему плану	30 минут после работы — краткое подведение итогов дня, коррекция планов на неделю. 1 час вечернего чтения профессиональной литературы по плану собственного развития

День недели	Рабочее время	Оставшееся время
Вторник	8 часов работы по рабочему плану	15–30 минут после работы — краткое подведение итогов дня, коррекция планов на неделю. 1 час вечернего чтения профессиональной литературы по плану собственного развития. Вечернее чтение художественной литературы
Среда	8 часов работы по рабочему плану	15–30 минут после работы — подведение итогов дня, коррекция планов на неделю. 1–2 часа после работы на изучение новостей и статей по ИТ и методологиям разработки в Интернете. Вечернее чтение профессиональной литературы по плану собственного развития
Четверг	8 часов работы по рабочему плану	15–30 минут после работы — краткое подведение итогов дня, коррекция планов на неделю. 1 час — систематизация ваших текущих знаний и навыков. Вечернее чтение художественной литературы
Пятница	7 часов работы по рабочему плану. 1 час после обеда на анализ выполнения собственного плана на эту рабочую неделю	Отдых и развлечения (это не значит, что в другие дни не должно быть отдыха или развлечений — вам лучше знать возможности вашего организма в отношении его выносливости, но в пятницу я рекомендую на время забыть о профессии и сменить область деятельности)
Суббота		Подведение итогов недели, работа над ошибками и пополнение собственной БД выученных уроков
Воскресенье		Чтение художественной литературы

Руководители многих компаний понимают, что мы живем в эпоху экономики знаний и что инвестиции в знания сотрудников — это часть капитализации

компании. В таких организациях выделяются дополнительные часы на самообучение за счет рабочего времени. Я рекомендую вам проводить такое самообучение в пятницу после анализа выполнения плана на текущую рабочую неделю.

При всей кажущейся напряженности этого графика он более чем выполним. Вы должны понимать, что чем дальше продвинетесь по Пути аналитика, тем меньше у вас будет времени на собственное профессиональное развитие и чтение художественной литературы.

Мне, как правило, приходилось работать до 2–3 ночи в течение первых 3–5 месяцев с момента выхода на новое место работы, где стояла задача сформировать аналитический отдел. Впоследствии рабочий день нормализовался, но редко когда он бывает менее 10–12 часов. Конечно, я не все время провожу на работе, но такова примерная доля моего личного времени, которое я отдаю работе.

Поэтому старайтесь заранее заложить максимально возможную основу в самих себя, пока вы молоды, энергичны и у вас есть на это время. Это ваш фундамент, и эти затраты в дальнейшем окупятся.

### 4.1.3. Не упускать важное

В процессе работы вы будете сталкиваться с ситуациями, когда невозможно найти ответы на какие-то вопросы «прямо сейчас», но нельзя ни в коем случае «забыть» об этих вопросах. Кроме того, появятся обязательства и задачи, которые вы не сможете выполнить «немедленно» в силу своей большой загрузки, но выполнить их нужно. Как научиться не забывать? Мой совет прост: записывайте.



Я активно использую инструмент Microsoft Outlook в своей работе. Для планирования совещаний пользуюсь функционалом организации встреч, назначая встречу в календаре. Этот же инструментарий задействую для планирования собственных задач, о которых не хочу забыть. Я назначаю встречу «сам себе». Плюсы такие: во-первых, наглядно, я всегда вижу, когда и что у меня в течение рабочей недели в режиме просмотра календаря на пять дней; во-вторых, Outlook заранее напомнит о такой «встрече» и у меня будет возможность закончить предыдущую работу и приступить к запланированной.



Немало информации содержит и переписка. Я помечаю важные письма специальными флажками в Outlook. Таким образом, вероятность того, что я «потеряю» какую-то важную информацию, стремится к нулю. Я помечаю письма в Outlook таким образом:

- ◆ красный флажок — надо выполнить мне;
- ◆ синий флажок — я выполнил, надо проконтролировать;
- ◆ желтый флажок — надо проконтролировать выполнение и результаты;
- ◆ зеленый флажок — важные ссылки, принятые решения;
- ◆ оранжевый флажок — проблемы для поиска решений;
- ◆ лиловый флажок — кандидаты в хорошие практики.

Еще один маленький секрет — письма самому себе с последующей отметкой флажком. Это самый простой способ вести электронную каталогизацию информации. Раз в неделю я стараюсь выделять время и, как правило, при планировании своей следующей рабочей недели «превращать» эти письма в задачи в своем плане, а с самих писем снимать выставленный ранее флажок.

#### 4.1.4. Культура речи

Читайте русских классиков художественной литературы. Наши коммуникативные способности во многом обусловлены словарным запасом и умением строить предложения. Вы не замечали, что во время чтения какой-либо книги ваши мысли и ваша речь «перенимают» стиль этой книги? Иногда это влияние выражено более явно, иногда менее, но оно есть. Чтение русской классической литературы обогатит ваши мысли и речь незаметно для вас самих. Читайте. Как можно больше читайте.

#### 4.1.5. Корпоративная культура и этика

В компании должен существовать документ, регламентирующий базовые (основные) правила работы и концепцию коммуникаций в компании. Однако такие документы есть не во всех компаниях.



Я рекомендую вам всегда помнить и стараться придерживаться определенных общих принципов этических и культурных норм.

1. Принцип адекватного реагирования: старайтесь сами и мотивируйте других вести себя в ключе: «что сделано, то сделано — давайте подумаем вместе, *как можно исправить ошибки и недоработки*».
2. Принцип конструктивной критики: «критикуя — предлагай».
  - А. Недопустимо высказывание замечаний в неуважительной и неконструктивной форме.
  - Б. Недопустимо высказывание замечаний общего характера («документ никуда не годится»), без конкретизации, в чем именно и где конкретно допущены ошибки или сделаны недоработки.
  - В. Недостаточно и простого формального указания на ошибку (например, «в варианте использования “Аутентификация пользователя” у вас неверно описан альтернативный поток») без информирования о том, что конкретно не так и что нужно исправить или какие требования надо выполнить.
3. Принцип постоянного улучшения процессов работы: старайтесь сами и мотивируйте других вести себя в ключе: «что сделано, то сделано — давайте подумаем вместе, *как лучше построить процесс, чтобы в будущем избежать подобных проблем и недостатков*».

В заключение приведу несколько примеров из своей персональной базы выученных уроков.



---

Не обсуждать решения другого руководителя с его подчиненными.

Хорошие формы возражений:

- У меня есть другое мнение насчет такого решения...
- Я не могу сказать, что однозначно одобряю подобный подход...
- Мне кажется, что есть более приемлемые/удачные варианты....

Не давать оценок профессиональной компетенции любого сотрудника перед лицом других сотрудников (тем более его подчиненных).

Все разговоры по душам вести вне офиса.

---

## 4.1.6. Полезные мелочи

Старайтесь придерживаться здравого перфекционизма в работе. Ищите компромиссы и идите на них. Даже если знаете, что «по науке» надо делать не так, но есть причины сделать не все идеально, будьте гибкими, нарушайте «научные» правила.

Развивайте в себе высокую работоспособность и производительность. Способность быстро выдавать качественный результат — ваш основной капитал.

Развивайте память. Аналитик должен помнить то, что не помнят многие.

## 4.1.7. Продолжайте...

Вести собственную базу выученных уроков, в которую заносите все сложные ситуации, которые будут возникать в вашей жизни. Анализируйте эти ситуации и вырабатывайте персональные лучшие практики.

Повышать уровень знания английского языка. Читайте статьи на английском в Интернете. Например, на <http://msdn.microsoft.com>.

Выполнять план собственного профессионального и личного развития. Пополняйте ваше хранилище знаний. Раз в полгода пересматривайте это хранилище, удаляйте лишнее и отмечайте собственный прогресс и рост. Всегда делитесь полученными знаниями с вашими коллегами. Делайте это бескорыстно, по пословице: «Делай добро и бросай его в воду».

## 4.2. Профессиональные и специальные навыки

На этом этапе своего карьерного и профессионального пути вы должны обладать следующими навыками (табл. 4).

Таблица 4. Профессиональные и специальные навыки аналитика

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
<b>Аналитик</b>	<p>Все теоретические знания младшего аналитика, а также:</p> <ul style="list-style-type: none"> <li>• знать и уметь применять ГОСТ 34-й и 19-й серий;</li> <li>• знать и уметь применять нотацию IDEF0, диаграммы EPC (extended Event Process Chain).</li> </ul> <p>Книги:</p> <ul style="list-style-type: none"> <li>• <i>Rosenberg Doug, Scott Kendall. Use Case Driven Object Modeling with UML: A Practical Approach;</i></li> <li>• <i>Rosenberg Doug. Agile Development with Iconix Process.</i></li> </ul> <p>Знать <b>основы RUP — дисциплины Requirements, Analysis &amp; Design.</b></p> <p>Знать и уметь <b>строить robustness и sequence diagrams, analysis model.</b></p> <p>Расширять свой кругозор в методологиях. Познакомиться с гибкими методологиями, например Iconix.</p> <p>Рекомендую регулярно посещать сайты <a href="http://www.uml2.ru">www.uml2.ru</a>, <a href="http://www.agilerussia.ru/">http://www.agilerussia.ru/</a></p>	<p>Все навыки младшего аналитика, а также:</p> <ul style="list-style-type: none"> <li>• знать, что такое ПУТ, и уметь его разрабатывать;</li> <li>• понимать, какие модели существуют и их место в разработке ПО;</li> <li>• уметь создавать модель анализа;</li> <li>• строить <b>Robustness- и Sequence-диаграммы</b>, понимать, зачем их вообще надо строить;</li> <li>• уметь читать программный код;</li> <li>• иметь навыки проведения презентаций</li> </ul>

### 4.2.1. Проектные коммуникации

Проектные коммуникации должны строиться в соответствии с планом управления коммуникациями, который является составной частью плана управления проектом (ПУП). Если в проекте есть такой план, обязательно ознакомьтесь с ним. Коммуникации — это «кровь проекта», и от них зависит очень многое.

Я не буду, как и обещал, уходить в теорию и рассказывать вам о процессе передачи информационного сообщения, возникающих искажениях при коммуникации, о том, что такое коммуникационные требования и что за формула  $K = N \times (N - 1) / 2$ . Об этом вы можете узнать сами, почитав литературу по управлению проектами. Вместо этого я приведу несколько конкретных уроков и практик из своей персональной базы выученных уроков.



### **Общие практики**

*Использовать мягкие формулировки собственного мнения, даже если оно категоричное:*

- Я думаю, что...
- Создается ощущение, что...
- По моему мнению, нам лучше...

*Добиваться четкой постановки задач и определения целей их выполнения.*

*При постановке задачи проводить оценку трудоемкости и уведомлять о времени, необходимом для выполнения задачи.*

### **Переписка**

*Определять целевую аудиторию для переписки/участия в совещаниях/участия в обсуждении задач.*

В переписках, в которых рано или поздно вопрос будет вынесен на уровень руководителя, сразу ставить руководителя в копию для понимания им полной картины происходящего.

Обращать внимание на то, кому отправлено письмо. От того, кто выбран в качестве адресатов письма, зависит не только стиль вашего письма или ответа, но и информация, которую вы имеете право отразить в письме на такую аудиторию.

При просьбе к инициатору совещания добавить кого-то в качестве участника совещания убедиться, что инициатор совещания понял, с какой целью просят добавить кого-то.

*Изучать контекст любого сообщения.*

До ответа на любое сообщение (письмо, задачи в системе управления задачами, обсуждения в системе управления запросами и т. д.) изучать всю информацию «вокруг» сообщения — вложения, связанные задачи и работы, историю переписки. Если такого изучения не получилось (нет времени) — отвечать всегда только на прямо заданные вопросы и уведомлять о том, когда будет подробный ответ.

*На вопросы в письме отвечать всегда только по существу, в деловой форме.*

Не втягиваться в письменные обсуждения, особенно если обсуждение происходит в негативной манере.

*Не поддаваться на провокации.*

Если в письменном сообщении содержится провокация — иногда допускается ответить в шуточной форме, но очень аккуратно. Лучше по возможности игнорировать письмо.

*Перед отправкой своего «гневного» сообщения сделать перерыв.*

В течение перерыва заняться чем-то другим или выйти на свежий воздух и вернуться к вопросу отправки письма не ранее чем через 15 минут, предварительно перечитав его. При письменном ответе можно использовать юмор или сарказм.

*Стараться переводить конфликтную переписку в плоскость устного общения.*

На подобных встречах:

- ◆ улыбаться;
- ◆ сначала выслушивать точку зрения оппонента;
- ◆ формализовывать мысли оппонента и отсекаать эмоции:  
— Я правильно понял, что ты имеешь в виду то-то и то-то?..  
— Я могу записать эту мысль таким образом: ... ?

Записывать мысли оппонента.

Не давать оценок.

Доносить свою позицию, но не доказывать, что только эта позиция правильная.

Выявлять разницу между позициями.

Формально фиксировать разницу в форме открытых вопросов.

Назначать вторую встречу для снятия открытых вопросов.

Максимально допустимое выражение недовольства:

- ◆ дать оценку тону сообщения: «если не было цели “зацепить” и оскорбить адресата, то таких-то и таких-то словосочетаний прошу в будущем избегать, поскольку трактую их как личный и неконструктивный “выпад”»;
- ◆ «выпады» не нужны, потому что у нас общие цели;
- ◆ не стоит подключать дополнительную аудиторию и вышестоящее руководство, если можно решить вопрос без их участия;
- ◆ и все-таки лучше не писать сообщения в подобном стиле, а поговорить лично.

*В переписке, при ее неконструктивности, отвечать: «Я не готов продолжать обсуждение этого вопроса»/«Я не готов продолжать обсуждение в таком русле».*

### **Встречи и совещания**

*Тщательно проверять повестку совещания и обращать внимание на то, чтобы из повестки было однозначно понятно, что конкретно будет обсуждаться.*

Например, пункт повестки «Обсуждение тестовых сценариев, которые будут использоваться для проверки первой версии архитектурного прототипа» трактовался тремя участниками собрания как непосредственное обсуждение того, как будет тестироваться прототип. На самом деле целью собрания было определить, что будет тестироваться в прототипе, не тратя времени на обсуждение, каким образом все будет осуществляться. Это пример того, как нечетко сформулированная повестка привела к непониманию и в конечном итоге к конфликту в команде «на ровном месте».

*Собирать отзывы после проведения сложных проблемных совещаний.*

*Хорошее сопроводительное письмо к сбору отзывов:*

«С целью обратной связи и выученных уроков прошу вас прислать отзыв о проведенном совещании на тему ..., которое состоялось ..., в двух-трех словах. Была ли встреча в целом продуктивной, полезной, что нужно изменить в проведении таких совещаний, как вы оцениваете практическую пользу совещания? Как можно было более эффективно провести встречу? Имеет ли смысл проводить подобные встречи в дальнейшем?»

Если на основе собранных отзывов будет оформляться сводный отчет или материалы ответов будут распространены далее, надо указать на это в сопроводительном письме.

## 4.2.2. Проверьте себя

Перед тем как продолжить чтение книги, проверьте себя. Ниже представлены утверждения — кандидаты в требования, постарайтесь определить тип каждого требования.

№ п/п	Утверждение	Тип требования
1	Должны создаваться связи между разными группами пользователей	
2	Пользователь видит в адресной книге только тех пользователей, с которыми контактировал в течение последних 20 дней, остальных — только если выбрать опцию «Показать всех»	
3	Версия программы для MS Windows не может иметь функциональность меньшую, чем существующая версия программы под MS DOS	
4	Новая версия программы должна позволить компании удерживать существующих клиентов и укрепиться на рынке	
5	Должна обеспечиваться устойчивая работа системы при наличии до 1000 одновременно работающих пользователей	
6	Должна обеспечиваться совместимость с MS DOS-версией программы при обмене конфигурационными файлами	
7	Действия, доступные для пользователей: <ul style="list-style-type: none"> <li>• создание пользователя;</li> <li>• удаление пользователя</li> </ul>	

**Ответ:** 1 — функциональное требование (FR); 2 — пользовательское требование (UREQ). В результате анализа этого требования будут выделены два варианта использования (Use Cases): а) просмотреть адресную книгу (актор — пользователь); б) составить список последних контактов (Included Use Case для Use Case — «Просмотреть адресную книгу»); 3 — ожидание заинтересованного лица (STKR). Это очень коварное ожидание. Потребуется провести огромную работу по выявлению (реверс-инжинирингу) требо-

ваний DOS-версии. Результат – функциональные, нефункциональные требования и варианты использования; 4 – ожидание заинтересованного лица (STKR). Это также весьма каверзное ожидание. На самом деле в нем скрыт «клад» из бизнес-требований и бизнес-целей. Требуется выявить бизнес-требования, бизнес-цели. Согласовать с ЗЛ; 5 – нефункциональное требование (NFR); 6 – допущение и ограничение (ASSUM); 7 – варианты использования (UC).

Если вы испытывали сложности при заполнении этой таблицы или если после прочтения ответов у вас возникло непонимание, почему типы требований распределены именно так, рекомендую посмотреть материалы раздела «Профессиональные и специальные навыки» главы «Младший аналитик». Если же после прочтения ответа у вас не осталось серьезных сомнений и вопросов – давайте пойдем дальше.

### 4.2.3. План управления требованиями



План управления требованиями — это контракт между аналитиком и менеджером проекта на выполнение аналитических работ. Я всегда рекомендую создавать план управления требованиями (ПУТ) в проектах, хотя бы в форме протокола совещания, чтобы зафиксировать то, какие требования будут разрабатываться в проекте, как и кто будет ими управлять и как будут проводиться согласование и утверждение требований.

ПУТ входит в состав плана управления проектом (ПУП) и должен соответствовать плану управления документами (ПУД) проекта. О том, что такое ПУП и ПУД, мы поговорим позднее, в разделе «Старший/ведущий аналитик». Сейчас же вам надо знать о существовании таких документов и учитывать их при разработке ПУТ.

ПУТ является не только контрактом на выполнение работ аналитика, но и основанием для проведения аудитов качества процессов разработки и управления требованиями в проекте.



Ниже приведен возможный состав ПУТ.

1. Введение.
2. Общее описание методологии анализа, разработки и управления требованиями в проекте.
3. Системный анализ и моделирование.
  - 3.1. Обзор основных моделей системного анализа.
  - 3.2. Управление артефактами (моделями) системного анализа.
4. Разработка требований.
  - 4.1. Типы требований.
  - 4.2. Типизация нефункциональных требований.
  - 4.3. Атрибуты требования.

5. Управление требованиями.
    - 5.1. Методология управления.
      - 5.1.1. Атрибуты требований.
      - 5.1.2. Полномочия по работе с требованиями.
      - 5.1.3. Обсуждение требований.
      - 5.1.4. Согласование требований.
      - 5.1.5. Утверждение требований.
    - 5.2. Жизненный цикл требований.
      - 5.2.1. Запросы заинтересованного лица.
      - 5.2.2. Все остальные типы требований.
    - 5.3. Трассировка требований.
  6. Управление изменениями в требованиях.
    - 6.1. Обработка запроса на изменение.
    - 6.2. Базовые версии требований (baselines) в проекте.
  7. Спецификации требований.
  8. Управление процессом анализа и разработки требований.
    - 8.1. Список основных артефактов и деятельности по управлению требованиями.
    - 8.2. Передача знаний бизнес-аналитику.
    - 8.3. Обучение и передача знаний системному аналитику.
    - 8.4. Постоянные улучшения процесса анализа и разработки требований.
- 

Как вы уже поняли, ПУТ — самый важный управленческий документ для аналитика в проекте. Обычно в компании существует шаблон ПУТ, адаптация которого под проектные нужды является одной из задач аналитика. На сайте книги вы можете скачать шаблон ПУТ для водопадного жизненного цикла разработки.

Следующие два раздела главы: «Концептуальная модель типов требований» и «Общие принципы управления требованиями» — сокращенные разделы шаблона ПУТ. Эти разделы наиболее важны для аналитика с точки зрения разработки и управления требованиями.

Создав ПУТ в проекте, аналитик выполняет системный анализ в соответствии с этим управленческим регламентом. Если требуется внести изменения в процесс анализа (например, команда решила отказаться от использования какого-то типа требований или изменить способ согласования/утверждения требований), то сначала должен быть изменен и заново согласован ПУТ.

Еще раз напоминаю, что ПУТ может не быть оформлен как отдельный документ, а представлять собой обычный протокол совещания по методам разработки

и управления требованиями. И все же я рекомендую создавать именно документ, согласовывать его с архитектором, тест-менеджером и менеджером проекта и обязательно утверждать у спонсора проекта. Это позволит быть уверенным, что требования будут управляемыми, достаточными для разработки архитектуры и обеспечения качества и что спонсор проекта согласен за все это платить.

## 4.2.4. Концептуальная модель типов требований

В этом разделе приводится описание концептуальной модели типов требований и их атрибутов.

### Типы требований



В табл. 5 полужирным шрифтом выделены **основные** типы требований. Эти типы требований я называю основными, потому что их вполне достаточно для разработки системы, и если нет каких-либо других причин для управления более «мелкими» требованиями, лучше ограничиться таким набором.

Таблица 5. Типы требований

Название требования	Аббревиатура	Краткое описание/назначение
Ответы и собранная информация — Answers	ANSW	Сохранить оригинальный ответ, проследить, как он учтен в требованиях, обосновать принятие решений и формулировку требований
Запросы заинтересованных лиц — Stakeholders' Requests	STKR	<b>Сформулировать непротиворечивый и полный перечень запросов заинтересованных лиц.</b> <b>Трассировки на BR и BRULE являются проверочными</b>
Бизнес-требования — Business Requirements	BR	Сформулировать непротиворечивый и полный перечень бизнес-требований (высокоуровневые цели организации в качестве заказчика системы, которые определяют, почему организации нужна такая система, то есть описывают бизнес-цели, которых организация намерена достичь с ее помощью)

Продолжение ⇨

(Продолжение)

Название требования	Аббревиатура	Краткое описание/назначение
Бизнес-правила — Business Rules	BRULE	Сформулировать непротиворечивый и полный перечень бизнес-правил (положения, определяющие или ограничивающие какие-либо стороны бизнеса с целью защитить структуру бизнеса, контролировать или влиять на его операции)
Глоссарий — Terminology	TERM	Термины и понятия системы
Ограничения и допущения — Assumptions	ASSUM	Выявленные системным аналитиком внешние ограничения на систему со стороны заинтересованных лиц, бизнес-требований и бизнес-правил
Технологические особенности — Technology Characteristics	TECH	Внутренние ограничения системы — результат анализа технологий, платформ разработки и инструментальных средств для разработки продукта. Анализ проводится системным архитектором
Требования сертифицирующих организаций — Certification Requirements	SERT	Выявленные требования сертифицирующих организаций, которым должна отвечать система
Стандарты и ГОСТы — State Standarts	STSTD	Выявленные в ходе первичного и бизнес-анализа требования ГОСТов, которым должна отвечать система. В эту категорию не попадают архитектурные ГОСТы и RFC
Характеристики аналогичных/наследуемых систем — System Characteristics	CHAR	Выявленные полезные свойства/характеристики систем, которые должна унаследовать система
Концепция создания и развития продукта — Business Vision	BVISION	Выявленное видение развития/создания продукта с точки зрения бизнеса, ожиданий ЗЛ, требований к стандартизации и полезных/ценных для пользователей характеристик сторонних систем
Концепция системы — Technical Vision	TVISION	Техническая концепция реализации выявленного бизнес-видения с учетом выявленных ограничений с точки зрения бизнеса и ожиданий ЗЛ, современных технологий, платформ разработки и инструментальных средств разработки

Название требования	Аббревиатура	Краткое описание/назначение
Пользовательские требования – User Requirements	UREQ	Требования к системе, сформулированные непосредственно ее будущими пользователями
Варианты использования – Use Cases	UC	Вариант использования системы каким-либо актором: набор последовательных действий, приносящих актору – инициатору выполнения варианта использования – значимый и ощутимый результат
Функциональные требования – Functional Requirements	FR	Тестируемые утверждения, отвечающие на вопрос «Что может делать система?» и описывающие функциональность системы в стиле «Система должна делать то-то и то-то...»
Нефункциональные требования – Non-functional Requirements	NFR	Тестируемые утверждения, часто содержащие количественные показатели, которым должна удовлетворять система после реализации ее требуемого поведения

На рис. 31 показана диаграмма **основных** типов требований в проекте.

## Типизация нефункциональных требований

Часть нефункциональных требований, которые находятся в зоне ответственности разных ролей и которые рекомендуется выявлять во всех проектах, идентифицируются отдельным типом.

Название требования	Аббревиатура
Требования к пользовательскому интерфейсу	GUI
Требования к взаимодействию с внешними системами	ICE (Interface Control Element)
Требования к документации	DOC
Требования к данным	DATA
Требования к сертификации	CERT
Требования к окружению	NFR, environment
Требования к аппаратно-техническим средствам	NFR, hardware
Требования к программным средствам	NFR, software
Требования к развертыванию системы	NFR, deployment
Требования к производительности	NFR, performance

Продолжение ⇨

(Продолжение)

Название требования	Аббревиатура
Требования к надежности	NFR, reliability
Требования к масштабируемости	NFR, scalability
Требования к безопасности	NFR, security
Требования к эргономике	NFR, ergonomics
Требования к лицензированию	NFR, licensing
Требования к совместимости	NFR, compatibility
Требования к поддержке	NFR, support

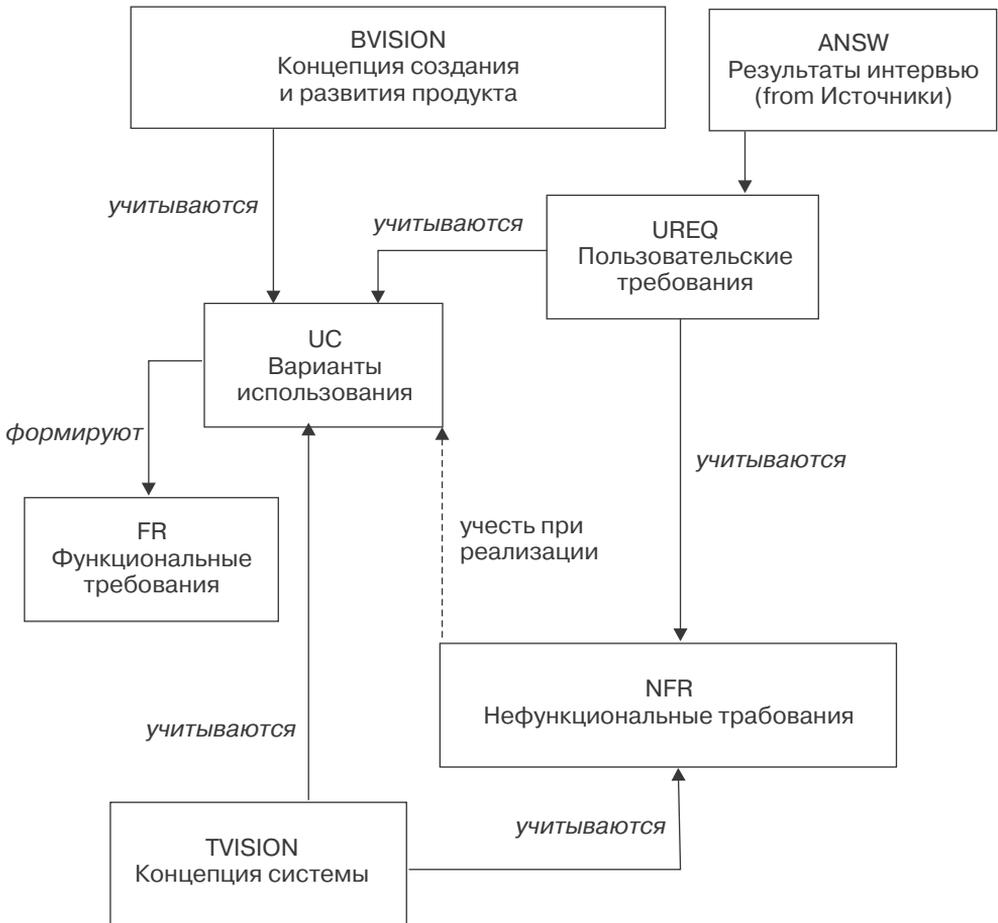


Рис. 31. Основные типы требований в проекте

На рис. 32 приведена диаграмма видов (типов) нефункциональных требований.

## Нефункциональные требования

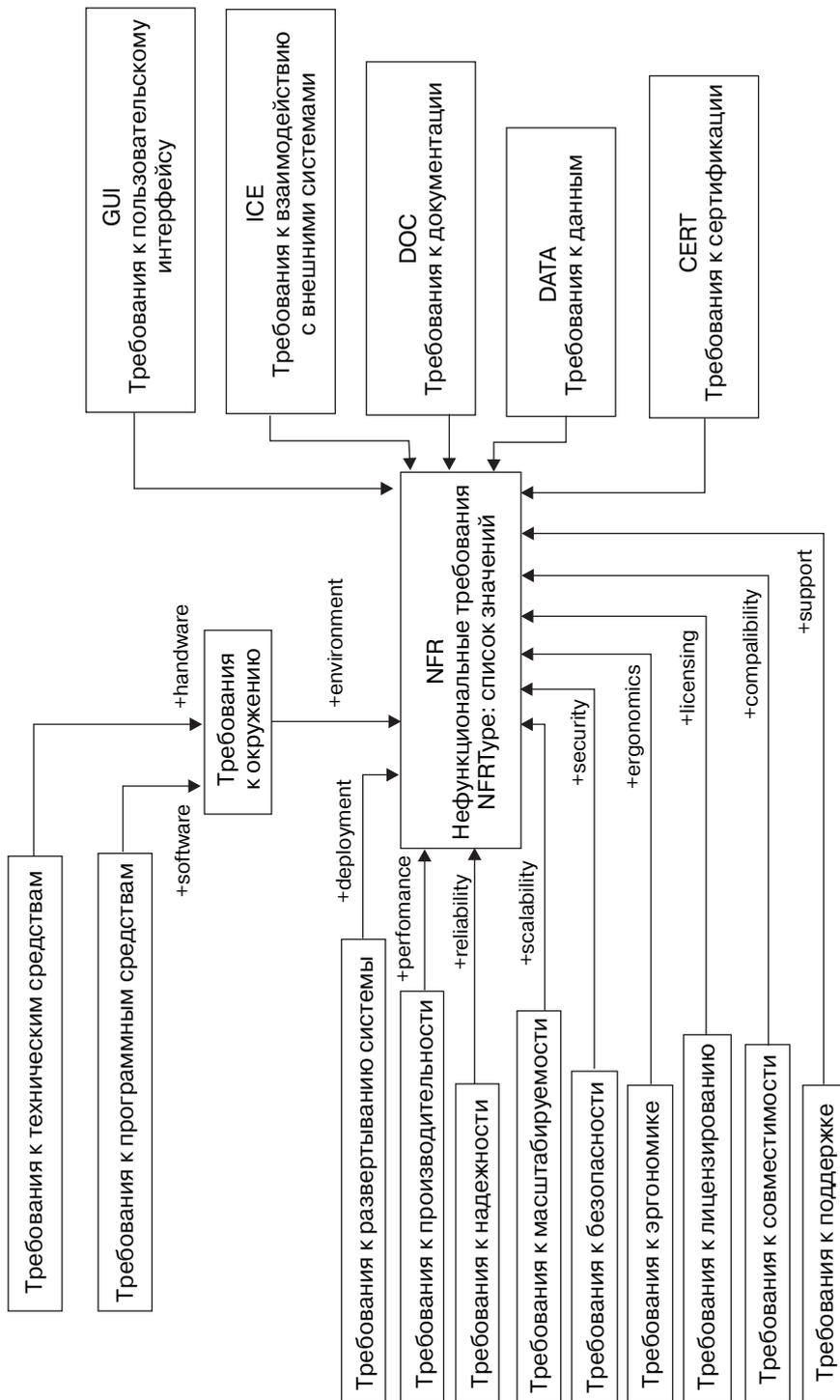


Рис. 32. Виды нефункциональных требований

## Атрибуты требования

Каждый из рассмотренных выше типов требований имеет одинаковый набор атрибутов.

Название атрибута	Назначение	Возможные значения атрибута
<b>NFRType</b>	У той части NFR требований, у которых не указаны аббревиатуры на рисунке выше, тип требования задается значением этого атрибута. Как отдельный тип требования такие требования не выделяются	См. рисунок выше: подписи вида +support
<b>Priority</b>	Приоритет в уточнении, реализации и тестирование требования. Значение атрибута задается менеджером проекта	<p>Hot (срочно) – требование необходимо реализовать срочно, даже ранее, чем требования с высоким приоритетом. Приоритет используется в основном для принятия управленческих решений. Введен для возможности управления очередностью проработки, реализации и тестирования требований, чтобы требования с приоритетами «средний» и «низкий» можно было поднять в очереди над требованиями с приоритетом «высокий».</p> <p>High (высокий) – требование с максимальным приоритетом. Обычно изначально приоритизируется совместно с заказчиком.</p> <p>Medium (средний) – требование среднего приоритета. Обычно изначально приоритизируется совместно с заказчиком. Значение по умолчанию.</p> <p>Low (низкий) – требование низкого приоритета. Обычно изначально приоритизируется совместно с заказчиком</p>
<b>Complexity</b>	Сложность реализации требования (в сложности реализации требования не учитывается сложность тестирования, например подготовка тестовых стендов и/или тестовых данных)	<p>Challenge – требование архитектурно значимое, имеет серьезные технологические риски в реализации. Либо необходимо провести технологические исследования, связанные с этим требованием, что является испытанием для проектной команды. Проставляется менеджером проекта по решению ССВ проекта</p>

Название атрибута	Назначение	Возможные значения атрибута
	<p>Значение атрибута определяется совместно членами проектной команды. Задается менеджером проекта после согласования с проектной командой. Используется менеджером проекта для определения рисков</p>	<p>Solution – требование уровня «решение» – архитектурно значимое, относится к сути разрабатываемого решения. Проставляется менеджером проекта совместно с архитектором.</p> <p>Normal (обычное с точки зрения реализации) – требование, не требующее архитектурных решений. Проставляется менеджером проекта совместно с архитектором. Значение по умолчанию.</p> <p>Easy – требование уровня быстрых изменений, которое может, но не должно быть проверено матрицами трассировок. Проставляется менеджером проекта совместно с разработчиком и архитектором</p>
<b>Cost</b>	<p>Суммарная оценка трудоемкости полной реализации требования в человеко-часах. Значение атрибута задается менеджером проекта после формирования иерархии требований при уточнении оценки трудоемкости проекта. Этот атрибут изменяется также в процессе оценки трудоемкости при Impact Analysis.</p> <p>Задается для пользовательских требований (UREQ), вариантов использования (UC), функциональных требований (FR) и нефункциональных требований (NFR) следующих подтипов:</p> <ul style="list-style-type: none"> <li>• безопасность;</li> <li>• совместимость;</li> <li>• надежность;</li> <li>• производительность;</li> <li>• взаимодействие с внешними системами;</li> <li>• требования к пользовательскому интерфейсу (GUI);</li> <li>• требования к взаимодействию с внешними системами (ICE);</li> </ul>	<p>Минимальное значение атрибута не может быть менее 1 часа</p>

Продолжение ⇨

(Продолжение)

Название атрибута	Назначение	Возможные значения атрибута
	<ul style="list-style-type: none"> <li>• требования к документации (DOC);</li> <li>• требования к данным (DATA);</li> <li>• требования к сертификации (CERT)</li> </ul>	
<b>Assigned To</b>	Участник (участники) проекта, которому назначена работа над требованием. Значение атрибута задается менеджером проекта. Значение данного атрибута должно обязательно анализироваться и, возможно, изменяться при изменении состояния требования	Участник обязательно должен быть в составе проектной команды
<b>Status</b>	Статус требования. Значение атрибута определяется участником проектной команды, которому назначена работа над этим требованием (поле Assigned To)	Описание жизненного цикла каждого типа требования приводится в разделе «Общие принципы управления требованиями»
<b>Target Release</b>	Номер версии продукта, где требование будет реализовано. Значение атрибута задается менеджером проекта	Значение Target Release отражает реальный номер версии, в которой требование планируется реализовать, например «1.0.0»
<b>Phase</b>	Фаза проекта, в которой реализуется требование. Разработка определенной версии продукта может выполняться в нескольких фазах	Например, фаза 1 – прототип, фаза 2 – выпуск первой коммерческой версии продукта. Target Release этой коммерческой версии продукта будет 1.0.0
<b>Cause</b>	Текстовое поле для указания запросов на изменение (Enhancement Requests), номеров запросов на разработку функциональности (Feature Request), номеров протоколов совещаний или просто для указания комментариев по разъяснению перевода требования из одного состояния в другое	Свободный текст длиной не более 255 символов

Состав атрибутов требований может быть дополнен или сокращен для реализации возможности принимать управленческие и технические решения в каждом конкретном проекте.

## 4.2.5. Общие принципы управления требованиями

Прежде чем мы перейдем к рассмотрению этого раздела ПУТ, приведу несколько общих принципов управления требованиями.



Обязательно поднимайте вопрос о проведении однорангового ревью аналитических артефактов (моделей, требований, спецификаций), разрабатываемых в проекте. Одноранговое ревью — это изучение результатов работы специалиста (в данном контексте — аналитика) другим специалистом той же специализации, то есть, например, одноранговое ревью функциональных требований — это процесс изучения одним аналитиком функциональных требований, сформированных другим аналитиком. Причем ревьюер может не быть членом проектной команды, в которую входит автор этих функциональных требований. Это мера обеспечения качества требований, и крайне важно выполнять ее до согласований с другими проектными ролями, поскольку она позволяет исправить ошибки в требованиях на раннем этапе и не тратить время других специалистов на изучение некачественных результатов. Я настойчиво рекомендую проводить эту процедуру во всех проектах.

Не следует строить взаимодействие с архитектором в ключе «как надо реализовать систему». После разработки требований архитектору следует показать, какую функциональность предстоит изменять/создавать, и объяснить почему (пройтись по трассировкам на источники таких требований). Следует всегда совместно с архитектором выделять архитектурно значимые требования и управлять ими.

После прохождения однорангового ревью требования должны быть согласованы сначала с архитектором системы, а затем с тест-менеджером. Это также мера обеспечения качества требований, которая позволяет убедиться в реализуемости и достаточности созданных требований для реализации, с одной стороны, и в том, что получившийся программный продукт может быть протестирован на основании сформированных требований — с другой.

Теперь рассмотрим другие аспекты управления требованиями.

### Управление атрибутами требований

Название атрибута	Ответственный за изменение значения атрибута
Priority	Менеджер проекта
Complexity	Менеджер проекта
Cost	Менеджер проекта

Продолжение ↗

(Продолжение)

Название атрибута	Ответственный за изменение значения атрибута
Assigned To	Менеджер проекта
Status	Для данного атрибута существует своя машина состояний, где указаны ответственные за каждое состояние требования (см. «Описание состояний жизненного цикла требований» в шаблоне ПУТ). Крайне упрощенная машина состояний требования приведена в разделе «Управлять состояниями требований» в главе «Младший аналитик»
Target Release	Менеджер проекта
Phase	Менеджер проекта
Cause	Системный аналитик, участник проектной команды, переводящий требование в какой-то новый статус

## Согласование требований

Согласование требований проектной командой осуществляется через изменение статусов требований согласно машине состояний (см. «Описание состояний жизненного цикла требований» в шаблоне ПУТ).

Требование не может считаться согласованным без подтверждения его тестируемости тест-менеджером проектной команды.

Согласование требований с заказчиком проводится на уровне спецификаций (документов с требованиями).

Спецификация требований считается согласованной, если ее согласовали все участники согласования данного документа.

Все требования, которые вошли в согласованный документ с требованиями, считаются согласованными, за исключением тех требований, которые в процессе согласования были исключены из списка согласования.

## Утверждение требований

Утверждение требований проектной командой осуществляется через изменение статусов требований согласно машине состояний (см. диаграмму статусов для конкретного типа требования в разделе 3.2.10).

Утверждение требований с заказчиком реализуется на уровне спецификаций требований.

Спецификация требований считается утвержденной, если его утвердили все лица, ответственные за утверждение данного документа.

Все требования, которые вошли в утвержденный документ с требованиями, считаются утвержденными, за исключением тех требований, которые в процессе утверждения были исключены из списка утверждения.

Хочу напомнить, что все перечисленные разделы ПУТ приведены с иллюстративными целями. Вы можете и должны адаптировать эти разделы к каждому проекту, в котором принимаете участие, исходя из проектной специфики и проектных ограничений. Так, например, если заказчик свободно ориентируется в вашем инструменте управления требованиями, то согласование и утверждение требований в виде спецификаций требований никому не нужны и в соответствующих разделах ПУТ должно появиться другое описание процесса согласования и утверждения требований.

## 4.2.6. Моделирование

В 1994 году Гради Буч и Джеймс Рамбо, работавшие в компании Rational Software, объединили свои усилия для создания нового языка объектно-ориентированного моделирования. За основу ими были взяты методы моделирования, разработанные Бучем (Booch) и Рамбо (Object Modeling Technique — OMT). OMT был ориентирован на анализ, а разработки Буча и Рамбо — на проектирование программных систем. В октябре 1995 года была выпущена предварительная версия 0.8 унифицированного метода (англ. Unified Method). Осенью 1995 года к компании Rational присоединился Айвар Якобсон, автор метода Object-Oriented Software Engineering — OOSE. Данный метод обеспечивал превосходные возможности для спецификации бизнес-процессов и анализа требований при помощи сценариев использования. OOSE был также интегрирован в унифицированный метод, в основе которого лежит универсальный язык моделирования — Unified Modeling Language (UML) — язык графического описания для объектного моделирования в области разработки программного обеспечения.

На этом этапе основная роль в организации процесса разработки UML перешла к консорциуму OMG (Object Management Group). Группа разработчиков в OMG, в которую также входили Буч, Рамбо и Якобсон, выпустила спецификации UML версий 0.9 и 0.91 в июне и октябре 1996 года.

На волне растущего интереса к UML к разработке новых версий языка в рамках консорциума UML Partners присоединились такие компании, как Digital Equipment Corporation, Hewlett-Packard, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle Corporation, Rational Software, Texas Instruments и Unisys. Результатом совместной работы стала спецификация UML 1.0, вышедшая в январе 1997 года. В ноябре того же года последовала версия 1.1, содержащая улучшения нотации, а также некоторые расширения семантики.

Формальная спецификация последней версии UML 2.0 опубликована в августе 2005 года. Семантика языка была значительно уточнена и расширена для поддержки методологии Model Driven Development (MDD).

Версия UML 1.4.2 принята в качестве международного стандарта ISO/IEC 19501:2005.

Эта информация взята с сайта <http://ru.wikipedia.org/wiki/UML>. Там же можно найти полезные ссылки по UML. Одна из ссылок приведет вас на сайт <http://www.umljokes.com/> – рекомендую уделить время данному ресурсу. Это поможет взглянуть на UML с неожиданной точки зрения.

В основу рассматриваемой в книге методологии легли две методологии разработки ПО – RUP [5] и Iconix [21]. В подразделе «Понимать методологии разработки ПО» раздела «Профессиональные и специальные навыки» главы «Младший аналитик» вы можете ознакомиться с кратким обзором этих методологий.

На рис. 33 показано использование моделирования в различных дисциплинах методологии RUP.

Из методологии RUP [5] были заимствованы дисциплины «Бизнес-моделирование», «Требования», а также роли процесса разработки ПО: «Системный аналитик», «Системный архитектор», «Архитектор приложений» (далее – «Архитектор»), «Бизнес-аналитик», включая их основные деятельности.

Основное преимущество методологии RUP – ее унифицированность, которая проверена на многих сложных проектах и признана мировым сообществом.

Недостатки данной методологии для большинства компаний – в ее тяжеловесности. В RUP все слишком «идеально». Методология включает в себя чересчур много ролей, которые в ходе своей деятельности создают множество проектных артефактов. Такая ситуация приводит к тому, что для реализации проекта по методологии RUP требуется масса ресурсов. Это оправданно в случае создания крупных продуктов, которые будут развиваться, причем не обязательно той же командой, которая разрабатывала программный продукт.

Вторая методология, которая была взята за основу при разработке предлагаемого подхода в системном анализе, – Iconix. На рис. 34 (<http://iconixprocess.com/?s=architect>) показано использование моделирования в Iconix [21].

Основные принципы процесса Iconix:

- ◆ движемся внутри, отталкиваясь от требований пользователя;
- ◆ движемся наружу, отталкиваясь от основных абстракций предметной области;
- ◆ спускаемся вниз – от высокоуровневых моделей к детальному проекту.

Процесс Iconix полезен для ответа на следующие вопросы о системе:

- ◆ каковы пользователи системы (акторы) и что они пытаются сделать;
- ◆ что такое объекты «реального мира» (предметной области) и какие между ними ассоциации;
- ◆ какие объекты участвуют в каждом прецеденте;

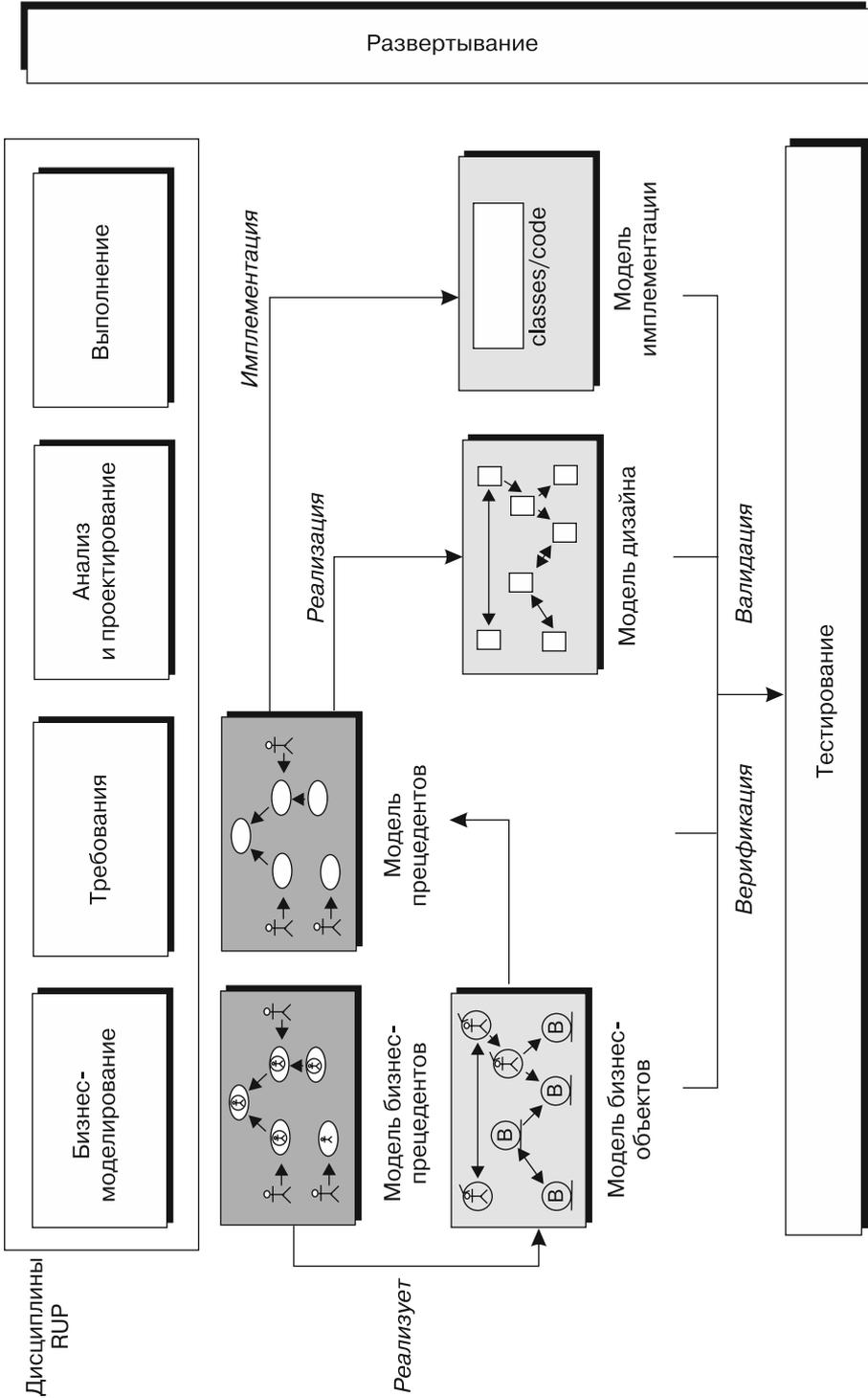


Рис. 33. Использование моделирования в различных дисциплинах методологии RUP

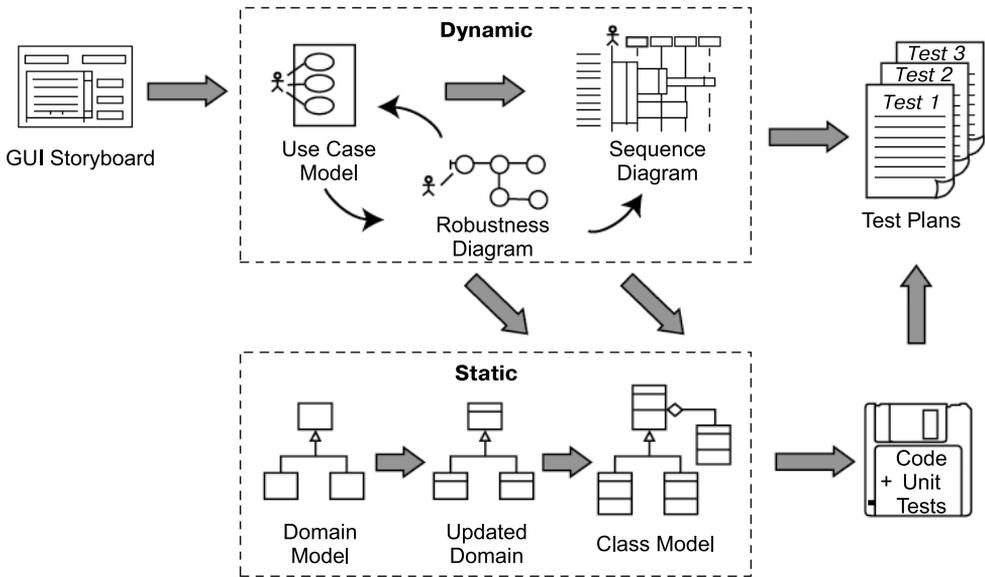


Рис. 34. Использование моделирования в Iconix

- ◆ как учесть ограничения, налагаемые режимом реального времени;
- ◆ как будет выглядеть система на самом нижнем уровне.

Разработка по Iconix — это разработка, управляемая моделями. Предполагается, что для создания работающего кода не обязательно создавать требования и их спецификации. В Iconix создаются UML-модели, и на их основе генерируется исполняемый код.

Основные преимущества этой методологии в том, что она обеспечивает плавный и понятный переход от «что должна делать система» к тому, «как она будет это делать». Кроме того, в ней реализован интерфейсно-ориентированный подход, позволяющий вовлечь заказчика в разработку системы на ранних этапах для снятия рисков разработки ПО, которое не будет соответствовать ожиданиям заказчика. Это Agile методология, девиз которой — Agility without Extreme.

Как и все Agile-методологии, методология Iconix предъявляет повышенные требования к квалификации проектной команды — умение читать и разбираться в моделях UML. И в этом заключается ее недостаток. Не всегда можно сформировать такую проектную команду.

Из данной методологии были взяты практики применения диаграмм устойчивости (пригодности) для анализа сценариев вариантов использования и активного использования моделирования.

Прежде чем перейти к предлагаемой мной методологии моделирования в системном анализе, хочу напомнить, что задача этой книги — не научить вас строить эти модели, а показать их место в методологии.



Итак, давайте рассмотрим мои предложения по использованию моделей в системном анализе. На рис. 35 представлены модели и последовательность их разработки, а также комментарии касательно основной пользы от создания этих моделей.

Как и в Icopix, область моделирования условно разделена на статическую и динамическую составляющие. Первая дает представление о составе системы, вторая описывает поведение системы, ее функции и в конечном итоге поведение классов, реализующих систему. Некоторые модели находятся на стыке этих областей.



Прежде чем продолжить чтение, рекомендую вам ознакомиться со статьей «Пример описания предметной области с использованием UML при разработке программных систем» Е. Б. Золотухиной, Р. В. Алфимова, которую вы можете найти на сайте [www.interface.ru](http://www.interface.ru).

Теперь подробнее рассмотрим модели предметной области на примере моделирования системы обмена файлами.

Система представляет собой сервис обмена файлами в корпоративной сети. Основная задача — обеспечить гарантированную доставку сообщений (файлов) в гибридной сетевой среде. Важной особенностью системы является обеспечение функций защищенной/безопасной доставки. Система обладает развитым веб-интерфейсом и позволяет интегрироваться с сервисами аутентификации/авторизации, а также предоставляет функции экспорта и выгрузки файлов из системы документооборота компании.

## Бизнес-модель

Описывает основные бизнес-сценарии с точки зрения взаимодействия бизнес-акторов и бизнеса компании. Бизнес-акторы по аналогии с актерами — внешние по отношению к бизнесу люди, организации, другие бизнесы, которые получают значимый и осязаемый результат от бизнеса рассматриваемой компании. Выявляются также бизнес-работники, которые являются сотрудниками или ролями в рассматриваемом бизнесе, — внутренние роли бизнеса. Бизнес-сценарий — сценарий взаимодействия бизнеса и бизнес-актора либо бизнес-работника, реализуемый для достижения целей бизнеса.

Источник	Назначение	Содержание	Используется
Результаты интервью ЗЛ	Выявление, классификация и формализация сведений о целях бизнеса, бизнес-сценариях, бизнес-актерах и бизнес-работниках	Бизнес-сценарии, сгруппированные по бизнес-актерам, бизнес-работникам и целям бизнеса	Используется как источник информации для построения модели предметной области и для выявления бизнес-требований и бизнес-правил

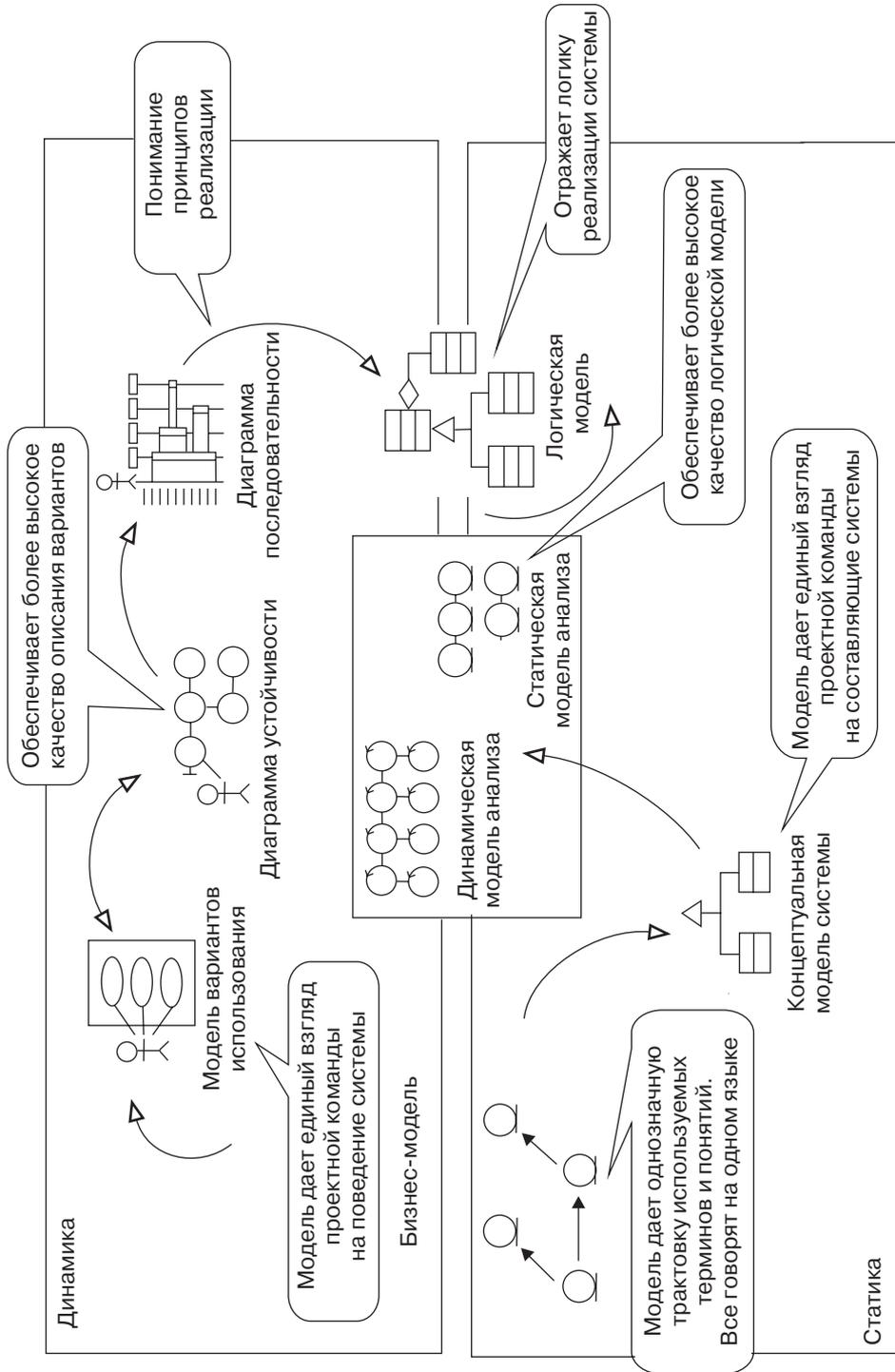


Рис. 35. Модели и последовательность их разработки

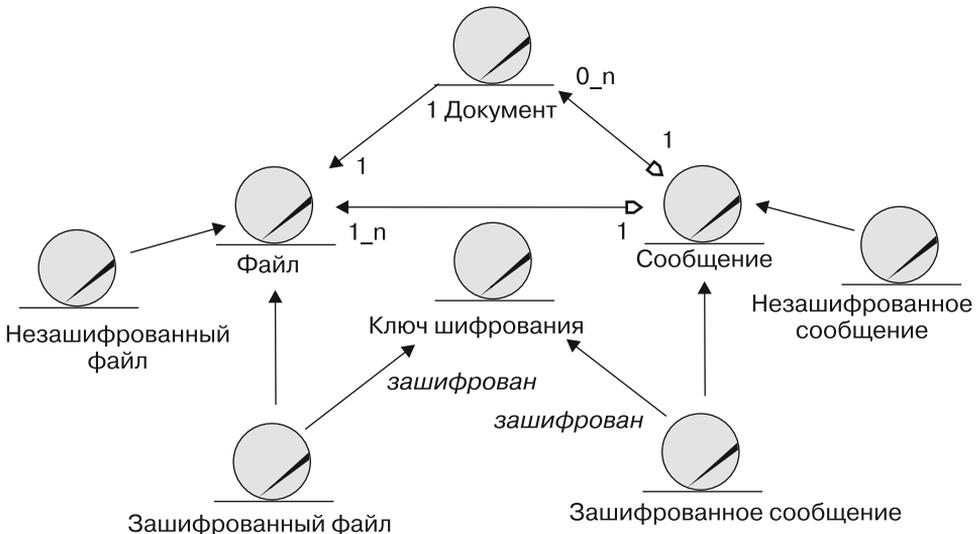
Бизнес-модель можно строить с использованием не только концепции бизнес-сценариев, но и инотаций IDEF0, eEPC и др. В рассматриваемом примере бизнес-модель достаточно простая, и я предлагаю вам построить ее самостоятельно.

## Модель предметной области

Описывает основные сущности предметной области и взаимосвязи между ними.

Источник	Назначение	Содержание	Используется
Результаты интервью ЗЛ, бизнес-модель	Выявление, классификация и формализация сведений обо всех аспектах предметной области, определяющих свойства разрабатываемой системы	Бизнес-сущности и их взаимосвязи	Используется как источник информации для создания концептуальной модели системы

Пример модели предметной области для системы обмена файлами показан на рис. 36.



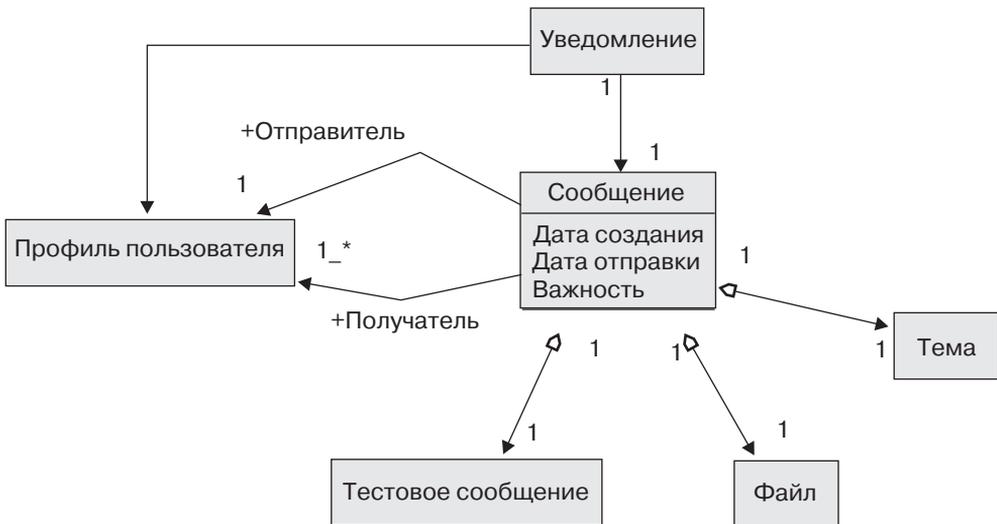
**Рис. 36.** Пример диаграммы модели предметной области

## Концептуальная модель системы

Описывает основные сущности системы и взаимосвязи между ними.

Источник	Назначение	Содержание	Используется
Модель предметной области	Выявление, классификация и формализация сведений о разрабатываемой системе	Содержит основные (с точки зрения аналитика) кандидаты в классы системы и связи между ними, реализующие бизнес-сущности модели предметной области и их основные атрибуты, а также дополнительные классы системы, реализующие очевидные общесистемные походы (например, класс «профиль пользователя»). <i>Не содержит методов классов, типов атрибутов</i>	Используется для унифицированного понимания концепции создаваемой системы всей проектной командой. Кандидаты в классы этой модели будут использованы впоследствии при построении диаграмм пригодности и последовательности

Эта модель для системы обмена файлами показана на рис. 37.



**Рис. 37.** Пример диаграммы концептуальной модели системы

В классе «Сообщение» есть некоторые атрибуты, что противоречит определению. Однако это нормальная практика. Выделяя их, аналитик хотел «сказать» архитектору, что это основные и крайне важные атрибуты не с точки зрения реализации системы, а с точки зрения ее концепции.

## Модель вариантов использования (Use Case Model)

Описывает варианты использования системы со стороны акторов и дает единый взгляд проектной команды на поведение системы. Применяется для функциональной декомпозиции (об этом мы говорили в подразделе «Выделять подсистемы и определять их функции» раздела «Профессиональные и специальные навыки» главы «Младший аналитик»):

Источник	Назначение	Содержание	Используется
Результаты интервью, анкетирования и опросов ЗЛ, проведенных с целью выявить желаемое поведение разрабатываемой системы	Модель вариантов использования показывает, как пользователи взаимодействуют с системой и что система делает в ответ на эти взаимодействия	Содержит все варианты использования системы в структурированном виде, отношения между ними, точки взаимодействия пользователя и системы, краткое описание вариантов использования	Используется для формирования унифицированного взгляда на систему в целом с точки зрения ее использования акторами; для выявления функциональных подсистем; для построения диаграмм пригодности, последовательности и View of Participated Classes

Эта модель для системы обмена файлами показана на рис. 38.

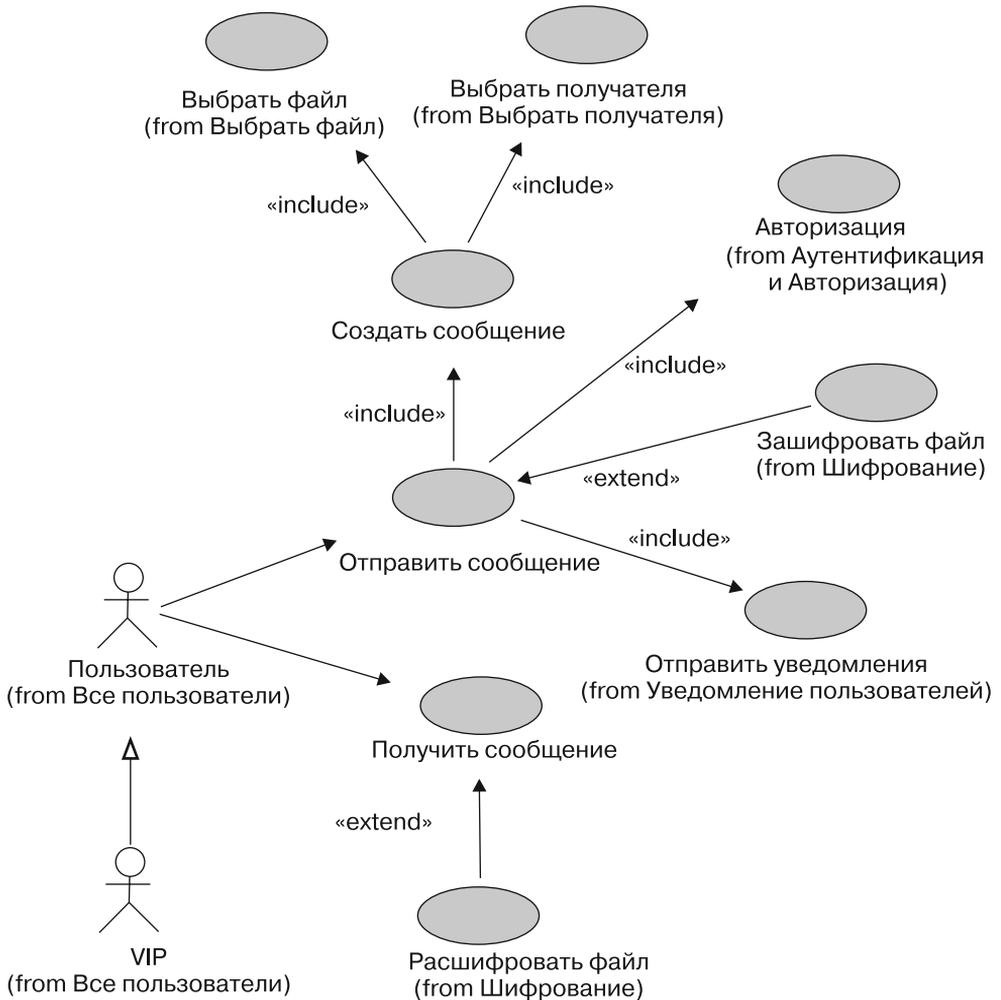
Хочу обратить ваше внимание, что при разработке Use Case-модели функциональные требования в общепринятом понимании (в виде положительных утверждений «что должна делать система») можно не выявлять, потому что все варианты использования по своей сути есть функции системы, которые реализуют функциональные требования. И если мы имеем возможность определиться с реализацией функциональных требований сразу, то и тратить время на выявление функциональных требований не обязательно.

Я рекомендую гибридный вариант — определять только высокоуровневые функциональные требования к системе в виде иерархии требований не выше третьего уровня вложенности, а детальную проработку функциональности обеспечивать за счет Use Case-моделирования.

После описания вариантов использования (ВИ) строятся диаграммы пригодности/устойчивости.

## Модель анализа пригодности варианта использования

Анализ пригодности варианта использования проводится через построение диаграммы пригодности/устойчивости (Robustness Diagram).



**Рис. 38.** Пример модели вариантов использования

Основная задача диаграммы пригодности/устойчивости – уточнить описание вариантов использования. Именно поэтому стрелка между моделью вариантов использования и диаграммами устойчивости на рис. 39 двунаправленная. Вообще проверять качество написания вариантов использования можно, либо читая и анализируя их, либо строя диаграммы устойчивости. Для некачественно описанных вариантов использования построить диаграммы устойчивости в соответствии с правилами их построения невозможно. Подобная проверка – достаточно трудоемкая работа, поэтому обычно ее выполняют только для наиболее важных вариантов использования при условии согласования проведения таких работ менеджером проекта (например, архитектурно значимых или критических с точки зрения поведения системы).

Данная практика заимствована из методологии Iconix [21], я рекомендую вам обратиться к первоисточнику, чтобы детально разобраться с этой диаграммой.

Источник	Назначение	Содержание	Используется
Описание варианта использования. Сущности из концептуальной модели системы	Служит связующим звеном между описанием варианта использования и аналитическими моделями	Содержит интерфейсы, контролеры и сущности, необходимые для выполнения всей функциональности варианта использования	Используется для выявления «пропущенных» (не выявленных ранее) классов и кандидатов в классы



Давайте рассмотрим небольшой пример. Предположим, есть описание варианта использования «Отправить сообщение».

### Цель

*Отправить сообщение.*

### Основной поток

*Пользователь инициирует создание нового сообщения. Система отображает форму «Отправить сообщение». Пользователь вводит текст сообщения. Пользователь инициирует отправку сообщения. Система отправляет сообщение.*

Пытаемся построить диаграмму пригодности по этому сценарию. Происходит это так.

*Пользователь* — «рисуем» пользователя. *Иницирует создание нового сообщения. Система отображает форму «Отправить сообщение».* «Рисуем» элемент типа интерфейс «Форма “Отправить сообщение”» и показываем стрелкой инициацию.

*Пользователь вводит текст сообщения.* Предполагается, что пользователь вводит текст в форме «Отправить сообщение»? Да, так и есть. Продолжаем анализ.

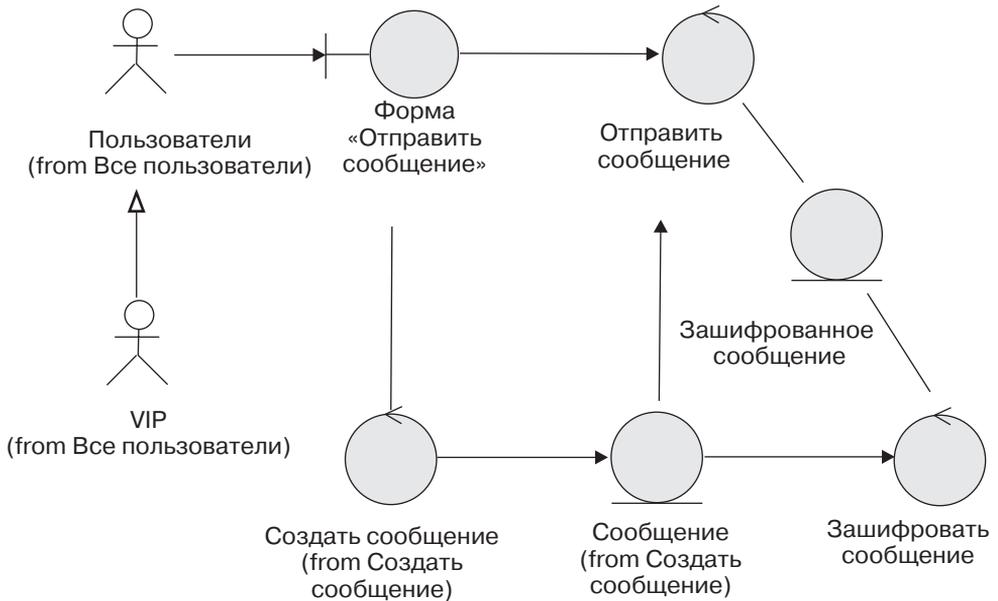
*Пользователь инициирует отправку сообщения. Система отправляет сообщение.*

Стоп. Ошибка. А когда, собственно, сообщение было создано? Какой элемент надо «нарисовать» на диаграмме пригодности, чтобы проиллюстрировать данное действие? В текущем варианте сценария варианта использования это невозможно. Исправляем вариант использования. Теперь этот шаг выглядит следующим образом: *пользователь инициирует отправку сообщения. Система создает и отправляет сообщение.*

Стоп! Но ведь мы разрабатываем систему, у которой есть требование: «Существенной особенностью системы является обеспечение функций защищенной/безопасной доставки». Опять ошибка. В конечном варианте этот шаг вновь исправляется и выглядит как: *пользователь инициирует отправку*

сообщения. Система создает сообщение. Система шифрует сообщение. Система отправляет сообщение.

В процессе всех этих рассуждений строилась диаграмма пригодности, показанная на рис. 39.



**Рис. 39.** Модель анализа пригодности. Первый шаг

Как я уже говорил, не все типы элементов могут быть связаны со всеми типами элементов. Правила построения диаграмм пригодности, которые показывают, какие типы элементов могут быть связаны с другими типами, приведены на рис. 40.

Это рисунок из книги Дага Розенберга и Кендалла Скотта *Applying Use Case Driven Object Modeling with UML* [9], которую я настойчиво рекомендую вам изучить. На сайте <http://iconixprocess.com/> вы также сможете найти немало полезной информации.

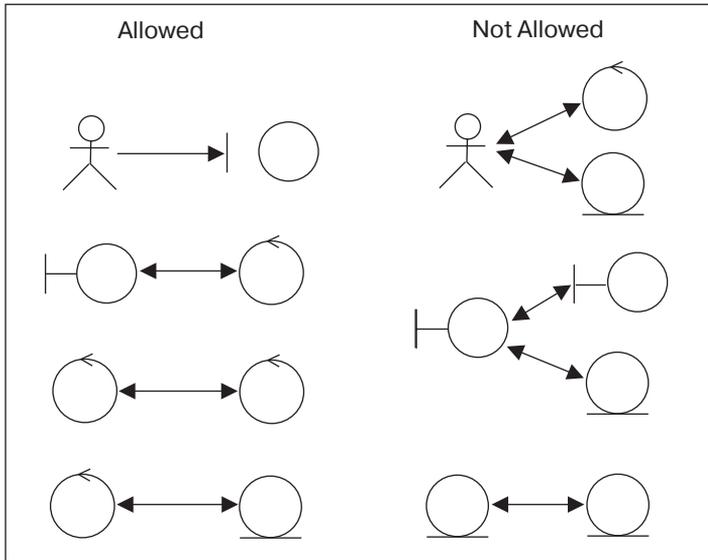
Итак, после проведения анализа пригодности варианта использования «Отправить сообщение» с применением диаграммы пригодности исправленный вариант имеет следующий вид.

### Цель

*Отправить сообщение.*

### Основной поток

*Пользователь инициирует создание нового сообщения. Система отображает форму «Отправить сообщение». Пользователь вводит текст сообщения. Пользователь инициирует отправку сообщения. Система отправляет сообщение.*



**Рис. 40.** Правила построения диаграмм пригодности

*Пользователь инициирует отправку сообщения. Система создает сообщение. Система шифрует сообщение. Система отправляет сообщение.*

Но и это еще не все. Анализируем дальше. Мы знаем, что «система позволяет интегрироваться с существующими сервисами аутентификации/авторизации». Это наталкивает нас на мысль о том, что могут быть ограничения на возможность отправки сообщения разным адресатам. Стоп! А где мы вообще сказали про адресатов? Это ведь целый новый вариант использования — «Выбрать адресатов». Поправляем наш вариант использования. Теперь он выглядит так:

### **Цель**

*Отправить сообщение.*

### **Основной поток**

*Пользователь инициирует создание нового сообщения. Система отображает форму «Отправить сообщение». Пользователь вводит текст сообщения. Пользователь инициирует выбор адресатов сообщения. Система выполняет вариант использования «Выбрать адресатов». Пользователь инициирует отправку сообщения. Система создает сообщение. Система шифрует сообщение. Система отправляет сообщение.*

Для упрощения примера не будем создавать выявленный вариант использования. Но мы только что предотвратили критическую ошибку в требованиях — выявили пропущенный аналитиком вариант использования. Это очень важно.

Однако это еще не все. Мы остановились на проверке прав пользователя. Действительно, прежде, чем отправить сообщение, система должна убедиться, что пользователь имеет такие права (рис. 41).

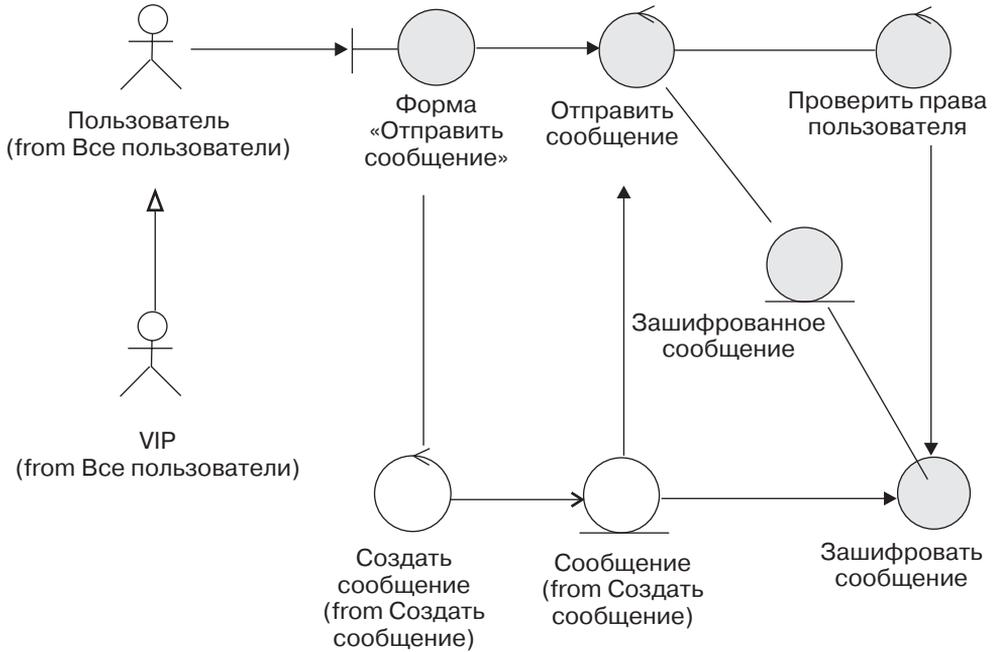


Рис. 41. Модель анализа пригодности. Второй шаг



Финальный откорректированный вариант использования (ВИ) «Отправить сообщение» выглядит так.

Начальный текст ВИ	Финальный текст ВИ
<p><b>Цель</b> Отправить сообщение</p> <p><b>Основной поток</b> Пользователь инициирует создание нового сообщения. Система отображает форму «Отправить сообщение». Пользователь вводит текст сообщения. Пользователь инициирует отправку сообщения. Система отправляет сообщение</p>	<p><b>Цель</b> Отправить сообщение</p> <p><b>Основной поток</b> Пользователь инициирует создание нового сообщения. Система отображает форму «Отправить сообщение». Пользователь вводит текст сообщения. Пользователь инициирует выбор адресатов сообщения. Система выполняет вариант использования «Выбрать адресатов». Пользователь инициирует отправку сообщения. Система создает сообщение. Система проверяет права пользователя на отправку сообщения такого типа указанным адресатам. (A1) Система шифрует сообщение. Система отправляет сообщение.</p> <p><b>Альтернативный поток</b> Пользователь не имеет прав на отправку сообщения такого типа указанным адресатам. Система информирует пользователя об ошибке «Недопустимый тип сообщения для адресатов»</p>

Конечно, рассмотренный пример достаточно абстрактный, но он показывает, какую пользу может принести анализ пригодности варианта использования.

Лучше, когда проверку на пригодность варианта использования выполняет не тот же аналитик, который описывал сценарий варианта использования. Свежий взгляд позволяет увидеть массу мелочей и нюансов, которые могут ускользнуть от «замыленного» взгляда автора.



Даже если менеджер проекта не выделяет время на выполнение таких работ, потренируйтесь для себя. Постройте в свое свободное время такие диаграммы для сложных вариантов использования. Вы поймете, прочувствуете суть этого метода и улучшите свои навыки аналитического мышления. Со временем вы привыкнете так думать, это станет для вас естественным. Такой навык очень важен и для создания качественной логической модели системы.

## Диаграмма активности

Описывает сценарий варианта использования системы в виде алгоритма.

Источник	Назначение	Содержание	Используется
Детализированный сценарий варианта использования системы	Графическое представление последовательности и условий выполнения активностей прецедента	Активность, точки принятия решений, отражает возможность параллельного выполнения активностей	Используется для детального и наглядного описания алгоритма прецедента

Данная модель для системы обмена файлами представлена на рис. 42.

## Диаграмма последовательности

Описывает кандидаты в классы и их взаимодействия в виде обмена сообщениями и вызовами методов друг друга. Строится для объектов — кандидатов в классы системы.

Источник	Назначение	Содержание	Используется
Детализированный сценарий варианта использования системы. Сущности концептуальной модели. Сущности, выявленные в ходе анализа пригодности вариантов использования	Графическое представление взаимодействий между объектами во времени	Взаимодействующие объекты, последовательность операций и тип операции	Используется для выявления поведения классов и формирования методов классов

Эта модель для системы обмена файлами приведена на рис. 43.

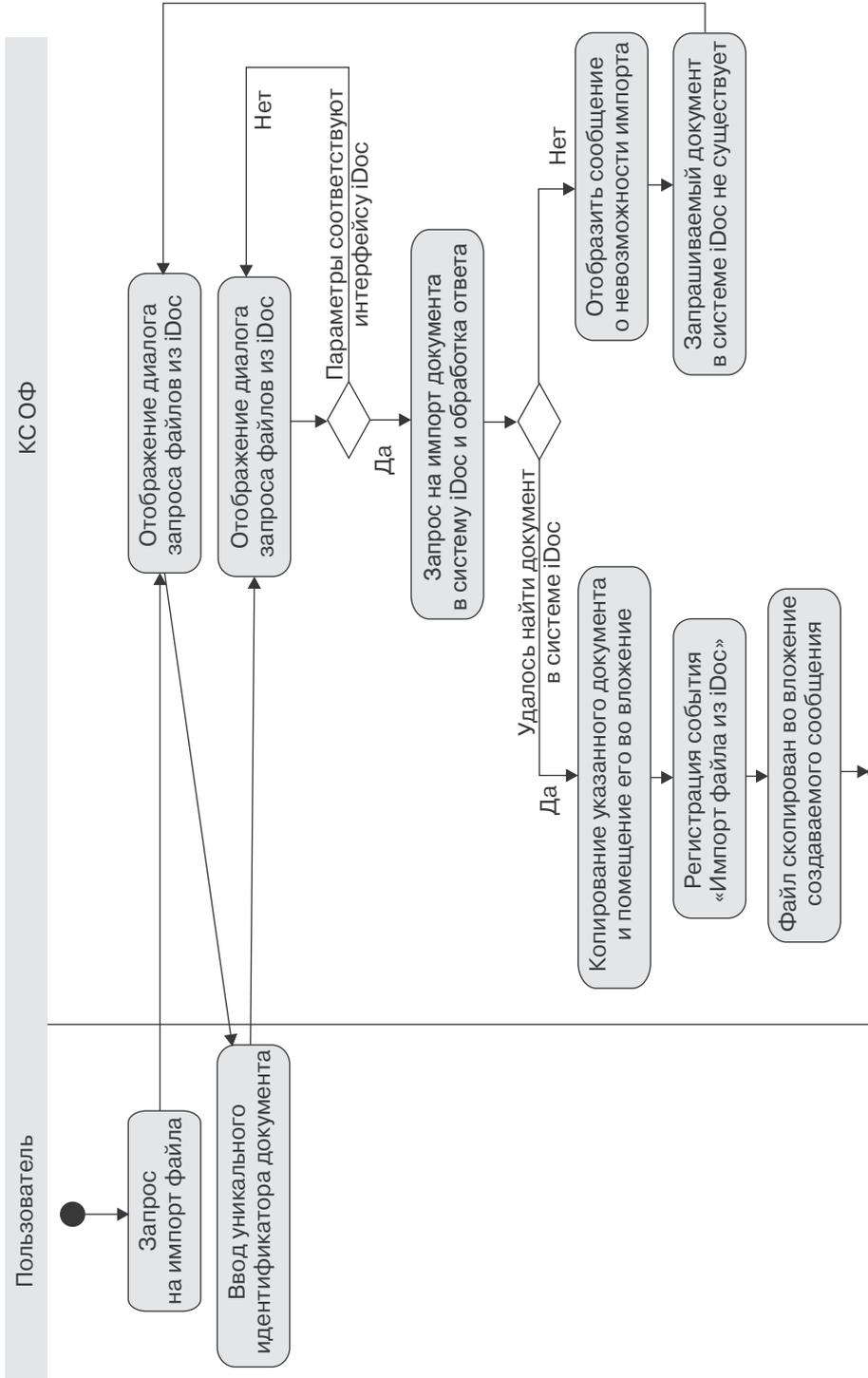


Рис. 42. Диаграмма активности

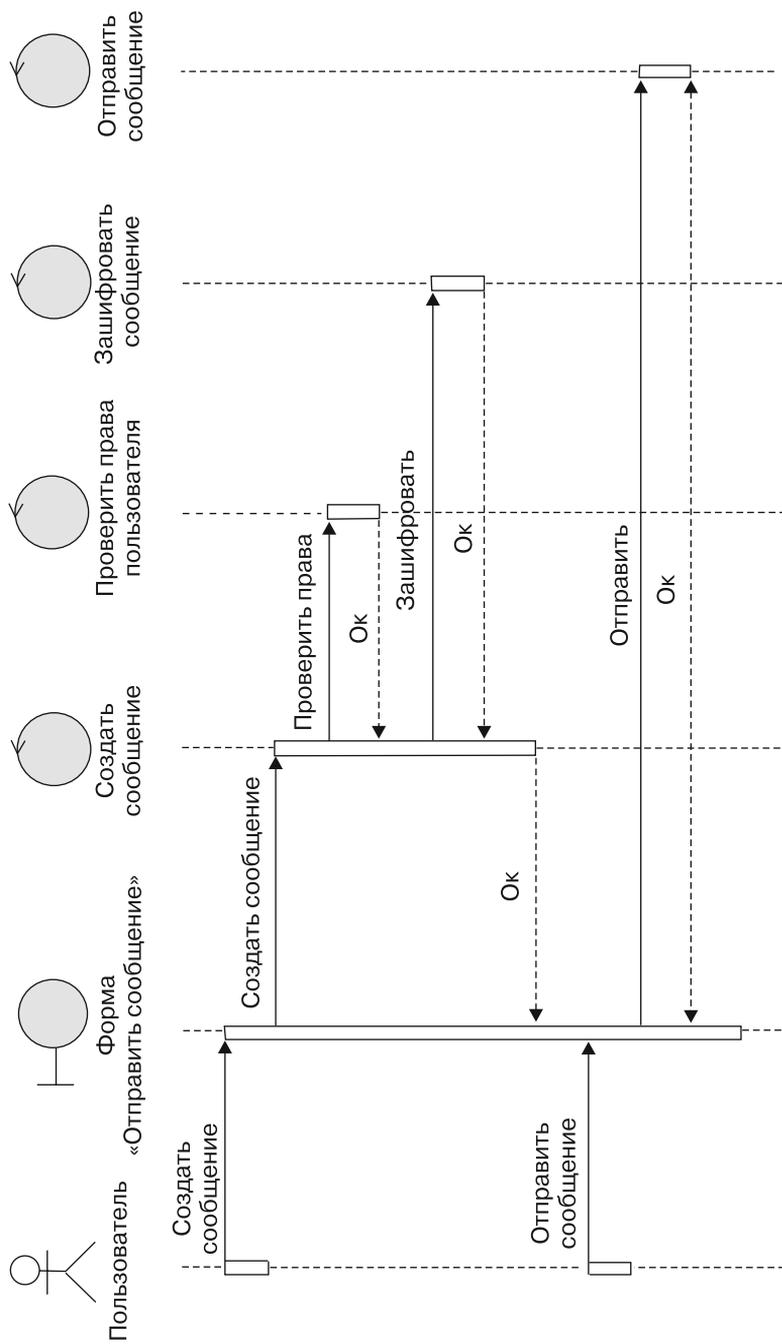


Рис. 43. Диаграмма последовательности

Важно понимать, что на представленных диаграммах отражаются не классы, а кандидаты в классы, и не все из них станут классами. Например, кандидат в класс «Создать сообщение» может быть методом класса «Сообщение», а не отдельным управляющим классом.

## Модель анализа

### Статическая модель анализа

Статическая часть модели нужна для выявления дополнительных взаимосвязей и состава выявленных ранее сущностей, для дальнейшей детализации. Это более пристальный взгляд на состав системы.

Источник	Назначение	Содержание	Используется
Сущности концептуальной модели. Сущности, выявленные в ходе анализа пригодности вариантов использования	Общий взгляд на статическую модель сущностей системы	Основные сущности системы и связи между ними	Используется для анализа установленных при проведении анализа пригодности вариантов использования связей между классами-кандидатами, выявления избыточных и неоптимальных связей, с целью устранения первых и оптимизации вторых

Даже если анализ пригодности вариантов использования не выполнялся, но строилась концептуальная модель и выделялись каким-то образом классы-кандидаты, я рекомендую строить эту модель.

При использовании инструмента **Rational Rose механика построения этой диаграммы** и проведения анализа получившихся сущностей и их связей такова: вы создаете новую диаграмму классов, на которую поочередно с помощью *drag & drop* «натаскиваете» классы-кандидаты из Explorer модели Rational Rose. Связи между классами будут строиться инструментом автоматически. Все, что вам нужно, — анализировать, какие новые связи появились после добавления очередного класса-кандидата, нет ли циклических связей, избыточных связей или просто противоречивых связей. Если вы обнаруживаете какие-то проблемы, то выделяете класс и средствами Rose переходите к списку всех диаграмм, где данный класс участвует. Это можно сделать, нажав **Ctrl+F** после выделения класса и выполнив поиск. Далее вы перемещаетесь между диаграммами и понимаете контекст образования какой-то из двух противоречивых связей и соответственно исправляете одну связь. Естественно, если вы исправляете связь в модели анализа, то следует исправить ее и в модели пригодности, а это автоматически означает, что необходима проверка текста сценария варианта использования. Как видите, с помощью данной модели выполняется не только контроль созданных сущностей и связей между сущностями, но и обратный контроль сценариев вариантов использования.

## Динамическая модель анализа

Динамическая часть модели нужна для выявления дополнительных взаимосвязей и состава выявленных ранее классов типа Control для дальнейшей детализации. Это более пристальный взгляд на поведенческую составляющую системы.

Источник	Назначение	Содержание	Используется
Сущности, выявленные в ходе анализа пригодности вариантов использования	Общий взгляд на динамическую модель сущностей системы	Основные контролеры системы и связи между ними	Используется для анализа установленных при проведении анализа пригодности вариантов использования связей между контролерами с целью выявления избыточных и неоптимальных связей, а также для устранения первых и оптимизации вторых

В результате работы с динамической моделью анализа какие-то классы типа Control могут слиться в один, какие-то могут быть разбиты на несколько. Классы типа Control в будущем станут либо управляющими классами, которые могут не обладать собственными характеристиками (атрибутами), но управляют поведением системы, либо методами других классов, которые непосредственно реализуют поведение системы.

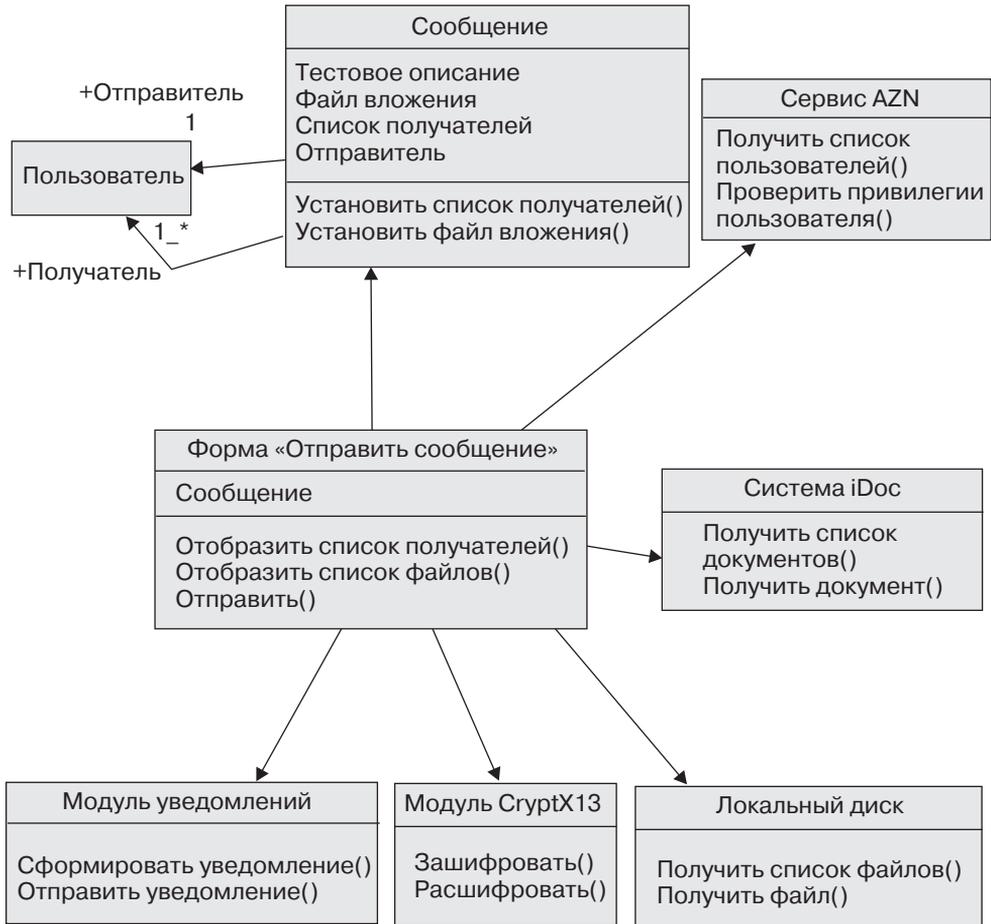
Модель анализа сглаживает переход к логической модели системы. Польза для проектной команды (косвенная) — более качественная логическая модель.

## Логическая модель системы

Отражает логику реализации. Любой специалист, умеющий читать модели UML, может посмотреть на логическую модель, проверить логику реализации системы, высказать свои замечания или пожелания. В основном данная модель представляет интерес для архитектора системы. В идеале эту модель должны строить системный аналитик совместно с системным архитектором. Это последняя модель, которую разрабатывает аналитик.

Источник	Назначение	Содержание	Используется
Детализированные сценарии вариантов использования. Диаграммы последовательности. Модели анализа	Отражение распределения поведения системы по классам. Логическая модель системы — модель классов системы с атрибутами, методами, но без деталей реализации	Содержит классы реализации, их атрибуты и основные методы. Не зависит от выбранного языка, платформы. В этой модели еще не применяются ни архитектурные паттерны, ни паттерны объектно-ориентированного моделирования	Используется для построения модели реализации системы

Эта модель для системы обмена файлами приведена на рис. 44.



**Рис. 44.** Логическая модель системы

Следующая модель в последовательности разработки моделей — модель дизайна. Но ее строит уже архитектор.

Мы рассмотрели последовательность разработки моделей, пройдя весь путь от создания бизнес-модели и модели предметной области до создания логической модели, реализующей основную логику системы. Проанализировали, что представляет собой каждая модель, каково ее назначение, какую пользу проектной команде она приносит и как используется в процессе разработки.

Кроме перечисленных, при моделировании системы также применяются другие модели, которые разрабатываются в основном системным архитектором и системным дизайнером.

## Модель реализации

Источник	Назначение	Содержание	Используется
Логическая модель	<p>Моделирование видения архитектора по реализации системы.</p> <p>Это дальнейшее развитие логической модели системы — модель классов системы с атрибутами, методами, деталями реализации</p>	<p>Полное описание классов системы и детали реализации — используемые паттерны и фреймворки.</p> <p>В данной модели применены архитектурные паттерны и паттерны объектно-ориентированного моделирования</p>	Используется для разработки кода системы

## Модель компонентов

Источник	Назначение	Содержание	Используется
<p>Концепция системы.</p> <p>Логическая модель системы</p>	Моделирование основных компонентов системы и их зависимостей	<p>Описание компонентов и их зависимости.</p> <p>Распределяет функционал системы путем связи каждого класса с конкретным компонентом системы</p>	Используется для разработки модели развертывания, понимания принципов построения системы, ее компонентной декомпозиции и интерфейсов взаимодействия между компонентами

## Модель развертывания

Источник	Назначение	Содержание	Используется
<p>Модель компонентов.</p> <p>Модель реализации</p>	Моделирование распределения компонентов системы между физическими устройствами	<p>Физические устройства, принимающие участие в функционировании системы, и протоколы взаимодействия между ними.</p> <p>Распределяет процессную модель системы по конкретным процессорам (физическим устройствам) путем связи каждого процесса с конкретным устройством</p>	Используется для развертывания системы

## Сводная таблица моделей, используемых в разработке ПО



---

Ниже представлены все рассмотренные модели, цель и источники создания, назначение, содержание и способы их дальнейшего использования (табл. 6).

---

### 4.2.7. Соответствие типов требований и моделей

Области моделирования и разработки требований в системном анализе тесно связаны между собой.

Давайте рассмотрим, как связаны модели и типы требований, которые мы обсуждали в разделе «Концептуальная модель типов требований». Начнем с возможных вариантов использования. По мере того как аналитик строит модель вариантов использования, для каждого варианта создается аналогичное требование в проекте требований — вариант использования (UC). Если вы работаете с инструментами Rational, то эта операция выполняется двумя щелчками кнопкой мыши. Затем можно быстро перемещаться от варианта использования в модели Rational Rose к соответствующему требованию в проекте Requisite PRO и наоборот. Кроме того, благодаря штатной функциональности интеграции инструментов Rational Rose и Rational Requisite PRO, находясь непосредственно в модели, можно просмотреть и изменить атрибуты требования UC.

Теперь, когда в проекте требований появилось соответствующее требование для варианта использования, можно проводить трассировки в соответствии с концептуальной моделью требований (рис. 45).

Приступим к анализу двух важнейших типов требований в разработке системы: требований концепции создания и развития продуктов (BVISION) и требований концепции системы (TVISION).

Концептуальная модель требований концепции создания и развития продуктов (BVISION) показана на рис. 46.

Как видим, для формирования требований этого типа особенно важны бизнес-модель и модель предметной области. В модели бизнеса источниками для выявления бизнес-требований и бизнес-правил являются бизнес-сценарии, а в модели предметной области бизнес-правила выражаются в виде диаграмм сущностей предметной области с их взаимосвязями и важными атрибутами. Собственно, вся модель предметной области является в той или иной степени бизнес-правилами, поскольку описывает, какими сущностями и атрибутами оперирует бизнес.

Таблица 6. Сводная таблица моделей, используемых в разработке ПО

Модель/ диаграмма	Цель	Источник	Назначение	Содержание	Использование
Бизнес- модель	Описание основных бизнес-сценариев с точки зрения взаимодействия бизнес-акторов и бизнеса компании	Результаты интервью ЗП	Выявление, классификация и формализация сведений о целях бизнеса, бизнес-сценариях, бизнес-акторах и бизнес-работниках	Бизнес-сценарии, структурированные по бизнес-акторам, бизнес-работникам и целям бизнеса	Как источник информации для модели предметной области, а также для выявления бизнес-требований и бизнес-правил
Модель пред- метной области	Описание основных сущностей предметной области и взаимосвязей между ними	Результаты интервью ЗП, бизнес-модель	Выявление, классификация и формализация сведений обо всех аспектах предметной области, определяющих свойства разрабатываемой системы	Бизнес-сущности и их взаимосвязи	Как источник информации для концептуальной модели системы
Концеп- туальная модель системы	Описание основных сущностей системы и взаимосвязей между ними	Модель предметной области	Выявление, классификация и формализация сведений о разрабатываемой системе	Основные (с точки зрения аналитика) кандидаты в классы системы и связи между ними, реализующие бизнес-сущности модели предметной области и их основные атрибуты, а также дополнительные классы системы, реализующие очевидные общесистемные подходы (например, класс «профиль пользователя»).	Для унифицированного понимания концепции создаваемой системы всей проектной командой. Кандидаты в классы этой модели впоследствии будут использованы при построении диаграмм пригодности и последовательности

Продолжение ⇨

*Не содержит методов классов, типов атрибутов*

(Продолжение)

Модель/ диаграмма	Цель	Источник	Назначение	Содержание	Использование
Модель вариантов использования системы (Use Case Model)	Описание вариантов использования системы со стороны акторов. Функциональная декомпозиция (см. подраздел «Выделять подсистемы и определять их функции» раздела «Профессиональные и специальные навыки» главы «Младший аналитик»)	Результаты интервью, анкетирования и опросов заинтересованных лиц, проведенных с целью выявить желаемое поведение разрабатываемой системы	Отражение того, как пользователи взаимодействуют с системой и что система делает в ответ на это	Варианты использования системы в структурном виде, отношения между ними, точки взаимодействия пользователя и системы, краткое описание вариантов использования	Формирование унифицированного взгляда на систему в целом с точки зрения ее использования акторами. Выявление функциональных подсистем. Построение диаграмм пригодности, последовательности и View of Participated Classes
Модель анализа пригодности варианта использования	Уточнение описания вариантов использования	Описание варианта использования. Сущности из концептуальной модели системы	Связующее звено между описанием варианта использования и аналитическими моделями	Интерфейсы, контролеры и сущности, необходимые для выполнения всей функциональности варианта использования	Выявление «пропущенных» (не выявленных ранее) классов и кандидатов в классы
Диаграмма активности	Описание сценария варианта использования системы в виде алгоритма	Детализированный сценарий варианта использования системы	Графическое представление последовательности и условий выполнения активностей прецедента	Активности, точки принятия решений, возможность параллельного выполнения активностей	Детальное и наглядное описание алгоритма прецедента

Модель/ диаграмма	Цель	Источник	Назначение	Содержание	Использование
Диаграмма последовательности	Описание кан-дидатов в классы и их взаимо-действия в виде обмена обще-ниями и вызовов методов друг друга. Строится для объектов кан-дидатов в классы системы	Детализиро-ванный сцена-рий варианта использования системы. Сущности кон-цептуальной модели. Сущности, выявленные в ходе анализа пригодности вариантов ис-пользования	Графическое пред-ставление взаимодей-ствий между объекта-ми во времени	Взаимодействующие объ-екты, последовательность и тип операций	Выявление поведения классов и формирова-ние методов классов
Стати-ческая модель анализа	Выявление допол-нительных взаи-мосвязей и со-става выявленных ранее сущностей, дальнейшая детализация. Это более присталь-ный взгляд на состав системы	Сущности кон-цептуальной модели. Сущности, выявленные в ходе анализа пригодности вариантов ис-пользования	Общий взгляд на статическую модель сущностей системы	Основные сущности си-стемы и связи между ними	Анализ установлен-ных при проведении анализа пригод-ности вариантов использования связей между классами-кандидатами, вы-явление избыточных и неоптимальных связей с целью устра-нения первых и опти-мизации вторых

Продолжение ⇨

(Продолжение)

Модель/ диаграмма	Цель	Источник	Назначение	Содержание	Использование
<b>Динамическая модель анализа</b>	Выявление до-полнительных взаимосвязей и состава вы-явленных ранее классов типа Copn- trol, дальнейшая детализация. Это более присталь-ный взгляд на поведенческую составляющую системы	Сущности, выявленные в ходе анализа при одности вариантов ис-пользования	Общий взгляд на динамическую модель сущностей системы	Основные контролеры системы и связи между ними	Анализ установлен-ных при проведении анализа пригодности вариантов использо-вания связей между классами типа Copn- trol с целью выявления избыточных и не-оптимальных связей, устранения первых и оптимизации вторых
<b>Логическая модель</b>	Отражение логи-ки реализации	Детализиро-ванные сцена-рии вариантов использования. Диаграммы последователь-ности. Модели ана-лиза	Отражение того, как распределяется по-ведение системы по классам. Логическая модель системы — модель классов системы с атрибутами, мето-дами, но без деталей реализации	Классы реализации, их атрибуты и основные методы. Не зависит от выбранного языка, плаформы. В этой модели еще не применялись ни архи-тектурные паттерны, ни паттерны объектно-ориентированного моде-лирования	Построение модели реализации системы
<b>Модель реали-зации</b>	Отражение клас-сов для реализа-ции в программ-ном коде	Логическая модель	Моделирование виде-ния архитектора по реализации системы. Дальнейшее развитие логической модели	Полное описание клас-сов системы и детали реализации — использу-емые паттерны и фрейм-ворки.	Разработка кода си- стемы

Модель/ диаграмма	Цель	Источник	Назначение	Содержание	Использование
Модель компонентов	Отражение компонентной архитектуры системы	Концепция системы. Логическая модель системы	системы — модель классов системы с атрибутами, методами, деталями реализации Моделирование основных компонентов системы и их зависимостей	В этой модели применены архитектурные паттерны и паттерны объектно-ориентированного моделирования Описание компонент и их зависимостей. Распределение функционала системы путем связи каждого класса с конкретным компонентом системы	Разработка модели развертывания. Понимание принципов построения системы, ее композиции и интерфейсов взаимодействия между компонентами
Модель развертывания	Описание развертывания системы на физических устройствах	Модель компонентов. Модель реализации	Моделирование распределения компонентов системы между физическими устройствами	Физические устройства, принимающие участие в функционировании системы, и протоколы взаимодействия между ними. Распределение процессорной модели системы по конкретным процессорам (физическим устройствам) путем связи каждого процесса с конкретным устройством	Развертывание системы

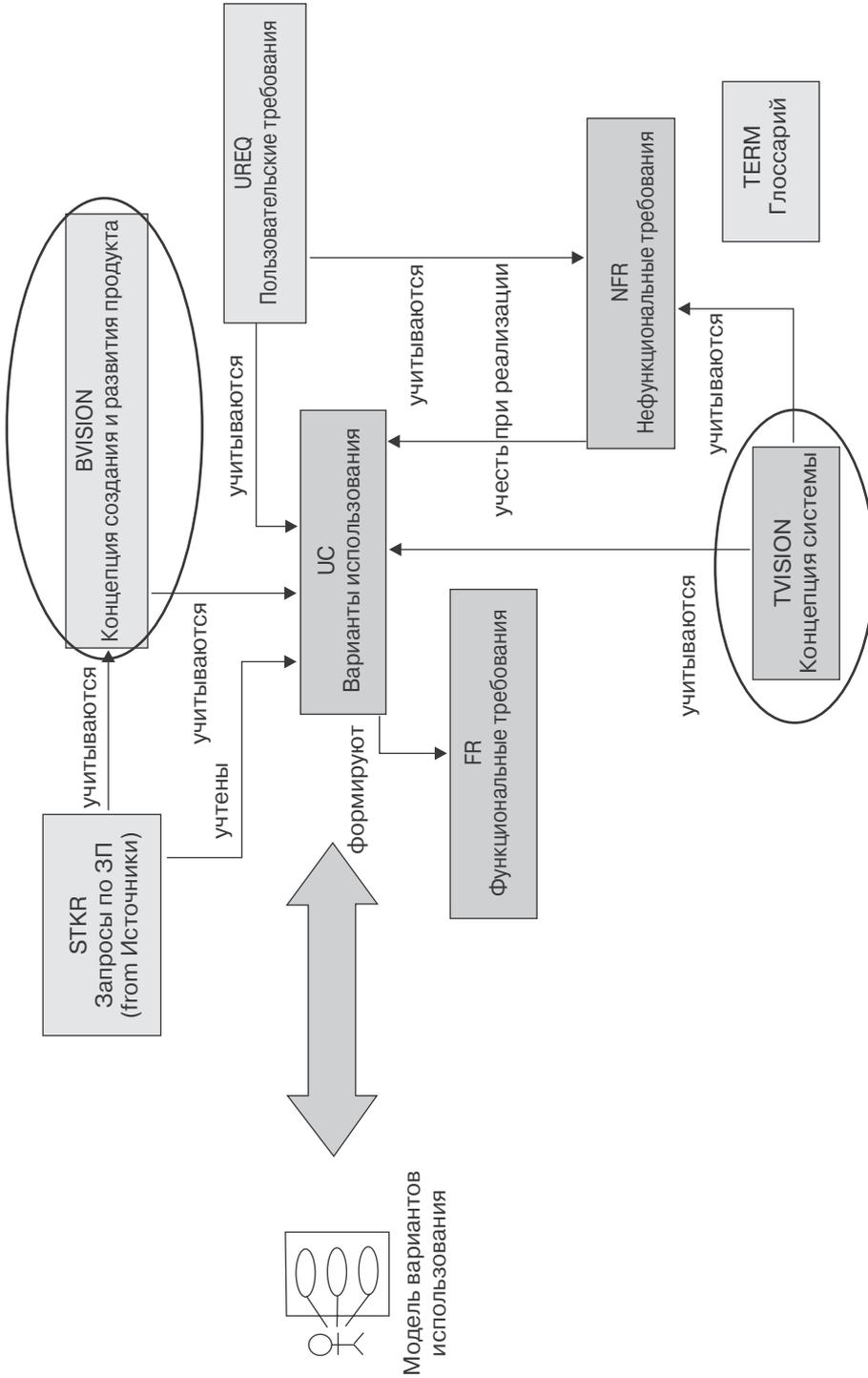
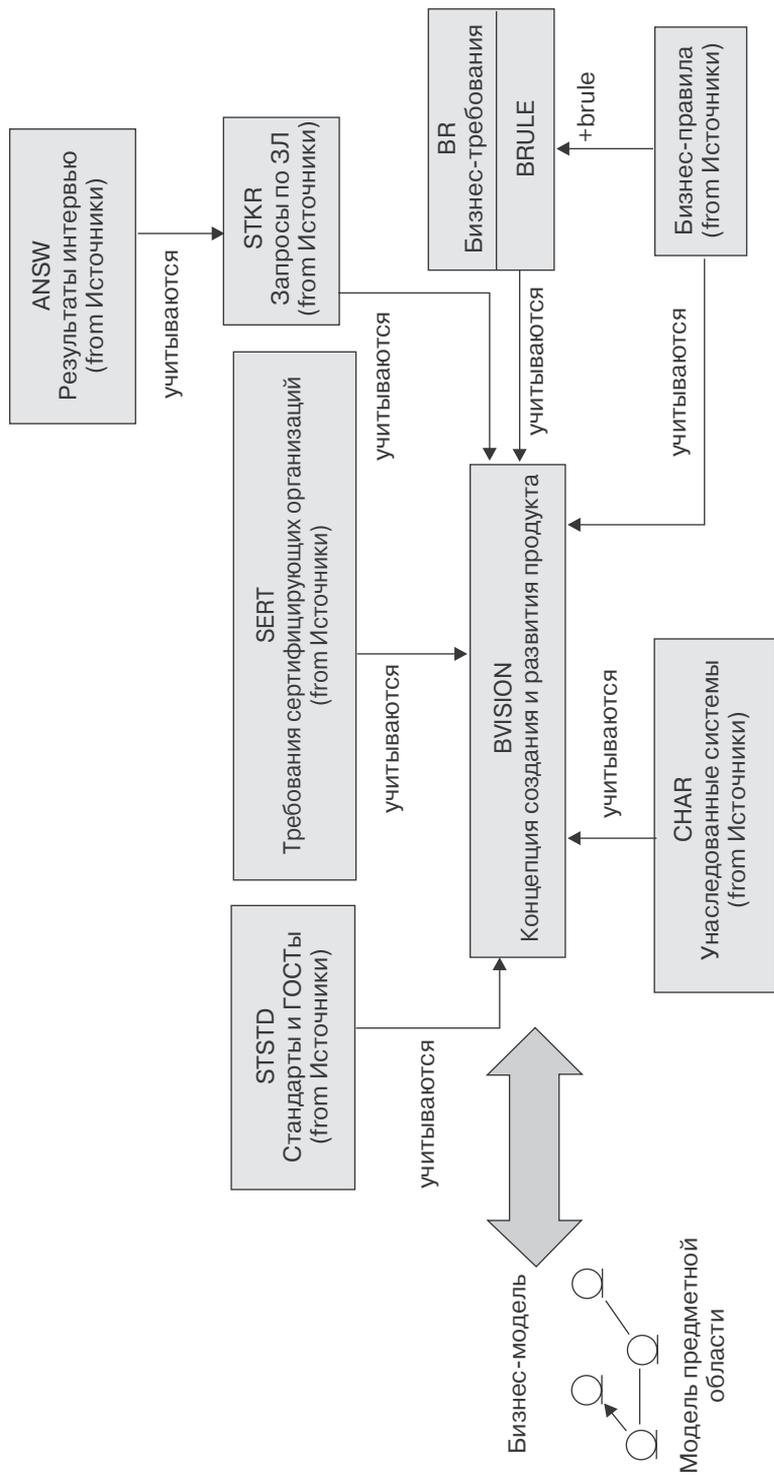


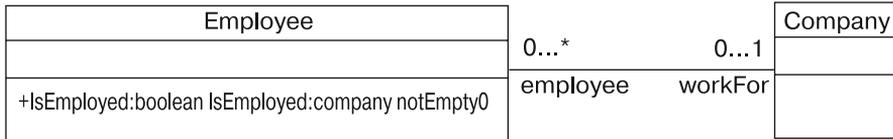
Рис. 45. Соответствие модели системы и требований к системе. Варианты использования



**Рис. 46.** Соответствие модели системы и требований к системе. Бизнес-модель и Business Vision



Для описания непосредственно в моделях ограничений бизнеса и системы язык UML расширяется языком **Object Constraint Management**. **Рекомендую вам изучить его конструкции и использовать в своей деятельности.** На рис. 47 приведен пример бизнес-правила на языке UML.



**Рис. 47.** Пример бизнес-правила на UML

Концептуальная модель требований концепции системы (TVISION) показана на рис. 48.

Для формирования требований этого типа особенно важны ограничения, допущения, характеристики унаследованных систем и результаты анализа технологий. Но это не все. Модель предметной области выступает в качестве бизнес-правила, а концептуальная модель системы может рассматриваться как требования типа TVISION. Кроме того, на эти требования существенное влияние оказывают компонентная модель и модель развертывания, которые создает архитектор.



Таким образом, в системном анализе моделирование и разработка требований выполняются параллельно в течение всего жизненного цикла разработки ПО, при этом модели и требования находятся в постоянном взаимном влиянии и уточнении.

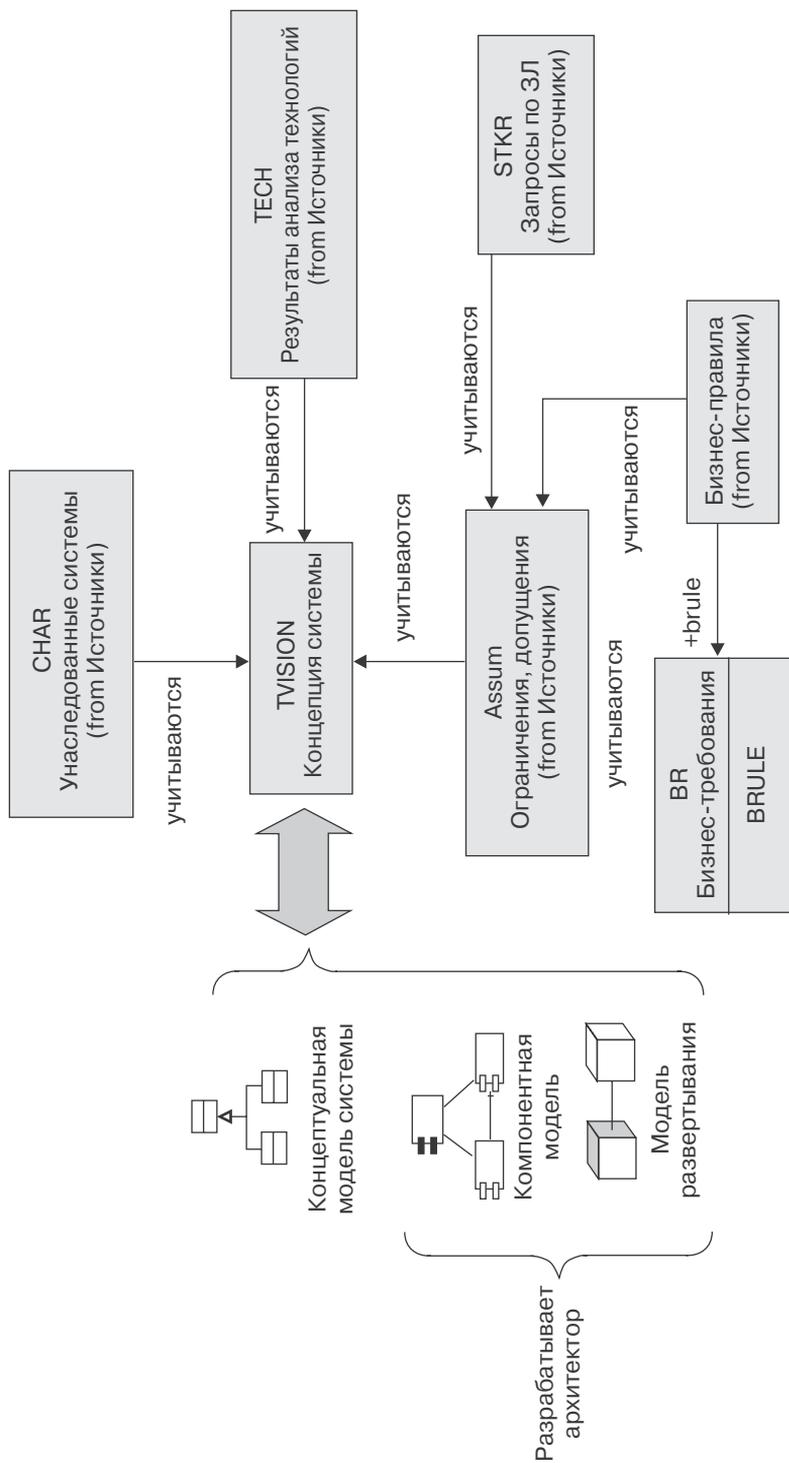
## 4.2.8. Трассировки

Мы уже неоднократно употребляли слово «трассировки» в контексте прослеживаемости дальнейшей судьбы требований.



Основная цель создания трассировок — убедиться в том, что мы не делаем того, что не нужно бизнесу или заинтересованным лицам, с одной стороны, и в том, что мы не забыли ничего из того, что действительно нужно бизнесу или заинтересованным лицам, — с другой.

Обычно трассировки строятся от одного типа требований к другому, далее к третьему, четвертому типам — пока не дойдем до «конечных» типов требований. «Конечным» типом требований обычно выбирается «вариант использования» или «функциональное требование» и «нефункциональное требование». Вся информация о трассировках, их назначениях и «конечных» типах требований фиксируется в ПУТ.



**Рис. 48.** Соответствие модели системы и требований к системе. Концептуальная модель системы и Technical Vision

В результате можно построить «дерево» трассировок, которое позволит проследить, как учтено «исходное» требование в «конечном». На рис. 49 показан пример такого «дерева».

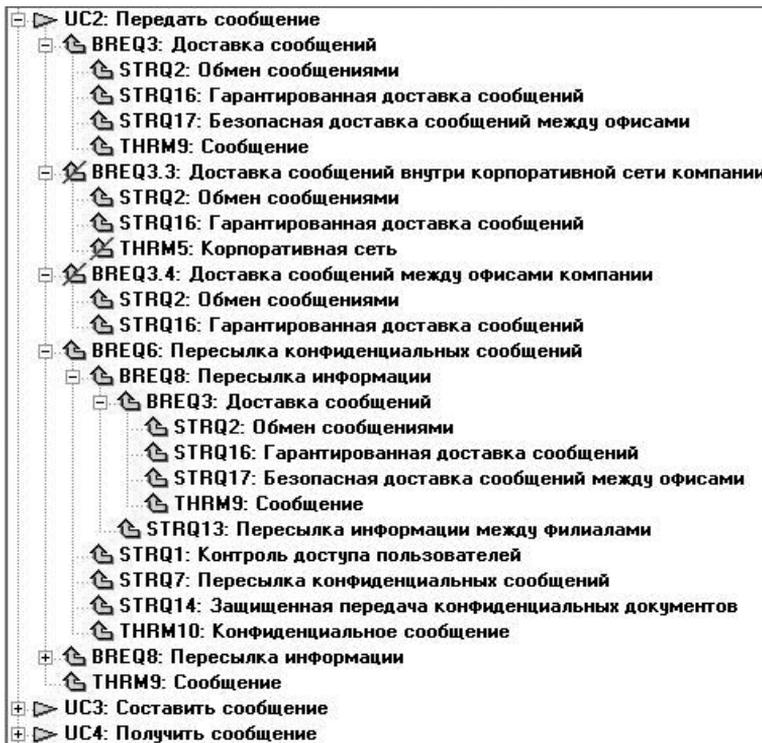


Рис. 49. Пример «дерева» трассировок требований

Обратите внимание на перечеркнутую стрелку на рис. 49. Это автоматическая функция инструмента Requisite PRO, которая говорит аналитику о том, что какое-то требование в цепочке трассировок изменилось и необходимо пересмотреть правильность и актуальность трассировок. Эта функциональность весьма полезна при проведении импакт-анализа (анализа влияния) запросов на изменение/создание функциональности.

Некоторые трассировки в большинстве проектов выполнять не обязательно, однако есть проекты, в которых я бы рекомендовал делать большинство трассировок. Решение о том, делать или не делать трассировки и если делать, то какие именно трассировки делать, принимает менеджер проекта совместно с системным аналитиком и тест-менеджером.

Все трассировки условно можно разделить на причинно-следственные (основные) и проверочные (дополнительные).

Причинно-следственные трассировки нужны для разработки качественного продукта. На диаграмме концептуальной модели типов требований они по-

казаны в виде стрелок. Такие трассировки являются инструментом аналитика и позволяют ему выполнить качественный системный анализ.

Проверочные трассировки необходимы в первую очередь менеджеру проекта как ответственному за качество продукта. Для выполнения системного анализа эта информация не нужна.



Если первый тип трассировок я рекомендую строить в упрощенном варианте всегда, непосредственно при выполнении анализа требований и выявлении из «исходных» требований следующих требований в иерархии типов требований, то по второму типу трассировок вы должны объяснять менеджеру проекта преимущества их создания и настаивать на включении в проект соответствующих работ.

От выполнения проверочных трассировок зависят успех проекта и удовлетворенность заказчика. На рис. 50 проверочные трассировки выделены курсивом, причинно-следственные — полужирным.

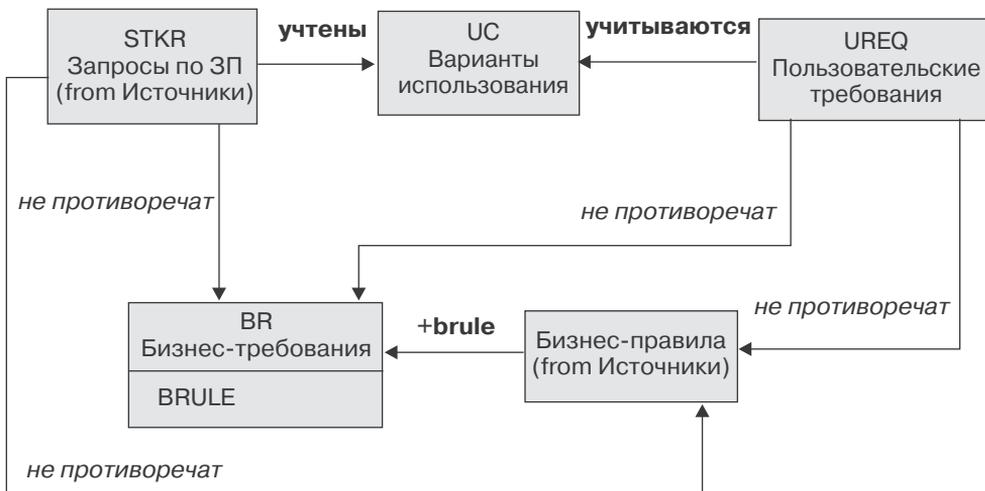


Рис. 50. Проверочные трассировки требований

### 4.2.9. Минутка расслабления

Теперь, когда вы знаете многое о системном анализе, разработке и управлении требованиями и моделировании, рассмотрим несколько несерьезных примеров из жизни, иллюстрирующих некоторые важные моменты в работе аналитика.



Часть этих примеров я прочитал на развлекательном сайте <http://zadolba.li/>, часть почерпнул из жалоб моих друзей и знакомых на какие-то информационные системы. В целом это довольно забавные ситуации, но если вдуматься, то в некоторых из них можно найти и повод для размышлений.

Пример	Комментарии
<p><i>Система контроля доступа. На двери электромагнитный замок. Чтобы выйти из помещения, клиент должен нажать кнопку. Естественно, объявления об этом никто не видит, стрелку, указывающую на кнопку, тоже, хотя все написано огромными буквами и выделено красным цветом. Когда говорю: «Нажмите кнопку слева от двери», клиент судорожно обшаривает дверь, начиная почему-то справа, и нажимает кнопку выключения света.</i></p> <p><i>Мы долго мучились бы с этой системой, но один умный человек нам помог. «Ребята, – сказал он, – красный цвет вызывает у людей подсознательное чувство опасности, и эти ваши красные кнопки и стрелки человек игнорирует не потому, что дурак, а потому, что подсознание дает ему команду их не видеть. Покрасьте кнопку для выхода в зеленый цвет, стрелку тоже сделайте зеленой и посмотрите, что будет».</i></p> <p><i>Вы не поверите, но это сработало! Количество клиентов, не способных найти кнопку, сократилось на порядок. Все внезапно поумнели, а вопроса «Как тут выйти?» мы практически не слышим.</i></p> <p><i>Если все клиенты кажутся вам патологически идиотами, может, надо у вас самих что-то подправить?</i></p>	<p>Если вдуматься, ситуация наглядно показывает необходимость учитывать особенности восприятия интерфейса человеком и важность такой профессии, как дизайнер пользовательских интерфейсов.</p> <p>Если провести аналогию с информационными системами, то, уверен, каждый из вас наткнулся на такие «красные стрелки» в программах и на сайтах, когда поведение системы отличалось от, казалось бы, логичного и ожидаемого или когда приходилось очень долго искать одну из основных функций системы в ее многоуровневом меню</p>
<ul style="list-style-type: none"> <li>– А в графе «Фамилия» писать свою фамилию или название фирмы?</li> <li>– Свою фамилию.</li> <li>– Боже, какой неудобный и нелогичный сайт!</li> </ul>	<p>Скорее всего, при разработке этого сайта тщательный анализ пользовательских требований не проводился.</p> <p>Подобная ситуация может сложиться и в том случае, если аналитик не владеет предметной областью и не понимает бизнес-сценариев. Если у пользователя возникает соблазн ввести название фирмы в графе «Фамилия», возможно, перепутана последовательность полей или вообще пользователь почему-то должен вводить данные повторно. Например, он уже внес информацию о себе один раз в форме «Данные о компании» и полагал, что в форме оформления бизнес-кредита нужно ввести название фирмы, а остальные данные система найдет сама</p>
<ul style="list-style-type: none"> <li>– Ну почему нельзя сделать так, чтобы я могла ввести свой адрес и возраст самостоятельно, а не выбирать его из списка?</li> </ul>	<p>И снова о пользе профессии дизайнера пользовательских интерфейсов.</p>

Пример	Комментарии
	<p>Мотив разработчиков подобной системы понятен — такая реализация необходима для того, чтобы впоследствии можно было проводить статистические выборки на основании информации, введенной пользователем. Но сегодня этим часто злоупотребляют.</p> <p>Практически на всех сайтах, где надо ввести дату в формате «день — месяц — год», пользователи почему-то вынуждены выбирать данные из списка. А ведь эту же функцию системы можно реализовать и как непосредственный прямой ввод данных с контролем со стороны системы, и как выбор даты из календаря, что было бы намного удобнее для пользователя. Но чуть-чуть сложнее для разработчиков</p>
<p><i>Писали с другом программу для автоматизации процессов печати приложений к диплому в одном из учебных заведений. Делали быстро, но на совесть и в работоспособности ее были уверены на 150 %. Сломаться или затереться ничего не могло в принципе.</i></p> <p><i>Наконец настал момент долгожданной сдачи проекта. Заказчику программа понравилась, и все разбежались довольными. Через некоторое время вызывают: «Не работает, и все тут».</i></p> <p><i>Приезжаем, смотрим. Сбились поля. Текст должен быть посередине, а на деле съезжает вправо. Долго думали, что могло повлиять на это. Перерыли весь код — ничего не нашли. Решили проверить в режиме дебага, что может вызывать такие неполадки, — все работает, как швейцарские часы. Ступор. Совершенное непонимание. Проверяем несколько раз, ответ очевиден: программа исправна.</i></p> <p><i>Едем на место и просим показать, как сотрудники заполняют формы. Результат не заставил себя долго ждать: девушка-секретарь, стуча ногтями по пробелу, вручную сдвигала текст на середину экранного поля. Понятно, что при печати текст выравнивался по середине с учетом пробелов. Интересуемся у дамочки:</i></p> <ul style="list-style-type: none"> <li>– Зачем вы это делаете? Ведь в инструкции написано, что при печати текст будет отформатирован автоматически.</li> <li>– Инструкция — это вредная бумажка, которую читают законченные ламеры!</li> </ul>	<p>Самая большая ошибка здесь в том, что изначальная концепция системы — «при печати текст будет отформатирован автоматически», то есть не реализованы принципы WYSIWYG (What You See Is What You Get — то, что вы видите, то вы и получите).</p> <p>Но даже если оставить эту логику, аналитик должен был бы подумать о таком развитии событий, когда пользователь будет пытаться выравнивать текст, добавляя лидирующие пробелы. Этот момент можно было учесть при сохранении либо выводе документа на печать, например убирая лидирующие пробелы или непосредственно при вводе трех пробелов подряд отображая всплывающую на две секунды подсказку о том, что текст будет выровнен по центру при печати.</p> <p>Такое поведение системы должно было быть описано в соответствующем сценарии использования системы</p>

А в вашей жизни или в жизни каких-либо ваших знакомых были подобные ситуации?



## 4.3. Типичные проблемы и вопросы

В этом разделе приведены наиболее типичные проблемы и вопросы, с которыми сталкивается аналитик.

Вопрос / проблема	Рекомендация / комментарий
Управление требованиями? Нам это не нужно	<p>Достаточно распространенная позиция компаний, в которых долгое время была принята «гаражная разработка».</p> <p>Здесь нужно набраться терпения и доступно и последовательно объяснять, какие цели преследует управление требованиями, что это дает разным проектным ролям.</p> <p>Рекомендую вопросы управления требованиями разделять на две области: управление атрибутами требований (здесь обычно понимания больше — важность, приоритет, архитектурная значимость — понятия, знакомые и близкие большинству участников проектной команды) и управление состояниями требований (этот вопрос должен быть максимально интересен менеджеру проекта и менеджеру по тестированию)</p>
«Битва за WBS-аналитика»	<p>Вы будете постоянно сталкиваться с тем, что менеджер проекта будет стараться не включать работы по созданию трассировок требованиям, проведению импакт-анализа и созданию некоторых моделей в WBS проекта.</p> <p>Здесь рекомендация одна — вовлекайте в принятие таких решений другие проектные роли. Архитектор и тест-менеджер помогут вам убедить МП в важности выполнения этих работ. Кроме того, советую выполнять причинно-следственные трассировки независимо от настроений менеджера проекта, включая эти трудозатраты в затраты на подготовку спецификаций соответствующих требований или моделей</p>
Удавка времени, или Быстрее, еще быстрее	<p>Времени будет не хватать. Всегда. Как бы быстро и продуктивно вы ни работали, вы постоянно будете чувствовать дыхание менеджера в затылок.</p> <p>Кроме рекомендаций по эффективному планированию собственного времени и развития навыков самоорганизации, о которых мы говорили в разделе «Личностные навыки», я рекомендую вам всегда давать оценку трудоемкости поставленных вам задач и согласовывать ее с менеджером проекта.</p> <p>В ваших силах превратить «удавку времени на вашей шее» хотя бы в деловой галстук</p>

Вопрос / проблема	Рекомендация / комментарий
Как заслужить уважение разработчика?	<p>Очень многие разработчики считают, что аналитик — это неудавшийся программист.</p> <p>Заслужить уважение разработчика можно, только разговаривая с ним на одном языке. Научитесь читать программный код. Создавайте ясные и недвусмысленные требования. Понимайте основы ООП и научитесь строить грамотные диаграммы последовательности.</p> <p>Рекомендую вам взять домой несколько модулей из любой разработанной в вашей компании системы, разобраться в классах и построить диаграммы последовательности. Это позволит вам улучшить навыки и лучше узнать принятые в компании стандарты кодирования (если они, конечно, есть)</p>
Так нужны все-таки спецификации требований или нет?	<p>Со временем возникает очень большой соблазн отказаться от создания спецификаций требований в пользу моделирования и управления требованиями.</p> <p>Есть как минимум два серьезных аргумента против:</p> <ul style="list-style-type: none"> <li>• заказчик более охотно работает со спецификациями требований, чем с какой бы то ни было другой формой организации требований;</li> <li>• когда разработкой продукта занимается распределенная команда из разных компаний, спецификации являются иногда единственным возможным интерфейсом, так как в разных компаниях могут использоваться разные методологии и инструменты.</li> </ul> <p>Рекомендую выбор каждой спецификации требований делать обдуманно и обоснованно, так как это дополнительные трудозатраты</p>
Требования опять не нужны?	<p>Будьте готовы к тому, что этот вопрос будет возникать перед вами с завидной периодичностью. Наберитесь терпения и попытайтесь понять, почему ваши коллеги так говорят? Это просто привычка или что-то в процессе разработки и управления требованиями им неудобно? А может быть, они привыкли работать только со спецификациями и не хотят начинать учиться работать с какими бы то ни было инструментами?</p> <p>Старайтесь найти общий язык с командой. Требования в виде проекта требований нужны тогда, и только тогда, когда ими будут управлять и отслеживать развитие требований (проводить трассировки).</p> <p>В иных случаях действительно можно обойтись только спецификациями</p>
Что дальше с точки зрения Пути аналитика?	<p>На этом этапе вашей карьеры вы можете перейти либо в следующую категорию — «Старший аналитик» и остаться на Пути аналитика, либо в область разработки и попытаться стать разработчиком или системным архитектором</p>

## 4.4. Заключение

Мы рассмотрели необходимые аналитику личностные навыки, узнали, что такое ПУТ, и познакомились с его возможной структурой, последовательно прошли через все этапы моделирования и узнали, какие модели и для чего существуют в системном анализе.

Для более глубокого усвоения информации рекомендую вам прочитать все источники (книги и статьи), на которые я ссылался в этой главе, а также ознакомиться с другими материалами, указанными в квалификационной таблице навыков (табл. 7).

**Таблица 7.** Профессиональные и специальные навыки аналитика

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
Аналитик	<p>Все теоретические знания младшего аналитика, а также знание и умение применять ГОСТ 34-й и 19-й серий, нотацию IDEF0, диаграммы EPC (extended Event Process Chain).</p> <p>Книги:</p> <ul style="list-style-type: none"> <li>• <i>Rosenberg Doug, Scott Kendall. Use Case Driven Object Modeling with UML: A Practical Approach;</i></li> <li>• <i>Rosenberg Doug. Agile Development with Iconix Process.</i></li> </ul> <p>Знание основ RUP – дисциплины Requirements, Analysis &amp; Design.</p> <p>Знать и уметь строить Robustness- и Sequence-диаграммы, analysis model.</p> <p>Расширять свой кругозор в методологиях. Ознакомиться с гибкими методологиями, например с Iconix.</p> <p>Рекомендую регулярно посещать сайты <a href="http://www.uml2.ru">www.uml2.ru</a>, <a href="http://www.agilerussia.ru/">http://www.agilerussia.ru/</a></p>	<p>Все навыки младшего аналитика, а также:</p> <ul style="list-style-type: none"> <li>• знать, что такое ПУТ, и уметь его разрабатывать;</li> <li>• понимать, какие модели существуют и их место в разработке ПО;</li> <li>• уметь создавать модель анализа;</li> <li>• строить Robustness- и Sequence-диаграммы, понимать, зачем их вообще надо строить;</li> <li>• уметь читать программный код;</li> <li>• иметь навыки проведения презентаций</li> </ul>

# 5. Старший/ведущий аналитик

---

---

## 5.1. Личностные навыки

### 5.1.1. Самомотивация

Сосредоточьтесь на возможностях, предоставляемых работой, а не на ее недостатках. Как профессиональный аналитик вы уже можете управлять требованиями к продукту, проводить обучающие тренинги и презентации, развивать методологии системного анализа и разработки, заниматься совершенствованием процессов в компании.



---

На этом этапе карьеры должен появиться план построения собственной карьеры. Отнеситесь к этому серьезно. Сформулируйте собственные ожидания, постарайтесь «увидеть себя» через три года. Поговорите с вашим непосредственным руководителем. Возможно, он укажет вам на недостаточно развитые, с его точки зрения, личные и профессиональные качества, подскажет, над чем вам нужно поработать в первую очередь. Приведите свой план самообучения в соответствие с этими рекомендациями.

---

### 5.1.2. Навыки публичного выступления

На данном этапе вашей карьеры вам придется выступать с презентациями и докладами чаще, чем раньше.

Далее приведены некоторые рекомендации, которые касаются непосредственно навыка проведения презентаций, а также помогут вам в построении эффективных коммуникаций.

## Избегайте излишней претенциозности

Задача выступающего с презентацией человека — информировать, а не производить впечатление! Избегайте сложных формальных слов и фраз, чтобы не возвести стену непонимания между вами и аудиторией. Простота изложения — ключ к взаимопониманию.

## Банальность

В процессе создания конспекта вашего доклада избегайте избитых клише и тривиальных фраз. Используйте слова, которые точно отражают ваши идеи. Ваше послание будет более действенным, если вы применяете собственные слова и обороты.

## Жаргон

Прежде чем использовать в своей презентации корпоративный и специализированный жаргон или термины, вы должны убедиться, что аудитория поймет, о чем вы говорите. В противном случае ваши слова окажутся для аудитории непонятными.

Если необходимо применить специальные термины и слова, которые часть аудитории может не понять, постарайтесь дать краткую характеристику каждому из них, когда употребляете их в процессе презентации впервые. Можно применять сделанные в виде листовки «памятки», чтобы аудитория пользовалась ими в процессе презентации.

## Плохие привычки

Итак, допустим, вы сделали отличную презентацию и продемонстрировали хорошие визуальные материалы. Но аудитория не проявляет интереса. Причиной могут быть ваши привычки, которые раздражают аудиторию. Хуже того, вы даже можете не подозревать, что именно стало причиной отторжения. Ниже описаны самые распространенные раздражающие привычки.

*Слова-паразиты:* «значит», «на самом деле», «как бы» и т. п. Они отталкивают слушателей от самой сути презентации и сводят на нет ваш авторитет как докладчика. Кроме того, создают впечатление, что вы плохо приготовились к презентации.

*Движения-паразиты:*

- ◆ игра с ювелирными украшениями (например, с цепочкой или кольцом) или с усами или бородой (у мужчин), с локонами волос (у женщин);
- ◆ облизывание или покусывание губ;
- ◆ постоянное поправление очков;
- ◆ пощелкивание ручкой;

- ◆ позвякивание содержимым карманов;
- ◆ привычка на что-нибудь опираться.

*Излишняя жестикуляция.* Хуже отсутствия жестикуляции может быть только ее излишек.

*Вызывающая одежда.* К сожалению, вызывающая одежда говорит громче, чем вы.

*Безграмотность.* Орфографические ошибки свидетельствуют о вашей небрежности и несерьезности.

*«Говорит и показывает спина».* Постарайтесь во время презентации общаться именно со слушателями, а не с экраном, маркерной доской или флипчартом. Если вам необходимо обратить взор к вышеуказанным вещам, сделайте это, повернувшись не более чем на 45 градусов.

*Не используйте собственные пальцы в качестве указки.* Для этого существуют соответствующие аксессуары.

*Проводя презентацию, выступайте с речью:*

- ◆ плох тот докладчик, который читает непосредственно с листа. Один из самых простых способов потерять аудиторию — чтение презентации;
- ◆ не перечитывайте демонстрируемый на слайдах текст. Слушатели способны сделать это самостоятельно. Лучше потратьте больше словесных аргументов на оживление демонстрируемых графиков или картинок.



В дополнение приведу несколько приемов, которые мне всегда помогали в публичных выступлениях:

- ◆ не думать о будущих отзывах и оценках, которые дадут вашим выступлениям;
- ◆ думать о встрече с новыми незнакомыми и интересными людьми, с их убеждениями, страхами и опасениями;
- ◆ справиться с волнением помогает, если представить, что в зале находится один человек, с которым вам предстоит поговорить, и это ваш хороший друг;
- ◆ во время презентации найти глаза тех людей, которые готовы слушать или уже слушают вас с интересом, и поддерживать контакт в первую очередь с ними.

### 5.1.3. Продолжайте...

Изучать теорию в самых разных областях, смежные дисциплины. Это и требования, и архитектура, и управление проектами, и процессы разработки ПО в общем, и ИТ.

Вести собственную базу выученных уроков, в которую заносите все сложные ситуации, которые будут возникать в вашей жизни. Анализируйте эти ситуации и вырабатывайте персональные лучшие практики.

Выполняйте план собственного профессионального и личностного развития. Наполняйте ваше хранилище знаний. Раз в полгода пересматривайте это хранилище, удаляйте лишнее и отмечайте собственный прогресс и рост.

Делитесь полученными знаниями с вашими коллегами. Делайте это бескорыстно, как в поговорке: «Делай добро и бросай его в воду».

## 5.2. Профессиональные и специальные навыки

На этом этапе своего карьерного и профессионального пути вы должны обладать следующими навыками, перечисленными в табл. 8.

**Таблица 8.** Профессиональные и специальные навыки ведущего аналитика

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
Старший аналитик	<p>Все теоретические знания аналитика.</p> <p>Книги:</p> <ul style="list-style-type: none"> <li>• Данилин А., Слюсаренко А. Архитектура и Стратегия. «Инь» и «Янь» информационных технологий предприятия;</li> <li>• Boehm Barry W. A Spiral Model of Software Development and Enhancement;</li> <li>• A Guide to the Project Management Body of Knowledge. ANSI/PMI.</li> </ul> <p>Статья: Ebert C. Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques.</p> <p>Рекомендую регулярно посещать сайты <a href="http://msdn.microsoft.com/ru-ru/default.aspx">http://msdn.microsoft.com/ru-ru/default.aspx</a> и <a href="http://www-01.ibm.com/software/success/cssdb.nsf/topstoriesFM?OpenForm&amp;Site=rational&amp;cty=en_us">http://www-01.ibm.com/software/success/cssdb.nsf/topstoriesFM?OpenForm&amp;Site=rational&amp;cty=en_us</a></p>	<p>Все навыки аналитика, а также:</p> <ul style="list-style-type: none"> <li>• иметь детальное представление о ЖЦ проекта и продукта;</li> <li>• знать, что такое ПУД, и уметь создавать его;</li> <li>• уметь создавать логическую модель и модель данных;</li> <li>• уметь создавать простой программный код (этот навык нужен для закрепления навыка чтения программного кода. Никто так не поймет разработчика, как другой разработчик, и, если вы хотите разговаривать с разработчиками на одном языке, вам придется освоить программирование в минимальном объеме);</li> <li>• уметь профессионально проводить презентации;</li> <li>• проводить выученные уроки по практикам разработки и управления требованиями;</li> <li>• быть наставником для аналитиков;</li> <li>• уметь предотвращать и разрешать конфликты в проектах;</li> <li>• уметь выявлять риски и управлять ими</li> </ul>

## 5.2.1. Проведение презентаций и тренингов

В этом разделе мы остановимся на алгоритме проведения презентаций — планирование, создание, подготовка к выступлению и само выступление.

### Планирование презентации

Для проведения успешной презентации особенно важно следующее.

- ◆ Определить цель презентации.

Четко определите цель презентации, исходя из ожидаемых результатов ее проведения. Вы стремитесь что-то донести до аудитории, убедить ее, научить?

- ◆ Установить, кто ваша целевая аудитория, каков уровень ее подготовки.

Определите отношение аудитории к теме презентации, к вам лично, к вашей компании, проанализируйте должности и круг обязанностей заинтересованных лиц, их возраст, оцените знание аудиторией предмета. Спросите себя: «Насколько аудитория посвящена в эту тему?», «На кого я могу опираться в докладе?», «Какие практики считаются хорошими в этой аудитории?».

- ◆ Понять, чего хочет от вас аудитория.

Аудитория обычно ожидает от выступающего:

- ◆ знания предмета и подготовленности к выступлению;
- ◆ четко и понятно сформулированной идеи;
- ◆ умения быть кратким и концентрировать внимание аудитории на обсуждаемой теме;
- ◆ привязки излагаемой вами идеи к практике;
- ◆ организованности и соблюдения регламента.

### Создание презентации

Создайте структуру презентации в MS Power Point. Для этого определите ключевые моменты доклада на основе анализа аудитории. Надо ли произносить вступительное слово, объясняющее, кто вы и откуда, следует ли определять место этой презентации в каком-то цикле работ, есть ли необходимость осветить теоретические основы, если да, то какие, нужно ли мотивировать аудиторию на поиск каких-то решений?

На первом слайде, кроме темы доклада, всегда должны быть указаны ваша фамилия, должность и название компании, от имени которой вы выступаете.

На втором слайде рекомендуется определять цели презентации.

На третьем — показывать основные пункты содержания презентации, желательно с информацией, для кого эти пункты будут наиболее интересны.

На четвертом — рассказать о регламенте проведения презентации и правилах участия в ней: можно ли вас перебивать во время выступления, будет ли отведено время для вопросов и дискуссий и т. д.

Уточните содержание презентации:

- ◆ определите полный круг вопросов, которые вы собираетесь охватить;
- ◆ исключите из него вторичное, что никак не связано с определенной вами целью презентации и ожиданиями аудитории;
- ◆ определите, как должно изменяться поведение аудитории в ходе презентации, например: нейтральность — заинтересованность — скепсис — активное дискутирование — сомнения — приглашение к дальнейшему диалогу;
- ◆ предусмотрите меры управления эмоциональным климатом: шутки, анекдот, цитаты, примеры из жизни. Не забудьте учесть характеристики аудитории — ее возраст, вероисповедание, национальность, позицию в компании, чтобы эти составляющие презентации были уместны и никого бы не задели;
- ◆ определите подходящие визуальные образы и иллюстрации ваших идей; создайте иллюстрации в форме плакатов или больших и наглядных диаграмм;
- ◆ подберите практические примеры для пояснения ваших идей. Желательно, чтобы эти примеры были связаны с повседневной деятельностью аудитории презентации.

Старайтесь выдерживать оптимальную размерность слайда:

- ◆ одна идея — один слайд;
- ◆ один слайд — 5–6 строк;
- ◆ одна строка — одна мысль-высказывание;
- ◆ одна строка — 5–6 слов.

Подумайте над оформлением слайдов:

- ◆ выберите единую цветовую схему и придерживайтесь ее. Используйте корпоративный шаблон презентации, если он существует в компании;
- ◆ обязательно нумеруйте слайды. Оптимальный вариант указания номера — например, «Слайд 12 из 50». Это позволит слушателям в любой момент чувствовать движение презентации;
- ◆ не увлекайтесь спецэффектами при создании слайдов. Очень часто они отвлекают от основной мысли;
- ◆ не перегружайте слайд текстом. Оставляйте только главные мысли. Остальное включайте в раздаточные материалы;
- ◆ везде, где есть возможность, используйте графическое представление информации.

Составьте временную линейку презентации, например, в виде комментария «10/0,5» в первой строке примечания к слайду, что означает: «Я должен потратить 0,5 минуты на данный слайд, общая длительность этого раздела презентации — 10 минут».

## Подготовка к проведению презентации

Делайте тренировочные прогоны презентации, сначала самостоятельно, без учета временной линейки. Когда вы отточите свою речь, подгоните ее под временные ограничения, после чего попросите коллегу помочь вам выполнить несколько прогонов с «секундомером» по временной линейке презентации.

Учитывайте, что на реальное выступление вам потребуется как минимум на 20–30 % времени больше, чем на прогоне.

Выучите вступление и заключительные фразы.

Ознакомьтесь с помещением и оборудованием, где будет проходить презентация.

Подготовьте печатные материалы, письменные принадлежности для ее участников.

Позаботьтесь о прочем обеспечении презентации (вода, кофе, чай и т. п.).

Придите в аудиторию как минимум за 20 минут до начала презентации, проверьте, как звучит ваш голос, постарайтесь «почувствовать» помещение.

Убедитесь, что текст, графика и звук разборчивы с любого места.

Разложите раздаточные материалы для участников презентации.

Сложите ваши «вспомогательные» материалы в удобном для вас порядке.

Бегло просмотрите свою презентацию, освежив основные моменты, и проследите, как они влияют на цель презентации. Настройте себя на достижение цели презентации.

Постарайтесь успокоиться.

## Выступление

Используйте навыки публичного выступления, описанные в соответствующем разделе этой главы.

В конце презентации обязательно поблагодарите за время, уделенное вам. Последний слайд презентации должен содержать просьбу присылать отзывы о презентации и ваш e-mail.

Позаботьтесь о выполнении своих обещаний, данных во время презентации.

Более подробные материалы о подготовке и проведении презентаций вы можете скачать на сайте книги.

### 5.2.2. Реверс-инжиниринг требований/ обратное проектирование системы

Предпосылки реверс-инжиниринга требований:

- ◆ система работает хорошо, но никто не может уверенно сказать, как она работает, и не знает всех ее функций;

- ◆ изменились бизнес-требования/стратегия ИТ, и необходимо адаптировать систему;
- ◆ появилась новая технология, и требуется улучшение системы;
- ◆ возникли новые ожидания ключевых заинтересованных лиц, и нужно улучшить систему.

Для решения этих вопросов есть как минимум два пути: разработать новую систему или изменить существующую. Например, когда требования бизнеса значительно изменились, целесообразно разработать новую систему. Решение об этом принимает ССВ (Change Control Board) продукта. Мы остановимся на варианте, когда решено все-таки модернизировать систему. Здесь и возникает главная проблема: как же можно улучшать «черный ящик»? Сначала нужно понять, что «у нее внутри». ©

С этой целью и проводят реверс-инжиниринг требований, или обратное проектирование системы. Основные цели обратного проектирования системы:

- ◆ избежать повторной разработки с нуля уже существующих и нормально работающих функций системы;
- ◆ выявить и зафиксировать требования к системе для обеспечения возможности изменения ее функциональности в будущем;
- ◆ повысить качество системы путем тестирования системы по восстановленным требованиям.

Как и при прямом анализе, рекомендуется:

- ◆ определить границы системы;
- ◆ восстановить функциональность системы в виде Use Case-модели;
- ◆ построить модель анализа (логическую модель);
- ◆ построить архитектурные модели.

Сложность задачи заключается в том, что при прямом анализе мы опускаемся с высокого уровня абстракции ниже и ниже путем последовательной детализации. А при реверсном подходе должны подниматься с уровня конкретной реализации и программного кода выше и выше, в сторону более абстрактной модели системы.



---

В качестве вычислительно независимой модели рекомендую рассматривать модель вариантов использования и концептуальную модель системы. Если это возможно и код хорошо структурирован, лучше сразу проводить функциональную декомпозицию и восстанавливать требования для отдельных подсистем.

---

В качестве платформоориентированной модели часто рассматривают модель дизайна или программный код (рис. 51).

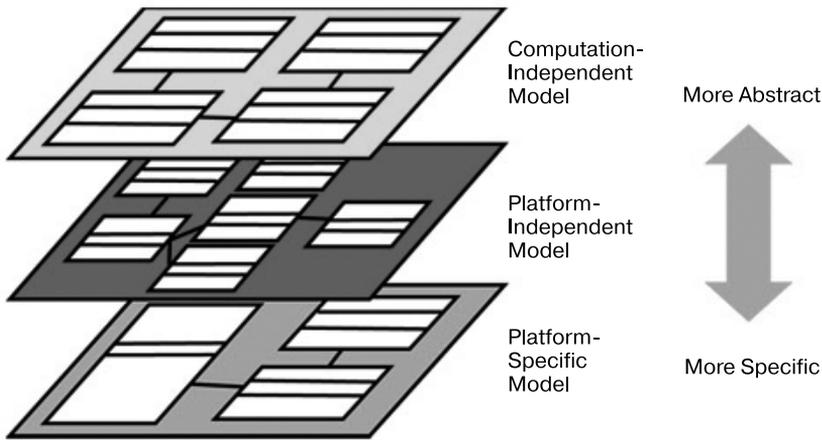


Рис. 51. MDA — Model-Driven Architecture (OMG)

Однако зачастую модели дизайна не существует в принципе (а если ее построить с применением автоматизированных средств моделирования, она, как правило, нечитаемая), поэтому я рекомендую использовать платформонезависимую модель в виде логической модели системы. При этом не нужно строить логическую модель, отражающую все классы реализации. Достаточно выделить какие-то основные классы и их взаимосвязи. В этом аналитику должен помочь архитектор. Но советую вам самостоятельно анализировать исходный код, чтобы строить логическую модель.

В идеале параллельно с этой работой системный аналитик совместно с системным архитектором восстанавливает архитектуру системы. Восстановление архитектуры — отдельная объемная тема, которая выходит за рамки нашей книги.



Кроме чтения кода, естественным и очень удобным способом его изучения является анализ кода в режиме отладки: когда вы запускаете программный код на исполнение, прерываете исполнение кода и в среде отладки кода пошагово выполняете инструкции. Основная сложность здесь в настройке полноценного рабочего места разработчика на рабочем месте аналитика. Аналитик не всегда сможет сделать это самостоятельно, поскольку данная процедура иногда требует серьезных затрат времени на обустройство ПК и настройку рабочей среды приложения. Возможно, потребуется помощь разработчика или менеджера по конфигурации.

Такой подход не всегда может быть реализован ввиду сложности/невозможности развертывания среды системы в режиме отладки. Более простой вариант — организовать тестовый стенд, на котором развернуть систему и изучать ее поведение.

В общих чертах алгоритм по построению логической модели такой: после беседы с архитектором или разработчиком об основных классах, их назначении и взаимодействии начать изучать программный код и строить диаграммы последовательности. Разработанные диаграммы последовательности нужно обязательно согласовывать с разработчиками. Не стремитесь отразить все методы всех выбранных значимых классов. Отображайте только самое важное.

Еще одним источником информации для построения логической модели является база данных. Сущности логической модели реализуются классами модели реализации, и в конечном итоге объекты многих из классов реализации сохраняются в базе данных. Поэтому очень важно построить схему базы данных и постараться, в частности, выявить из нее классы логической модели.

После того как вы построите диаграммы последовательности и выявите классы из БД, используйте практику построения аналитических моделей для создания логической модели. Для этого постройте новую диаграмму и с помощью *drag & drop* последовательно поместите на нее классы, которые создали при разработке диаграмм последовательности. Если класс А вызывает метод класса Б, свяжите эти классы в логической модели ассоциацией  $A \rightarrow B$ . Если класс Б вызывает класс А, сделайте эту ассоциацию двунаправленной. На данном этапе требуется еще раз согласовать драфт логической модели с разработчиком и постараться вместе с ним определить отношения между классами типа «агрегация». Отображая классы в логической модели, всегда пишите в описании класса логической модели название класса (или нескольких классов) модели реализации, то есть названия конечных классов программного кода. Это трассировка классов логической модели на классы модели реализации.

Параллельно надо изучать всю доступную документацию, чтобы построить модель вариантов использования. Особое внимание обращайте на пользовательскую и техническую документацию. Из описаний действий пользователя очень часто можно выявить варианты использования системы.



---

Кроме того, интервьюируйте экспертов предметной области и владельцев бизнес-процессов, которые эта система автоматизирует. Как они используют систему? Каковы сценарии их взаимодействия с системой? Эта информация — ключевая для выявления вариантов использования системы и функциональных требований. Если вы не можете «вытянуть» из эксперта весь сценарий взаимодействия, зафиксируйте его видение в форме функциональных требований. Это эффективная практика.

---



---

Построив модель вариантов использования и имея очередной драфт логической модели, вы должны выполнить самую сложную задачу — связать их воедино. Для этих целей идеально подходят диаграммы *View Of Participated Classes (VOPC)*. Это трассировка вариантов использования системы на классы логической модели. На такой диаграмме классов отображаются классы и ассоциации между ними, которые реализуют конкретный вариант использования. Строго говоря, VOPC-диаграммы строятся для каждого Use Case

Realization — сквозного прохода по сценарию варианта использования без альтернативных потоков. Однако я не рекомендую тратить время на явное выделение Use Case Realization — достаточно «привязать» к одному варианту использования несколько диаграмм классов для разных Use Case Realization. Тем не менее здесь, как всегда, нет жесткого правила. Если это поможет вам или проектной команде — выделяйте Use Case Realization в явном виде. В процессе работы, возможно, в очередной раз изменится логическая модель. Все изменения должны быть согласованы с разработчиками и архитектором системы.

---

На этом этапе реверс-инжиниринга я рекомендую еще раз пересмотреть диаграммы последовательности и привязать к ним шаги сценариев варианта использования. Обычно это осуществляется путем размещения слева от диаграммы текста сценария варианта использования таким образом, чтобы шаг сценария был напротив взаимодействия классов, реализующих данный шаг.

Выполнив эти действия, вы получите описание системы в виде акторов и вариантов использования ими системы с диаграммами VOPC, диаграммами последовательности и логической модели. Если вам ранее не приходилось заниматься функциональной декомпозицией, то сейчас самое время сделать это. По окончании функциональной декомпозиции можно говорить о том, что вы завершили обратное проектирование системы и выполнили значительную часть реверс-инжиниринга требований.

Однако это еще не все. Необходимо выявить нефункциональные требования. Здесь основным источником информации должны стать технические инструкции по развертыванию и установке системы, описание ограничений системы в рабочих инструкциях пользователей и результаты нагрузочного тестирования.

Надо быть готовыми к тому, что некоторую функциональность системы или части логической модели будет просто невозможно восстановить или построить. Причинами здесь могут быть плохо структурированный и некомментированный код, по которому невозможно получить консультацию разработчика; отсутствие документации и экспертов.



---

В процессе реверс-инжиниринга обязательно управляйте требованиями и стройте трассировки между разными типами требований. Я рекомендую вам использовать типы требований FR, UC, NFR, TERM (см. подраздел «Концептуальная модель типов требований» раздела «Профессиональные и специальные навыки» в главе «Аналитик»), а также ввести добавочные требования для документации, которую вы будете анализировать, — пользовательской документации, технических инструкций по установке и развертыванию, рабочих инструкций.

---



---

Если вы работаете с инструментом Rational Requisite PRO, советую воспользоваться функцией импорта документа, это позволяет размечать документ в Microsoft Word путем выделения требования непосредственно в тексте

документа. Очень удобно импортировать в проект требований всю доступную документацию и выделять в ней добавочные требования. Выделенные добавочные требования рекомендую трассировать на варианты использования, функциональные и нефункциональные требования, которые вы выявили из анализа добавочных. Это очень важно для всестороннего анализа требований и устранения противоречивости и неоднозначности.



Для управления проектом реверс-инжиниринга большое значение имеют итеративность и последовательно-поступательное движение — от общих вариантов использования и главных классов к более детальным вариантам использования и более полному списку классов. Степень гранулярности должна определяться целями проведения реверс-инжиниринга. Если требуется внести изменения в какую-то одну функцию одной подсистемы — нет смысла ни описывать все варианты использования этой подсистемы, ни тем более пытаться выполнить обратное проектирование для всей системы в целом. Это целесообразно, если запрос на изменение не влияет на нефункциональные требования, в противном случае вполне возможно, что потребуется восстановление требований.

### 5.2.3. ООМ и паттерны проектирования



Старший аналитик должен иметь и развивать навыки объектно-ориентированного анализа и проектирования. Я рекомендую вам в свое свободное время изучить паттерны ООАП:

- ◆ <http://oodad.asf.ru/Patterns.aspx>;
- ◆ <http://www.dotsite.ru/Solutions/patterns/>;
- ◆ [http://www.oodad.org/main/00AD\\_Books.htm](http://www.oodad.org/main/00AD_Books.htm).

Я учился по сайтам [oodad.asf](http://oodad.asf.ru) и [dotsite](http://www.dotsite.ru). На сайте [oodad.asf](http://oodad.asf.ru), на мой взгляд, хорошо представлена информация общего характера, важная для понимания паттернов. На сайте [dotsite](http://www.dotsite.ru) вы найдете примеры кода, который рекомендую анализировать в режиме отладки. Попробуйте немного побыть архитектором — возьмите любую из разработанных вами логических моделей и попробуйте применить по отношению к ней паттерны ООАП. Затем попытайтесь реализовать часть уточненной модели на любом объектно-ориентированном языке программирования.

Кроме изучения паттернов проектирования, советую разобраться в наиболее перспективных, с одной стороны, и наиболее простых для понимания — с другой, приемах и практиках программирования.

На момент создания книги для ОС Windows это технологии ASP .NET и фреймворк WCF от компании Microsoft. Я рекомендую обязательно ознакомиться со

статьей «Первый взгляд на Windows Communication Foundation» по адресу <http://www.gotdotnet.ru/blogs/sergun/6549/>.

## 5.2.4. Риски, требования к качеству продукта и WBS аналитика

Для каждого разрабатываемого продукта выдвигаются разные требования к качеству. Отвечает за выдвижение таких требований бизнес-дивизион, в частности, менеджер по развитию продукта.

Я рекомендую выдвигать требования к качеству в соответствии с ГОСТ 28195-89. «Оценка качества программных средств». Кроме того, может быть полезен стандарт ISO 9126, который представляет собой набор стандартов и определяет шесть основных категорий качества программного продукта: **Functionality, Reliability, Usability, Efficiency, Maintainability, Portability**.



---

В зависимости от выдвинутых требований к качеству и от характеристик разрабатываемой системы могут меняться методология разработки ПО и состав выполняемых работ (WBS). На сайте книги вы можете скачать дополнительный материал «Выбор ЖЦ-проектов в зависимости от характеристик системы и требований к качеству продукта», который поможет вам с выбором жизненного цикла разработки ПО. В этом разделе мы поговорим о Work Breakdown Structure (WBS) — иерархической структуре работ проекта.

---

Каждая работа в WBS должна быть нацелена на выполнение требований к качеству системы либо на полное или частичное снятие рисков разработки некачественного продукта, например таких, как «неудовлетворенность заказчика» или «противоречие ожиданий заказчика правилам его бизнеса».



---

Риски качества конечного продукта:

- ◆ недостижения бизнес-целей заказчика;
- ◆ неприменимости системы в реальном бизнесе заказчика;
- ◆ неудовлетворенности заказчика;
- ◆ противоречия ожиданий заказчика правилам его бизнеса;
- ◆ неправильного понимания поведения системы;
- ◆ неполного понимания всего поведения системы;
- ◆ нереализации системой всех бизнес-целей заказчика;
- ◆ несоответствия поведения системы ожиданиям заказчика;
- ◆ неправильной реализации поведения системы;
- ◆ нереализации ожидаемых функций;
- ◆ пропущенных вариантов использования;

- ◆ отторжения и сопротивления внедрению системы;
- ◆ реализации требований, не нужных бизнесу;
- ◆ неудовлетворительной работы по скорости, масштабированию и нагрузке;
- ◆ неоптимальной реализации поведения;
- ◆ «Голд платины»;
- ◆ низкого качества тестирования;
- ◆ невозможности приемочных испытаний;
- ◆ сложности в поддержке и сопровождении продукта.

Подумайте, какие работы системного анализа надо выполнить, какие артефакты (модели, требования, трассировки) построить, чтобы полностью или частично снять каждый из перечисленных рисков. Это хорошая самопроверка, чтобы понять, насколько вы владеете методологией системного анализа и разработки требований.

Если вы испытываете затруднение, возможно, вам имеет смысл вернуться к подразделам «Концептуальная модель типов требований», «Моделирование» и «Трассировки» раздела «Профессиональные и специальные навыки» главы «Аналитик».



Некоторые риски снять выполнением аналитических работ невозможно, например такие, как «Риск изменения заказчиком утвержденных требований» или «Риск невыполнения заказчиком своих обязательств». С такими рисками должны справляться менеджеры проекта и продукта.

Теперь проанализируем возможные причины, приведшие к наступлению названных рисков.

<b>Риски качества конечного продукта</b>	<b>Вероятные причины наступления риска</b>
Недостижения бизнес-целей заказчика.	Не выявили и не утвердили с заказчиком все бизнес-требования заказчика.
Неприменимости системы в реальном бизнесе заказчика.	Не выявили и не утвердили с заказчиком все бизнес-правила в бизнесе заказчика.
Неудовлетворенности заказчика.	Не выявили и не утвердили с заказчиком все его ожидания.
Противоречия ожиданий заказчика правилам его бизнеса.	Не выполнили трассировку правил бизнеса на ожидания заказчика.
Неправильного понимания поведения системы.	Не выявили и не утвердили с заказчиком все варианты использования системы.
Неполного понимания всего поведения системы.	Высокий уровень описания выявленных вариантов использования.
Нереализации системой всех бизнес-целей заказчика.	

Риски качества конечного продукта	Вероятные причины наступления риска
<p>Несоответствия поведения системы ожиданиям заказчика.</p> <p>Неправильной реализации поведения системы.</p> <p>Нереализации ожидаемых функций.</p> <p>Пропущенных вариантов использования.</p> <p>Отторжения и сопротивления внедрения системы.</p> <p>Реализации требований, не нужных бизнесу.</p> <p>Неудовлетворительной работы по скорости, масштабированию и нагрузке.</p> <p>Неоптимальной реализации поведения.</p> <p>«Голд платины» — реализации никому не нужных «дорогих» функций.</p> <p>Низкого качества тестирования.</p> <p>Невозможности приемочных испытаний.</p> <p>Сложности в поддержке и сопровождении продукта</p>	<p>Не выполнили трассировку бизнес-требований на варианты использования.</p> <p>Не выполнили трассировку ожиданий заказчика и пользовательских требований на варианты использования.</p> <p>Не выявили и не утвердили в проектной команде все варианты использования системы.</p> <p>Не выявили и не утвердили в проектной команде все функциональные требования к системе.</p> <p>Не выполнили трассировку «от вариантов использования» к функциональным требованиям.</p> <p>Не выявили и не утвердили с заказчиком все требования пользователей системы.</p> <p>Не выполнили трассировку требований бизнеса на требования пользователей.</p> <p>Не выявили и не утвердили в проектной команде нефункциональные требования.</p> <p>Не выполнили трассировку нефункциональных требований на варианты использования.</p> <p>Не выполнили трассировку пользовательских требований на варианты использования.</p> <p>Не выполнили трассировку функциональных требований на нефункциональные.</p> <p>Не выполнили трассировку пользовательских требований на нефункциональные.</p> <p>Не выполнили трассировку бизнес-требований на нефункциональные.</p> <p><i>Не показали реализацию каждого значимого варианта использования системы конкретными классами системы («оторванная от поведения архитектура» может привести к излишней сложности архитектуры, кроме того, существует риск не заметить очевидных «пересечений» одной и той же функциональности по реализации разными классами). Без углубленного анализа поведения системы (построения диаграмм приведения системы, диаграмм активности, диаграмм последовательности, модели анализа, логической модели системы) есть риск несвоевременного выявления нефункциональных требований, важных с точки зрения реализации продукта.</i></p>

(Продолжение)

Риски качества конечного продукта	Вероятные причины наступления риска
	<p>Не выполнили трассировку утвержденных требований пользователя, <i>бизнес-требований, бизнес-правил</i> и ожиданий заказчика на варианты использования.</p> <p>Отсутствие базы для создания тестовых спецификаций — не выявили функциональных требований, нет описанных вариантов использования системы.</p> <p>Нет утвержденных функциональных требований для формальной сдачи-приемки системы заказчику.</p> <p>Отсутствие материалов для написания пользовательской документации, для разработки рекомендаций и обучающих курсов</p>

После определения вероятных причин наступления риска становится понятно, какие работы надо включить в WBS проекта, чтобы максимально снизить риски.

Далее приведен состав WBS, который на 90 % снижает вероятность возникновения перечисленных рисков.

Риски, снятые через управление требованиями, выделяются светлым шрифтом, риски, которые могут быть сняты через управление требованиями, а могут и не быть сняты по решению ответственных лиц, выделяются подчеркиванием.

Риски качества конечного продукта	Состав WBS
<p>Недостижения бизнес-целей заказчика — не выявили и не утвердили с заказчиком все его бизнес-требования.</p> <p>Неприменимости системы в реальном бизнесе заказчика — не выявили и не утвердили с заказчиком все его бизнес-правила.</p> <p>Неудовлетворенности заказчика — не выявили и не утвердили с заказчиком все его ожидания.</p> <p><u>Противоречия ожиданий заказчика правилам его бизнеса — не выполнили трассировку правил бизнеса на ожидания заказчика.</u></p> <p>Неправильного понимания поведения системы — не выявили и не утвердили с заказчиком все варианты использования системы.</p>	<p>WBS включает в себя работы по созданию <b>моделей</b>:</p> <ul style="list-style-type: none"> <li>• концептуальной модели предметной области;</li> <li>• концептуальной модели системы;</li> <li>• Use Case-модели на уровне: название + цель кейса + основной поток + альтернативные потоки (для кейсов, выбранных аналитиком, архитектором, МП и МПП);</li> <li>• диаграммы пригодности для выбранных кейсов с альтернативными потоками;</li> <li>• логической модели системы;</li> <li>• диаграммы активности выбранных кейсов с детальной спецификацией;</li> </ul>

Риски качества конечного продукта	Состав WBS
<p>Неполного понимания всего поведения системы — уровень описания выявленных вариантов использования очень высокий.</p> <p>Нереализации системой всех бизнес-целей заказчика — не выполнили трассировку бизнес-требований на варианты использования.</p> <p>Несоответствия и противоречия поведения системы ожиданиям заказчика — не выполнили трассировку ожиданий заказчика и пользовательских требований на варианты использования.</p> <p>Неправильной реализации поведения системы — не выявили и не утвердили в проектной команде все варианты использования системы.</p> <p>Нереализации ожидаемых функций — не выявили и не утвердили в проектной команде все функциональные требования к системе.</p> <p>Пропущенных вариантов использования — не выполнили трассировку «от вариантов использования» к функциональным требованиям.</p> <p>Отторжения и сопротивления внедрения системы — не выявили и не утвердили с заказчиком все требования пользователей системы.</p> <p>Реализации требований пользователей, не нужных бизнесу, — не выполнили трассировку требований бизнеса на требования пользователей.</p> <p>Неудовлетворительной работы по скорости, масштабированию и нагрузке — не выявили и не утвердили в проектной команде нефункциональные требования.</p> <p>Неоптимальной реализации поведения: <u>не выполнили трассировку нефункциональных требований на варианты использования;</u> не выполнили трассировку пользовательских требований на варианты использования;</p>	<ul style="list-style-type: none"> <li>• диаграммы последовательности для выбранных кейсов с детальной спецификацией;</li> <li>• по совместному решению аналитика, архитектора, МП: диаграммы активности выбранных кейсов с альтернативными потоками;</li> <li>• диаграммы последовательности для выбранных кейсов с альтернативными потоками;</li> <li>• модели данных;</li> </ul> <p><b>проекта требований</b>, интегрированного с моделью, в котором выделяются:</p> <ul style="list-style-type: none"> <li>• бизнес-требования;</li> <li>• бизнес-правила;</li> <li>• ожидания ЗЛ;</li> <li>• пользовательские требования;</li> <li>• ограничения;</li> <li>• допущения;</li> <li>• перечень поддерживаемых ОС;</li> <li>• Use Case;</li> <li>• функциональные требования;</li> <li>• нефункциональные требования;</li> <li>• трассировка бизнес-требований на пользовательские требования;</li> <li>• трассировка бизнес-требований на Use Cases;</li> <li>• трассировка бизнес-правил на Use Cases;</li> <li>• трассировка бизнес-правил на нефункциональные требования;</li> <li>• трассировка пользовательских требований на Use Cases;</li> <li>• трассировка Use Cases на функциональные требования;</li> <li>• трассировка ограничений на нефункциональные требования;</li> <li>• трассировка допущений на функциональные требования;</li> <li>• трассировка допущений на нефункциональные требования;</li> </ul>

(Продолжение)

Риски качества конечного продукта	Состав WBS
<p><u>не выполнили трассировку функциональных требований на нефункциональные;</u> <u>не выполнили трассировку пользовательских требований на нефункциональные требования;</u> <u>не выполнили трассировку бизнес-требований на нефункциональные требования;</u> <i>не показали реализацию каждого значимого варианта использования системы конкретными классами системы («оторванная от поведения архитектура» может привести к излишним усложнениям архитектуры, с другой стороны, к тому, что будут заметны очевидные «пересечения» одной и той же функциональности по реализации разными классами). Без углубленного анализа поведения системы (построения диаграмм пригодности вариантов использования, диаграмм активности, диаграмм последовательности, модели анализа, логической модели системы) есть риск несвоевременного выявления нефункциональных требований, важных с точки зрения реализации продукта.</i></p> <p>«Голд платины» — реализация никому не нужных «дорогих» функций — не выполнили трассировку утвержденных требований пользователя, бизнес-требований, бизнес-правил и ожиданий заказчика на варианты использования.</p> <p>Низкого качества тестирования — отсутствие базы для создания тестовых спецификаций — не выявили функциональные требования, нет описанных вариантов использования системы.</p> <p>Невозможности приемочных испытаний — нет утвержденных функциональных требований для формальной сдачи-приемки системы заказчику.</p> <p>Усложнения поддержки и сопровождения продукта — отсутствие материалов для написания пользовательской документации, разработки рекомендаций и обучающих курсов</p>	<p>по совместному решению аналитика, МП и менеджера продукта выполняются:</p> <ul style="list-style-type: none"> <li>• трассировка бизнес-требований на ожидания ЗЛ;</li> <li>• трассировка ожиданий ЗЛ на Use Cases;</li> <li>• трассировка нефункциональных требований на варианты использования;</li> </ul> <p>по совместному решению аналитика, архитектора, МП выполняются:</p> <ul style="list-style-type: none"> <li>• трассировка бизнес-требований на нефункциональные требования;</li> <li>• трассировка пользовательских требований на нефункциональные требования;</li> <li>• трассировка ожиданий ЗЛ на нефункциональные требования;</li> <li>• трассировка функциональных требований на нефункциональные требования;</li> </ul> <p><b>спецификаций:</b></p> <ul style="list-style-type: none"> <li>• «Словарь терминов и понятий»;</li> <li>• «Бизнес-правила и бизнес-требования»;</li> <li>• «Обзор вариантов использования»;</li> <li>• «Спецификация варианта использования системы» (для кейсов, выбранных аналитиком, архитектором, МП и МПР);</li> <li>• «Спецификация варианта использования системы» (для кейсов, выбранных аналитиком, архитектором, МП и МПР и утвержденных комиссией по контролю за изменениями);</li> <li>• «Требования к системе»;</li> <li>• «Обзор решения»;</li> <li>• «Словарь данных».</li> </ul> <p>Полноценное управление требованиями:</p> <ul style="list-style-type: none"> <li>• статус требований «предложено — одобрено (feasibility) — согласовано — утверждено — реализовано»;</li> <li>• обсуждение в виде дискуссий</li> </ul>

Как видите, можно разработать WBS, которая полностью или частично снимет максимально возможное количество рисков, однако обусловит значительное увеличение стоимости и времени выполнения проекта. Поэтому необходимо управлять рисками и требованиями к качеству.

Менеджер продукта должен выдвигать требования к качеству продукта, затем аналитик анализирует их и формирует список рисков по достижению требуемого качества продукта. Менеджер продукта должен ранжировать риски как критические, приоритетные, некритические, то есть фактически какие-то риски принять, а какие-то отметить как крайне необходимые к устранению.

Менеджер проекта должен управлять рисками — снимать вероятность наступления рисков путем включения соответствующих работ в WBS работ проекта.

Давайте посмотрим, как можно сократить время выполнения аналитических работ.

Время выполнения аналитических работ может быть сокращено ролями «менеджер продукта», «менеджер программы проектов» и «менеджер проекта».

Менеджер продукта и менеджер программы проектов должны стараться не заказывать создание спецификаций требований, если нет серьезной потребности в наличии таких спецификаций в виде документа. Кроме того, менеджер продукта и менеджер программы проектов могут сократить время и стоимость выполнения аналитических работ в проекте путем снижения требований к качеству конкретного продукта, а также за счет принятия определенных рисков.

Менеджер проекта должен планировать и проводить работы по повышению уровня знаний и навыков членов проектной команды по работе с аналитическими артефактами. В идеале у проектной команды должна быть такая квалификация, чтобы было удобнее работать с моделями и проектом требований, чем со спецификациями требований. Менеджер проекта может также сократить время и стоимость выполнения аналитических работ через обеспечение эффективных коммуникаций внутри команды (обсуждения, встречи, процедуры согласования и утверждения, разрешение конфликтных ситуаций).



Обобщим вышесказанное в форме ответственности ролей в управлении требованиями к качеству продукта.

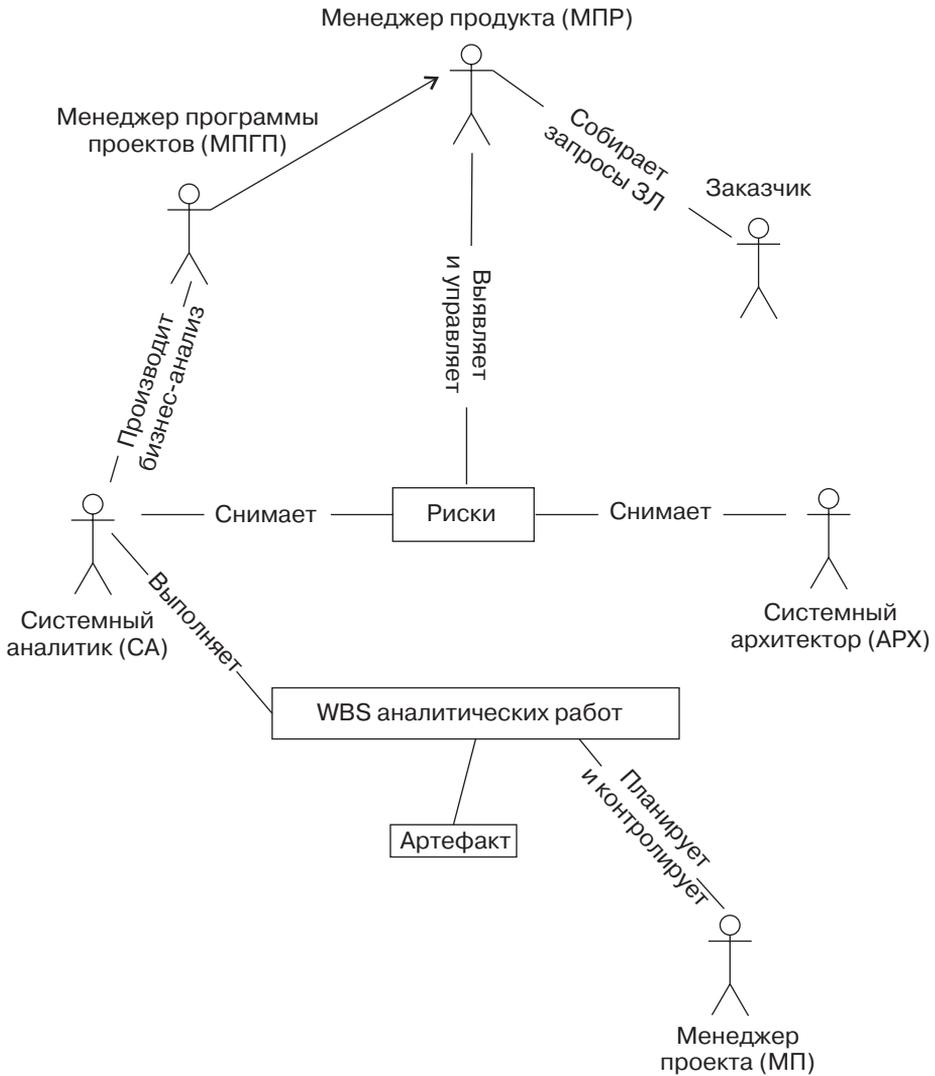
Роль	Ответственность
Менеджер продукта. Менеджер программы проектов	Снизить требования к качеству конкретного продукта. Выявлять и управлять рисками. Заказывать аналитические работы, а не спецификации
Менеджер проекта	Выбрать форму фиксации результатов работ аналитика. Обеспечить эффективные коммуникации внутри команды (обсуждения, встречи, процедуры согласования и утверждения, разрешение конфликтных ситуаций).

Продолжение ⇨

(Продолжение)

Роль	Ответственность
	Планировать и проводить мероприятия по повышению уровня знаний и навыков членов проектной команды по работе с аналитическими артефактами
Системный аналитик	Снять риски через выполнение аналитических работ

Проиллюстрируем сказанное рис. 52.



**Рис. 52.** Ответственность и координация проектных ролей с точки зрения рисков проекта

Здесь важно отметить, что риски достижения качества продукта снимает не только аналитик, но и архитектор. Архитектор выявляет из требований к качеству продукта архитектурные риски и отвечает за то, чтобы максимально снизить их.

В заключение я хочу еще раз акцентировать ваше внимание на важных особенностях обеспечения качества продуктов через требования к качеству продуктов, которые вы должны понимать и учитывать в своей деятельности.

1. Все требования к продукту выявляются из проекта в проект (из релиза в релиз), начиная с самого первого релиза, и в момент создания *i*-го релиза все актуальные требования являются требованиями продукта. Тем не менее когда мы планируем создавать новый продукт или развивать старый, то сразу можем сказать, какие требования качества выдвигаем к продукту. Например, если у нас продукт — система банковских счетов, то мы на 100 % выдвинем требование «Должны быть определены требования к защите и проверке данных», а если продуктом является калькулятор, то такого требования к качеству мы не выдвинем.
2. Следует разделять документацию и цель выполнения какой-то работы по выявлению и анализу информации. Документация — всего лишь способ закрепить результат анализа.
3. В зависимости от наличия каждого из предлагаемых требований качества будут выполняться определенные работы из WBS аналитика — это основной критерий наличия требования качества в списке требований к качеству продукта.
4. Согласование требований с заказчиком — длительный, затяжной процесс, который увеличивает время аналитических работ и сказывается на стоимости продукта. Почти каждое требование к качеству приводит к необходимости согласования с заказчиком.

### 5.2.5. SDLC проекта

Жизненные циклы разработки, представленные в данном разделе, не претендуют на то, чтобы быть единственно возможными и правильными.



Цель этого раздела — показать возможные пути построения методологии системного анализа в различных жизненных циклах. Следует отметить, что жизненные циклы не включают всех необходимых деятельности для выполнения проекта и акцентированы на областях системного анализа и частично архитектуры.

### Водопадный ЖЦ разработки

Ниже приводится выдержка из шаблона ПУТ, описывающего методологию анализа, разработки и управления требованиями в проекте при водопадном цикле разработки (шаблон можно скачать на сайте книги). При употреблении

в этой главе словосочетания «настоящий документ/ шаблон» имеется в виду шаблон ПУТ.

### **Создать план управления требованиями и подготовить инфраструктуру для разработки и управления требованиями (Requirement Management Planning)**

Менеджер проекта с участием системного аналитика проводит работу по созданию плана управления требованиями на основании настоящего шаблона и выполняет адаптацию шаблона модели типов требований проекта к модели типов требований проекта. Аналитик создает проект требований в Requisite PRO (или в любой другой автоматизированной системе управления требованиями) и вносит в план управления документами информацию обо всех артефактах, документах и спецификациях, касающихся анализа и разработки требований.

Данный этап выполняется параллельно с этапами Preliminary Analysis, Business Analysis и Conception Design.

### **Выполнить предварительный анализ (Preliminary Analysis)**

На этом этапе выявляются все заинтересованные в продукте лица, проводятся опрос заинтересованных лиц и анализ всех доступных источников информации (например, Enhancement Requests), которые были предоставлены заинтересованными лицами, выявляемая информация начинает фиксироваться в виде требований в Requisite Pro.

### **Оценить текущее состояние и описать бизнес объекта автоматизации (Business Analysis)**

Системный аналитик с участием бизнес-аналитика детализирует и уточняет запросы заинтересованных лиц (STKR), фиксирует конкретные требования сертифицирующих организаций, стандартов и ГОСТов, выявляет ожидания и допущения, выявляет и детализирует бизнес-требования и бизнес-правила; при необходимости создает бизнес-модель (Business Use Case Model или описание потоков работ — IDEF0) и первую версию модели предметной области.

### **Разработать концепцию автоматизации (Conception Design)**

На данном этапе заканчивается работа по написанию спецификации «Концепция создания и развития продукта» и начинается работа над спецификациями «Концепция системы» и документом «Концепция пользовательского интерфейса».

Этап Requirements Management Planning завершается до окончания этого этапа.

### **Описать желаемое поведение системы (Use Case Modeling)**

Создаются модель вариантов использования (Use Case Model), спецификации «Обзор вариантов использования» и «Спецификация варианта использования системы» (при необходимости).

## Провести анализ ожиданий и поведения системы (Use Case Analysis)

Перед дальнейшим чтением рекомендую ознакомиться с пунктом «Типы требований» подраздела «Концептуальная модель типов требований» раздела «Профессиональные и специальные навыки» главы «Аналитик».

Системный аналитик выявляет новые требования и выполняет трассировки:

- ◆ выявляются новые нефункциональные требования, и строится высокоуровневая иерархия нефункциональных требований (NFR);
- ◆ детализируются пользовательские требования (UREQ);
- ◆ выполняется трассировка бизнес-требований (BVISION) на варианты использования (UC) с целью проверки степени покрытия всего бизнес-видения (BVISION) вариантами использования (UC);
- ◆ осуществляется трассировка выявленного технического видения (TVISION) на варианты использования (UC) с целью проверки степени покрытия всей технической концепции (TVISION) вариантами использования (UC);
- ◆ сценарии вариантов использования анализируются на предмет выявления возможных ограничений на требуемое функционирование системы. Найденные ограничения формулируются как нефункциональные требования (NFR), проводится трассировка требований уровня технической концепции (TVISION) на все выявленные нефункциональные требования с целью проверки степени покрытия всей технической концепции (TVISION) нефункциональными требованиями (NFR);
- ◆ выполняется трассировка выявленных требований пользователей (UREQ) на варианты использования (UC) и нефункциональные требования (NFR) с целью проверки степени покрытия описанием поведения системы и выдвигаемыми нефункциональными требованиями всех требований пользователей;
- ◆ проводится трассировка нефункциональных требований (NFR) на Use Cases (UC) с целью отражения, какие именно Use Cases будут реализовывать какие нефункциональные требования (NFR).

Эти трассировки являются критическими для функциональности разрабатываемой системы, обосновывают правильность и востребованность будущего поведения системы и отражают конкретные потребности заинтересованных лиц, пользователей системы и технические достижения в отрасли, которые будут реализовываться поведением системы, и в каких именно Use Cases они будут реализовываться.

В случае выявления неполной степени покрытия BVISION, TVISION и UREQ требований UC требованиями менеджер проекта совместно с ССВ проекта решает, принимать или нет каждый конкретный риск неудовлетворения бизнес-концепции, технической концепции либо потребностей пользователей.

На этом же этапе появляется либо первичная архитектура системы, реализуемая в виде работающего прототипа системы с отсутствующей или значительно

урезанной функциональностью, либо спецификации (артефакты) в форме, утвержденной и зафиксированной в плане управления документами проекта.

Первичная архитектура должна отражать архитектурные решения на уровне состава и взаимодействия компонентов и слоев.

### **Провести системный анализ (System Analysis)**

Цель данного этапа — провести системный анализ и построить модель анализа и логическую модель системы.

### **Разработать иерархию функциональных требований (Functional Design)**

Функциональные требования выявляются итеративно — сначала параллельно с этапом Use Case Analysis выявляются только высокоуровневые функциональные требования, которые по мере выполнения System Analysis детализируются.

На этом этапе проводится работа по окончательному выявлению и структурированию детальных функциональных требований.

### **Провести реверс-инжиниринг требований из унаследованных спецификаций требований (Legacy SRS Analysis and Requirements Reengineering)**

Системный аналитик по решению ССВ проекта выполняет реверс-инжиниринг требований, перерабатывая унаследованные спецификации требований. Из переработанных спецификаций убирается/помечается как неактуальная (со ссылками на выполняемый проект требований) информация о преобразованных требованиях.

Данный этап может выполняться до окончания проекта, но завершить его рекомендуется до окончательного утверждения тест-планов.

### **Иллюстрация последовательности этапов разработки требований**

На рис. 53 показан общий вид цикла разработки на этапе анализа и разработки требований. Более точное представление можно сформировать после изучения плана управления документами (ПУД), который вы можете скачать на сайте книги.

С подробной информацией об этом жизненном цикле можно ознакомиться на сайте книги в шаблоне ПУТ.

### **Итеративный ЖЦ разработки**

Далее представлен фрагмент ПУТ конкретного производственного проекта. Этот фрагмент дает представление о возможных путях построения итеративного жизненного цикла разработки.

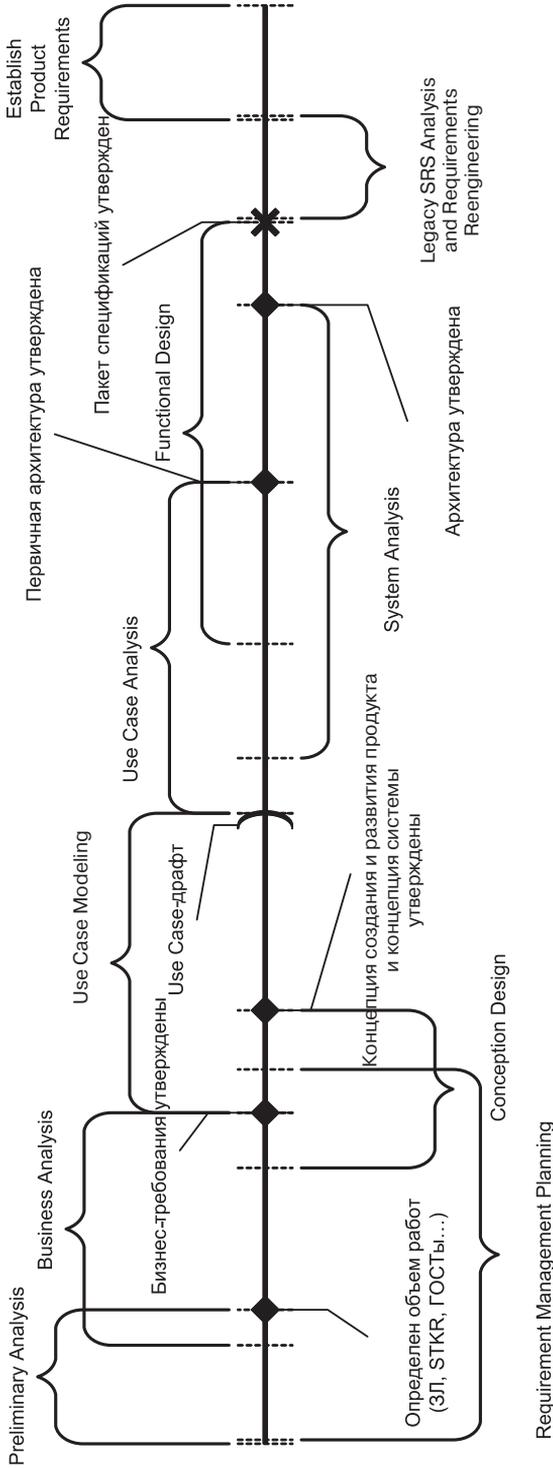


Рис. 53. Иллюстрация последовательности этапов разработки требований

## **Итерация 1. Бизнес-анализ и снятие основных архитектурных рисков**

Цель данной итерации — провести бизнес-анализ, приоритизировать выявленные Use Cases для архитектурно значимых кейсов, проработать архитектурные решения, создать прототип системы, реализующей описанное на итерации поведение системы.

На этой итерации завершается Preliminary Analysis, Requirement Management Planning; выполняются Business Analysis, Conceptual Design; выбираются архитектурно значимые Use Cases, которые детально описываются на этапе Use Case Modeling; выполняется Use Case Analysis, System Analysis и Functional Design, а также проводятся все основные работы по системному анализу, разработке требований и управлению ими.

На рис. 54 представлена графическая иллюстрация этапов первой итерации.

## **Итерация 2. Выявление нового поведения системы через реверс-инжиниринг функциональности ее DOS-версии**

Цель данной итерации — выявить новое поведение системы посредством реверс-инжиниринга функциональности DOS-версии системы.

На этой итерации выполняется Legacy Requirements Reverse Engineering; рассматривается концепция системы на этапе Conceptual Design; выявляются новые Use Cases, из которых, как и на первой итерации, выбираются архитектурно значимые Use Cases с целью детального описания на этапе Use Case Modeling; проводятся Use Case Analysis, System Analysis и Functional Design, а также все основные работы по системному анализу, разработке требований и управлению ими.

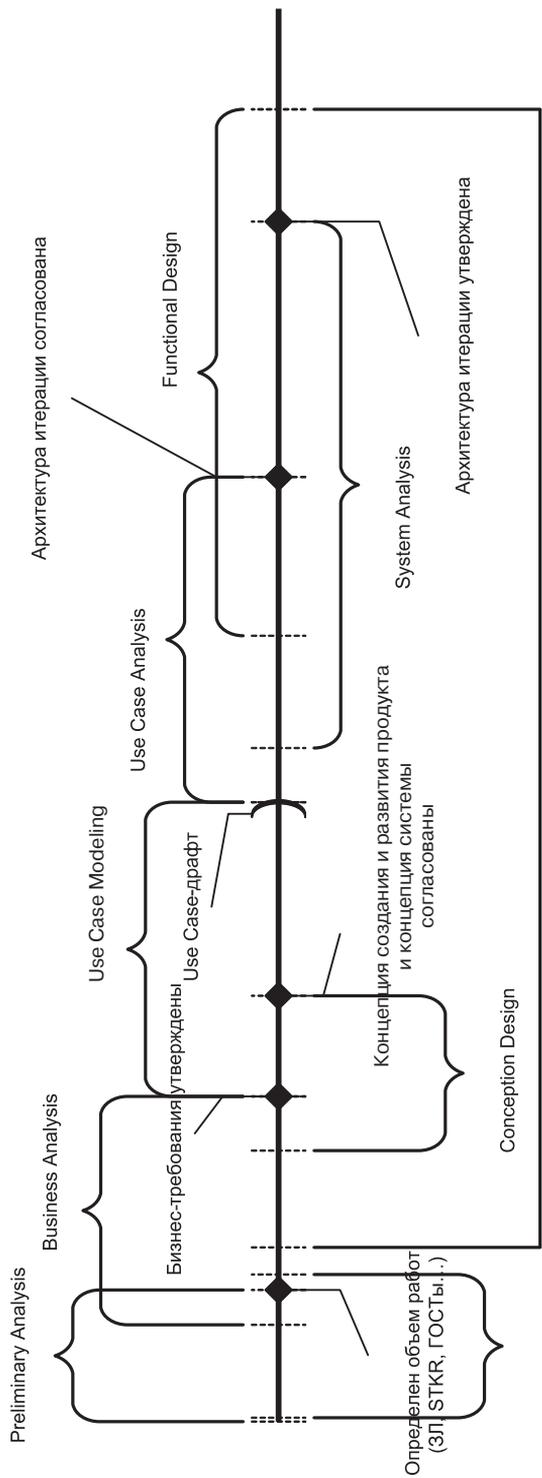
На рис. 55 представлена графическая иллюстрация этапов второй итерации.

## **Итерация 3. Выявление поведения системы, реализованного на уровне высокоуровневого описания функциональности**

Цель данной итерации — выявить новое поведение системы посредством реверс-инжиниринга кода, реализующего функциональность на основании высокоуровневого описания.

На этой итерации выполняется Legacy Requirements Reverse Engineering; рассматривается концепция системы на этапе Conceptual Design; выявляются новые Use Cases, из которых, как и на первой итерации, выбираются архитектурно значимые Use Cases с целью детального описания на этапе Use Case Modelling; проводятся Use Case Analysis, System Analysis и Functional Design, а также все основные работы по системному анализу, разработке требований и управлению ими.

На рис. 56 представлена графическая иллюстрация этапов третьей итерации.



**ИТЕРАЦИЯ 1**

Requirement Management Planning

**Рис. 54.** Иллюстрация этапов первой итерации

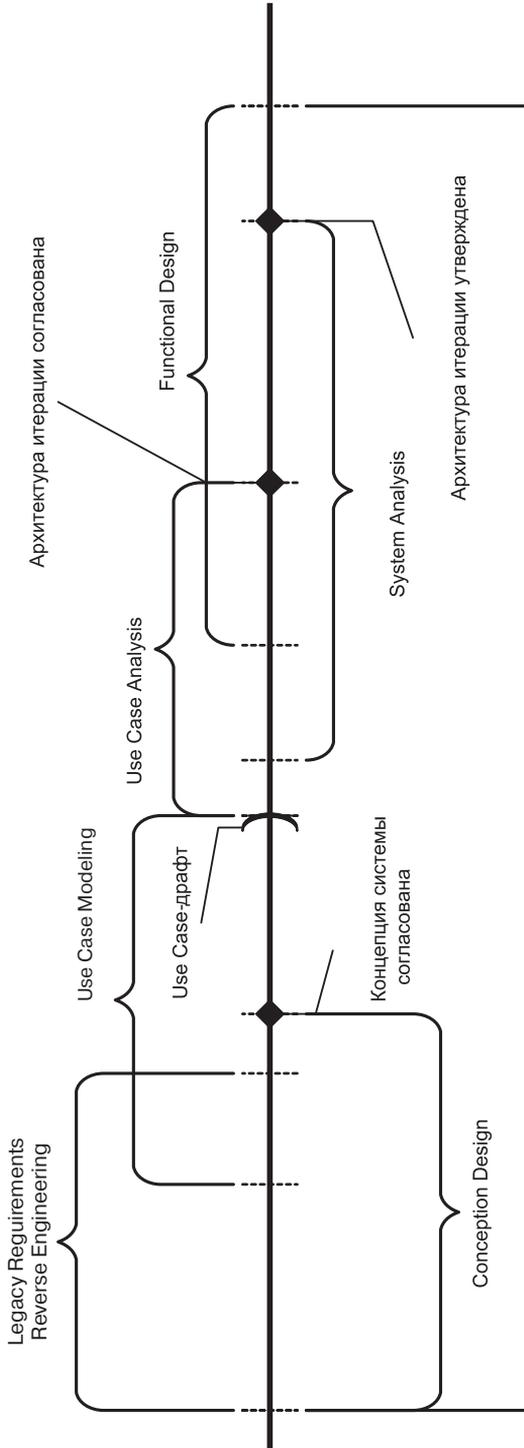
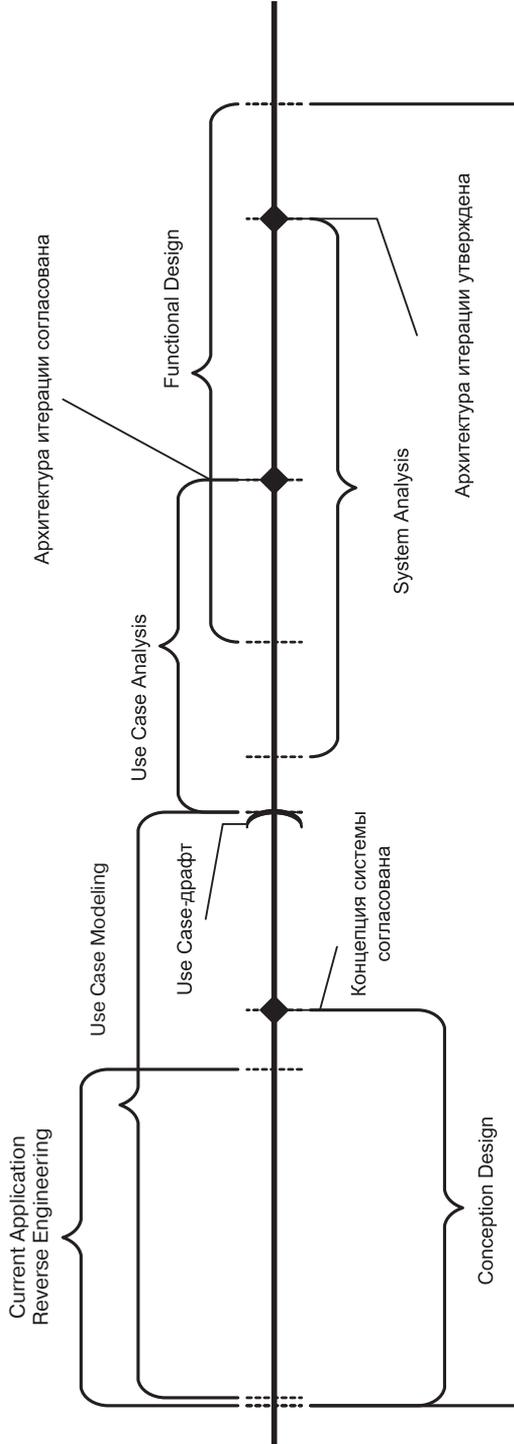


Рис. 55. Иллюстрация этапов второй итерации



### ИТЕРАЦИЯ 3

Рис. 56. Иллюстрация этапов третьей итерации

## **Итерация 4. Выявление и разработка дополнительной функциональности. Детальное GUI-прототипирование**

Цель данной итерации — выявить и разработать принципиально новое поведение системы и дополнительную функциональность (Мастер и т. д.), построить все прототипы графических интерфейсов системы.

На этой итерации выполняется дополнительное интервьюирование заинтересованных лиц **Additional Business Analysis**; **пересматривается концепция системы** на этапе Conceptual Design; выявляются новые Use Cases, из которых, как и на первой итерации, выбираются архитектурно значимые Use Cases с целью детального описания на этапе Use Case Modeling; проводятся Use Case Analysis, System Analysis (**по завершении данного этапа архитектура системы утверждена**) и Functional Design (по окончании этапа все требования к системе утверждены), а также все основные работы по системному анализу, разработке требований и управлению ими.

На рис. 57 представлена графическая иллюстрация этапов четвертой итерации.

## **Иллюстрация последовательности итераций и этапов разработки требований**

На рис. 58 дано общее представление итеративного цикла системного анализа и разработки требований в проекте.

## **Спиралевидная итеративная разработка, управляемая архитектурой**

Отличие этого SDLC от SDLC итеративной/водопадной разработки состоит в соблюдении основных принципов спиралевидной разработки по Boehm [14] (рис. 59). В данном жизненном цикле также используются практики и деятельности методологии RUP [5].

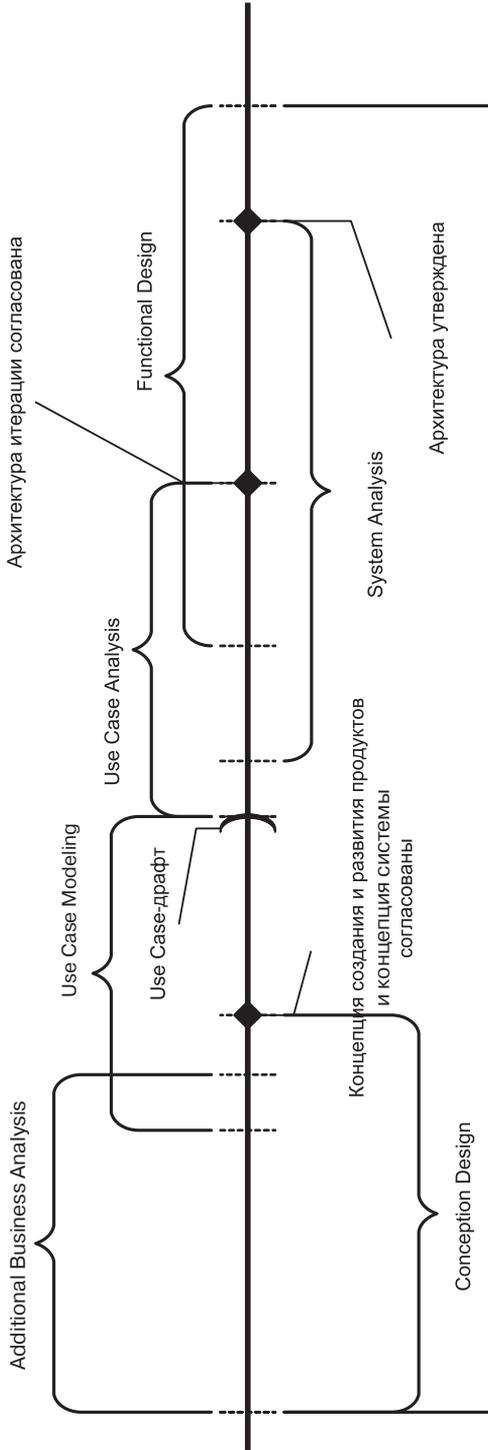
Жизненный цикл спиралевидной итеративной разработки, управляемой архитектурой, акцентирован на раннее и максимальное снятие архитектурных рисков, а также на детальную проработку архитектуры решения на наиболее раннем временном отрезке проекта.

### **Conception Creation**

*На этой итерации особое внимание уделяется определению требований качества, выявлению архитектурно значимых запросов и ожиданий заинтересованных лиц в разработке первичной архитектуры системы и проверке архитектуры на удовлетворение требований по качеству (главным образом по нагрузке и производительности). Результат — архитектурный прототип системы с реализованной основной функциональностью системы.*

### **Architecture Design**

*Цель архитектурной итерации — разработка архитектуры системы. Результат — архитектурный прототип системы с реализованной основной функциональностью системы и 20 % альтернативных потоков Use Cases.*



### ИТЕРАЦИЯ 4

Рис. 57. Иллюстрация этапов четвертой итерации

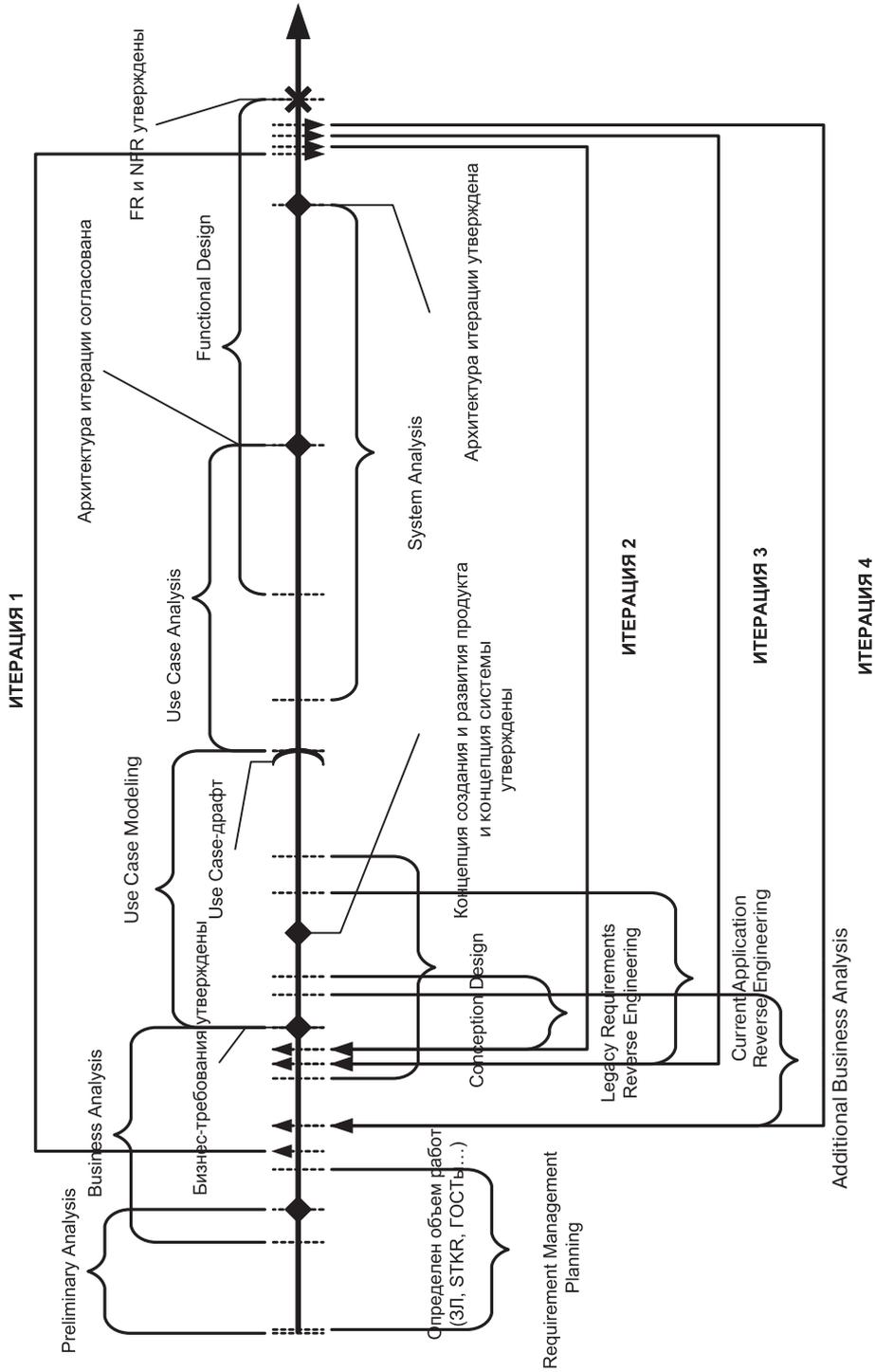


Рис. 58. Принципиальное представление итеративного цикла системного анализа и разработки требований

## Functionality Realization

*Цель функциональной итерации – разработка функциональности системы. Результат – рабочая версия системы с реализованной основной функциональностью системы и 40 % альтернативных потоков Use Cases.*

## Production

*Цель последней итерации – разработка всей оставшейся функциональности системы и стабилизация кода. Результат – полнофункциональная версия системы.*



---

Данный цикл разработки может применяться по отношению как ко всему объему проекта (ко всем Features), завершаясь выпуском релиза продукта, так и к нескольким итерациям в рамках одного релиза, с выпуском промежуточных, некоммерческих, версий продукта.

---

Я рекомендую вам посетить сайт <http://sadd4ru.codeplex.com/>, на котором находятся полезные методологические материалы по разработке архитектуры и детальное описание жизненного цикла Spiral Architecture Driven Development.

Сокращения:

- ◆ PA – Preliminary Analysis;
- ◆ PMP – Project Management Plan;
- ◆ BA – Business Analysis;
- ◆ CD – Conceptual Design;
- ◆ UCM – Use Case Modeling;
- ◆ UCA – Use Case Analysis;
- ◆ FD – Functional Design;
- ◆ AD – Architectural Design;
- ◆ SD – System Design;
- ◆ DEV – Development;
- ◆ Unit\_T – Unit Testing;
- ◆ Stress\_T – Stress Testing;
- ◆ FT – Functional Testing;
- ◆ CAT – Customer Acceptance Test;
- ◆ Integr\_T – Integrational Testing;
- ◆ TM – Test Management;
- ◆ SOW – Statement Of Work;
- ◆ Risk Mng – Risk Management;
- ◆ STKR – Stakeholder;
- ◆ ASSUM – Assumption;
- ◆ UREQ – User Requirements;
- ◆ QAttr – Quality Attributes – нефункциональные требования.



## Спиралевидная итеративная разработка, управляемая функциональностью

Отличие SDLC разработки, управляемой функциональностью (рис. 60), от разработки, управляемой архитектурой, в том, что итерации строятся вокруг последовательной спиралевидной разработки функциональности системы. Архитектура разрабатывается на итерации разработки основной функциональности, на первой итерации — в ее самом общем виде, без существенных трудозатрат на проработку архитектурных запросов, требований и разработку архитектурных механизмов.

Данный ЖЦ более гибкий, он позволяет ускорить разработку ПО, но есть риск построения недостаточно оптимальной архитектуры. В основе этого ЖЦ заложены методология Iconix и отдельные практики методологии Agile.



---

Если скорость разработки не критична для разрабатываемой системы, а другие характеристики не позволяют соотнести разработку с ЖЦ разработки, управляемой архитектурой, то я рекомендую стремиться к разработке продуктов по этому ЖЦ.

---

### Initiation

*На этой итерации особое внимание уделяется определению основных сценариев системы (основных потоков вариантов использования) и концепции системы. Результат — утвержденный заказчиком перечень Use Cases, основных сценариев системы и всех выявленных требований в виде User Stories.*

### Happy Path Definition

*Здесь акцент — на детализации основных сценариев системы (основных потоков вариантов использования), определении всех альтернативных потоков, детальном описании 20 % самых значимых альтернативных потоков Use Cases и создании прототипа GUI системы (см. методологию Iconix [21]). Результат этой итерации — частично рабочая версия системы (достаточная для демонстрации прототипа GUI и функциональности заказчику) с утвержденным заказчиком прототипом пользовательских интерфейсов, реализующая основную функциональность и 20 % значимых альтернативных потоков.*

### Functionality Realization

*На данной итерации требуется провести детализацию оставшихся альтернативных сценариев системы (альтернативных потоков вариантов использования), детальное описание следующих 20 % самых значимых альтернативных потоков Use Cases, создание статической и динамической объектной моделей системы (см. методологию Iconix [21]). Результат — рабочая версия системы с утвержденным заказчиком прототипом пользовательских интерфейсов, реализующая всю основную функциональность и 40 % значимых альтернативных потоков.*



## Production

На этой итерации акцент делается на детализации оставшихся альтернативных сценариев системы (альтернативных потоков вариантов использования), детальном описании последних 60 % альтернативных потоков Use Cases и создании финальной статической и динамической объектной модели системы (см. методологию Icopix [21]), а также планировании следующего релиза (версии продукта), документировании продукта и подготовке продукта к дистрибуции. Результат – полнофункциональная рабочая версия системы.

Сокращения:

- ◆ PA – Preliminary Analysis;
- ◆ PMP – Project Management Plan;
- ◆ BA – Business Analysis;
- ◆ CD – Conceptual Design;
- ◆ UCM – Use Case Modeling;
- ◆ UCA – Use Case Analysis;
- ◆ FD – Functional Design;
- ◆ AD – Architectural Design;
- ◆ SD – System Design;
- ◆ DEV – Development;
- ◆ Unit\_T – Unit Testing;
- ◆ Stress\_T – Stress Testing;
- ◆ FT – Functional Testing;
- ◆ CAT – Customer Acceptance Test;
- ◆ Integr\_T – Integrational Testing;
- ◆ TM – Test Management;
- ◆ SOW – Statement Of Work;
- ◆ Risk Mng – Risk Management;
- ◆ STKR – Stakeholder;
- ◆ ASSUM – Assumption;
- ◆ UREQ – User Requirements;
- ◆ QAtr – Quality Atributes – нефункциональные требования.



---

Данный цикл разработки может применяться по отношению как ко всему объему проекта (ко всем Features), завершаясь выпуском релиза продукта, так и к нескольким итерациям в рамках одного релиза, с выпуском промежуточных, некоммерческих, версий продукта.

---

## План управления документами

План управления документами регламентирует управление поставками и степень готовности документов по завершении этапов производства ПО. Он содержит в себе реестр артефактов и документов, которые должны быть в состоянии Draft или Release к определенным моментам жизненного цикла.



План управления документами является стержнем коммуникаций в проекте и должен согласовываться всей проектной командой.



Структура этого документа состоит из трех блоков.

Идентифицирующий блок информации:

Номер артефакта	Тип артефакта	Категория	Вид артефакта/ документа	Расположение документа	Описание артефакта/ документа
-----------------	---------------	-----------	--------------------------	------------------------	-------------------------------

Далее идет блок, представляющий фазы (этапы) жизненного цикла разработки:

PA	BA	CD	UCM	UCA	SA	FD	TD	D	DEV
----	----	----	-----	-----	----	----	----	---	-----

Последний блок непосредственно регламентирует работу со спецификациями и проектными документами:

Ответственный	Состояние	Плановая дата завершения	Список проверяющих (reviewers)	Список согласователей	Список утверждающих лиц
---------------	-----------	--------------------------	--------------------------------	-----------------------	-------------------------

Здесь **тип артефакта**:

- ◆ документ;
- ◆ модель;
- ◆ проект требования;
- ◆ прототип;
- ◆ спецификация;

**категория**:

- ◆ проектное управление;
- ◆ аналитика;
- ◆ архитектура;
- ◆ GUI-дизайн;
- ◆ тестирование;

**вид артефакта/документа**:

- ◆ заявка на создание продукта/услуги;
- ◆ устав проекта;

- ♦ бизнес-требования и бизнес-правила;
- ♦ концептуальная модель системы (Conceptual Model) и т. д.

#### Описание артефакта/документа

Краткое описание назначения и состава артефакта, например, для спецификации бизнес-требования и бизнес-правил: «Документ содержит описание всех выявленных в процессе сбора и анализа бизнес-требований и бизнес-правил. В документе описывается, каким образом разрабатываемая система связана с достижением бизнес-целей и что она должна для этого делать» (рис. 61).

В Microsoft Excel «автофильтр» позволяет получать различные представления информации ПУД. Наиболее полезное — срез проектных артефактов и их состояний к окончанию каждой фазы проекта. Полезные представления можно сформировать и по состоянию артефактов, по ответственным, по списку согласующих и утверждающих лиц.

Например, применив фильтр «показывать только значения RELEASE» в столбце CD, мы увидим, какие проектные артефакты должны быть закончены и утверждены к окончанию фазы Conceptual Design (рис. 62).



Четкий, хорошо продуманный план управления коммуникациями, который является частью плана управления проектом, разрабатывается менеджером проекта после формирования ПУД. Очень важно определить «правильных» проверяющих, согласующих и утверждающих лиц для каждого типа проектного документа. Ошибки в этом вопросе отрицательно скажутся на качестве конечного продукта.

Несмотря на то что по объему этот раздел получился менее объемный, чем остальные, вы без труда убедитесь, что ПУД содержит в себе огромное количество важнейшей информации, скачав шаблон ПУД на сайте книги.

ПУТ и ПУД — два самых главных управленческих артефакта в проекте с точки зрения анализа и разработки требований.

## 5.2.6. SDLC продукта: важность управления требованиями и основные методики инженерии требований к продуктам



Ниже приведены выдержки из статьи Кристофера Эберта Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques [20], директора компании Alcatel по исследованиям и разработкам, члена Alcatel Technical Academy, сертифицированного преподавателя CMMI. Компания Alcatel потратила много сил и средств на проведение всестороннего серьезного анализа по теме этого раздела и поделилась своим опытом, который я в упрощенном и немного переработанном виде и привожу здесь, так как считаю его действительно полезным.

## План управления документами/артефактами

№ артефакта	Тип артефакта	Категория	Тип артефакта / документа	Расположение документа	Описание артефакта / документа	РА	ВА	CD
11	Спецификация	Аналитика	Концепция системы		Документ описывает основные принципы проектного решения. Документ будет использоваться при разработке архитектуры.			RELEASE
13	Спецификация	Дизайн GUI	Концепция пользовательского интерфейса		Документ описывает общие принципы построения пользовательского интерфейса			Draft
14	Спецификация	Дизайн GUI	Спецификация прототипов пользовательского интерфейса		Документ описывает схемы взаимодействия интерфейсных форм и прототипы интерфейсных форм			Draft
16	Прототип	Архитектура и реализация	Первичная архитектура системы		Это работающий прототип с полностью отсутствующей или значительно урезанной функциональностью, подтверждающий архитектуру реализации на уровне состава и взаимодействия компонент и слоев			
17	Спецификация	Архитектура и реализация	Software Architecture Document. Обзор архитектуры		Документ описывает архитектуру Системы. В данном документе приводятся ссылки на Logical Model, Component Model, Implementation Model, Deployment Model, Data Model.			

Рис. 61. Внешний вид



Для того чтобы создать успешный продукт, необходимо определить потребности рынка и на их основе сформулировать цели и назначение продукта, которые затем реализуются в соответствии с четкими принципами управления проектом. Управление продуктом — это контроль продукта с момента его замысла до выпуска с целью добиться максимальной пользы для компании. Требования — своего рода строительные блоки, объединяющие различные этапы жизненного цикла продукта. Под жизненным циклом продукта мы понимаем совокупность всех операций, необходимых для того, чтобы описать, разработать, реализовать, скомплектовать, использовать, обслужить и прекратить выпуск продукта и его соответствующих вариантов. Тем не менее, согласно отчету 2003 CHAOS, лишь 52 % первоначально установленных требований к продукту находят свое отражение в его окончательной версии [10].

Согласно традиционному эмпирическому правилу определения требований, следует рассчитывать на то, что каждый месяц будет меняться от 1 до 3 % всех требований. Таким образом, в проекте, запланированном на два года, меня-

	UCM	UCA	SA	DEV	Ответственный	Состояние	Плановая дата завершения	Список проверяющих (reviewers)	Список согласователей	Список утверждающих
ASE					Архитектор			Системный аналитик, менеджер проекта	Менеджер программ проекта, менеджер проекта, системный аналитик, тест-менеджер	Системный архитектор, менеджер программ проекта, менеджер продукта
RELEASE					Проектировщик пользовательских интерфейсов			Системный аналитик, менеджер проекта	Менеджер программ проекта, менеджер проекта, системный аналитик, тест-менеджер	ССВ проекта
Draft	RELEASE				Системный аналитик			Системный аналитик, проектировщик пользовательских интерфейсов, системный архитектор	Менеджер проекта, проектировщик пользовательских интерфейсов, тест-менеджер	Менеджер программ проекта, Проектировщик пользовательских интерфейсов, тест-менеджер
		RELEASE			Архитектор			Системный аналитик	Менеджер проекта	Системный аналитик, Системный дизайнер, Менеджер программ проектов
		Draft	RELEASE		Архитектор			Системный аналитик	Менеджер проекта	Системный аналитик, Системный дизайнер, Менеджер программ проектов

плана управления документами

ется (добавляется, удаляется или модифицируется) свыше 30 % требований. Поскольку подобные изменения почти всегда приводят к переносу сроков выполнения проекта, многие подрядчики и клиенты предпочитают проекты, рассчитанные не более чем на год. Исследование показало, что такие проекты выполняются лучше [10], но сокращение длительности проекта и выбор подхода, предусматривающего итерационную работу, не всегда решают проблему.

Для того чтобы сократить количество изменяемых требований, необходимо эффективное управление жизненным циклом продукта, которое предусматривает анализ и совместную работу над продуктом в течение всего времени его существования и позволяет тем самым добиться наибольшей пользы для предприятия, его акционеров и клиентов. На рис. 63 показан упрощенный жизненный цикл продукта, которого придерживаются Alcatel и многие другие компании. Как правило, жизненный цикл продуктов соответствует стандартам IEEE 12207 или IEEE 1074, которые описывают процесс контроля (или анализа), выполняемого между основными этапами [11].

## План управления документами

№ артефакта	Тип артефакта	Категория	Тип артефакта / документа	PA	BA	CD	UCM	UCA	S
5	Документ	Проектное управление	Work Breakdown Structure. Иерархическая структура работ.	Draft	Draft	RELEASE			
6	Модель	Аналитика	Модель типов требований	Draft	Draft	RELEASE			
7	Документ	Проектное управление	Requirement Management Plan. План управления требованиями.	Draft	Draft	RELEASE			
8	Документ	Проектное управление	Analytical Work Breakdown Structure. Иерархическая структура аналитических работ.	Draft	Draft	RELEASE			
9	Спецификация	Архитектура и реализация	Концепция системы	Draft	Draft	RELEASE			
10	Спецификация	Аналитика	Концепция создания и развития продукта		Draft	RELEASE			
11	Документ	Проектное управление	Project Scope Statement. Описание содержания проекта.			RELEASE			

Рис. 62. План управления документами с фильтром

Менеджер по продукту может остановить разработку продукта в любой из контрольных точек жизненного цикла.

Несмотря на то что в теории и практике управления проектами и инженерии требований (**Requirement Engineering – RE**) особое внимание уделяется основным процессам, никаких серьезных исследований на предварительном этапе в них не предусматривается, хотя предварительные процессы входят в состав RE.

Зачастую это объясняется сложностью таких процессов (вследствие их гетерогенного характера, отсутствия четкого разделения ответственности за выполнение этих процессов, неясности их состава и непонятного влияния на доходы акционеров).

В лучшем случае предварительные операции упоминаются в процессе формулирования требований. Больше всего серьезной реализации предварительных процессов мешает тот факт, что руководители компаний отвечают за прибыли

А	FD	TD	D	DEV	Ответственный	Состояние	Список проверяющих (reviewers)	Список согласователей	Список утверждающих лиц
					Менеджер проекта		Бизнес-аналитик, системный аналитик, системный архитектор, тест-менеджер	Бизнес-аналитик, системный аналитик, системный архитектор, тест-менеджер	ССВ проекта
					Системный аналитик		Системный аналитик	Менеджер проекта, Системный	не требуется утверждение
					Системный аналитик		Менеджер проекта, системный архитектор, системный аналитик	Менеджер проекта, Системный архитектор,	ССВ проекта
					Менеджер проекта		Бизнес-аналитик, системный аналитик, системный архитектор, тест-	Бизнес-аналитик, системный аналитик, системный	ССВ проекта
					Архитектор		Системный аналитик, менеджер проекта	Бизнес-аналитик, менеджер проекта, системный	Системный архитектор, Бизнес-аналитик, менеджер
					Бизнес-аналитик		Менеджер проекта, системный архитектор, системный аналитик	Менеджер проекта, системный архитектор, системный аналитик	ССВ проекта (менеджер проекта, Бизнес-аналитик, менеджер продукта, системный архитектор,
					Менеджер проекта		Бизнес-аналитик, заинтересованные лица проекта	Менеджер проекта, Основные заинтересованные	Заказчик проекта, Спонсор проекта

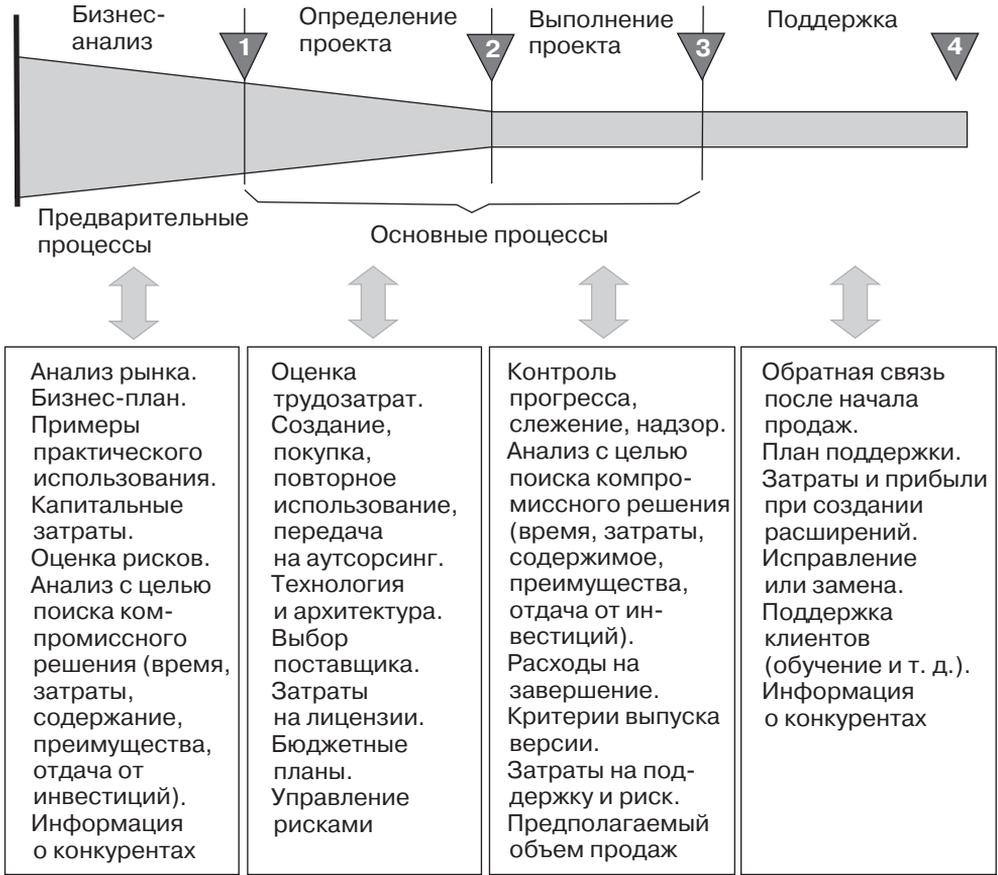
готовых документов к вехе Conceptual Design

и убытки в краткосрочной перспективе. Это вынуждает их сосредоточиваться на результатах текущего квартала (на выполняемых в данный момент проектах и контрактах), а не на будущих продуктах и платформах.

Компания Alcatel провела исследования своей продуктовой линейки и установила, что в среднем после старта проектов было изменено около 73 % требований. Треть изменений были техническими (например, невыполнимая спецификация), остальные обусловлены коммерческими причинами. Как правило, эти изменения предусматривали удаление или замену требований, но, поскольку требования имеют разное влияние и предусматривают разные трудозатраты, замена одного другим не всегда позволяла снизить трудозатраты на реализацию всего проекта в целом.

В результате компания Alcatel пришла к важным выводам:

- ◆ RE необходимо начинать как можно раньше и при управлении проектами следует соотносить текущий проект с портфелем проектов в целом;



**Рис. 63.** Упрощенный жизненный цикл продукта с четырьмя точками принятия решения и предварительными (слева) и основными (справа) процессами

- ◆ в проекте должны принимать участие все заинтересованные лица, причем их нужно наделить соответствующими полномочиями;
- ◆ поддержка жизненного цикла продукта должна быть обязательной для всех проектов, вне зависимости от их вида, размера или назначения;
- ◆ информация обо всех элементах портфеля проектов должна быть легкодоступна.

На основе этого анализа были сформулированы четыре методики, имеющие ключевое значение для эффективного управления жизненным циклом продукта. Кратко эти методики можно сформулировать следующим образом:

- ◆ создание эффективной управляющей группы для каждой версии продукта;
- ◆ концентрация на анализе в контрольных точках на этапе предварительных процессов как основа управления жизненным циклом продукта;
- ◆ оценка требований с различных точек зрения;

- ◆ взаимосвязь состава всего портфеля проектов и реализации отдельной версии продукта.

Рассмотрим их более подробно.

## **Создание эффективной управляющей группы для каждой версии продукта**

Менеджер по продукту должен учитывать долгосрочные бизнес-цели, выходящие за рамки одного продукта. Он определяет, что и как делать, отвечает за бизнес-успех всего портфеля продуктов, одобряет план выпуска продуктов и состав портфеля, а также устанавливает, что и как следует менять. Несет ответственность за всю цепочку ценностей продукта на протяжении его жизненного цикла. Главные вопросы для него: что мы оставляем, что развиваем и от чего отказываемся?

Менеджер по маркетингу продукта определяет, как продавать продукт или сервис. Он отвечает за успех на рынке и удовлетворенность клиентов. Четко представляет, каковы требования пользователей, тенденции на рынке, перспективы продаж и действия конкурентов. Сообщает о преимуществах продукта сотрудникам отдела продаж и клиентам. Составляет план продаж и отвечает за его выполнение. Для него основной вопрос заключается в том, на какие рынки ориентируется компания.

Менеджер проекта определяет, как лучше всего выполнить проект или контракт. Он гарантирует, что проект реализуется так, как задумано, отвечает за успех проекта в компании и у пользователей, руководит планом проекта и его выполнением. Его главный вопрос: как сделать так, чтобы все это было реализовано в заданный срок?

Архитектор продукта отвечает за техническую возможность реализации всех требуемых изменений в рамках выпуска версии продукта, за выбор правильных и современных технологий, удовлетворяющих всем требованиям по качеству новой версии системы.

Все вместе члены управляющей команды руководят проектом, оценивают затраты и прибыль от портфеля и отвечают за его общий успех.

## **Концентрация на анализе в контрольных точках**

Анализ в контрольных точках — это, по сути, средство обмена информацией и управления рисками. В процессе каждого такого анализа состояние проекта оценивается на основе predetermined критериев и принимается решение о том, стоит ли и как его продолжать. Основная точка отсчета — портфель, описывающий все продукты в компании и их рынки, соответствующие инвестиции.

Каждый продукт должен иметь список возможностей, запланированных на несколько следующих версий, — план выпуска продукта (указываются назначение, рынок версий продукта).

Исходя из плана выпуска продукта, управляющая группа должна составить технологический план выпуска (указываются архитектура и технология продукта), что позволяет выбирать поставщиков или создавать партнерские связи. Она же составляет отдельные планы выпуска версий и проектов, которые имеют срок выполнения, как правило, от нескольких месяцев до года.

Данный принцип реализуется как чек-листы контрольных точек проекта, выделенные заранее этапы в ЖЦ разработки ПО и как рекомендации по использованию статус-митингов проекта в качестве инструмента для обсуждения существующих и выявления новых рисков.

## Оценка требования с разных точек зрения

Требования должны оценивать все члены управляющей группы, чтобы тем самым гарантировать, что ситуация изучена со всех сторон.

Группа должна обсуждать каждое требование и определять конкретные практические ситуации, в которых оно играет важную роль, это необходимо для корректного внесения изменений и установки приоритетов.

Зачастую исходя из ошибочной трактовки требований рынка формируется неверный состав проекта, что вынуждает постоянно менять требования. Анализ влияний, таким образом, базируется на требованиях [12], а также на определении приоритетов и управлении портфелем — динамическом процессе принятия решения, направленном на создание правильного сочетания продуктов и выбор правильных проектов для реализации данной стратегии [13].

При оценке следует учитывать несколько параметров. Что собой представляют требования? Как они соотносятся с рынками и как коррелируют друг с другом? На что они влияют? Каким рынкам необходим продукт, кто выдвигает требования и по какой причине? Насколько эти требования обязательны для продукта или они просто достались в наследство от базового подхода и, возможно, сейчас уже устарели? Для того чтобы получить ответы на эти вопросы, управляющая группа должна оформлять требования в виде структурированного и упорядоченного документа. При описании требований в этом документе необходимо указывать и технические, и бизнес-оценки.

Кроме того, члены группы должны анализировать поступающие требования с учетом всего портфеля продуктов и перспективных планов развития компании, чтобы сопоставить максимальную пользу, которую может принести данное требование, с предельными затратами на его реализацию. Такой подход позволит избежать некорректного отображения представлений о рынке в расплывчатые требования.

## Взаимосвязь состава портфеля и требований к продуктам

Необходимо разрабатывать план проекта, который непосредственно привязан к требованиям продукта. Если план проекта или состав (требования) продукта

меняется, вся управляющая группа должна одобрить изменения и синхронизировать их с остальной частью проекта. Требования и данные о других, связанных с текущим продуктом решениях, а также информация о проекте должны быть доступны интерактивно.

Полное представление о портфеле в программных проектах означает, что заинтересованные лица могут получать доступ к информации о проекте и постоянно оценивать все проекты во всей их совокупности.

Если требования, ограничения или предположения меняются, управляющая группа должна соответствующим образом скорректировать оценки. Качество данных и списки требований — основа для принятия решений.

Данный принцип реализуется через процедуру управления изменениями, механизмы уведомлений и согласований (обсуждений) артефактов, а также поддержки дискуссий в системе управления требованиями.

Кратко рассмотрим, как эти принципы могут применяться при планировании развития и разработки продуктов в системе управления проектами компании.

## Планирование развития продуктов

В соответствии с принципом концентрации на анализе в контрольных точках жизненного цикла продуктов, «основная точка отсчета — это портфель, описывающий все продукты в компании и их рынки, а также соответствующие инвестиции. Каждый продукт должен иметь список возможностей, запланированных на несколько следующих версий. В этом списке указываются назначение, рынок, архитектура и технология продукта».



Данный принцип лежит в основе трехуровневого планирования, суть которого — в постепенном движении от стратегических целей в областях назначения, рынка, архитектуры и технологии продукта, через тактическое планирование и формирование бизнес-видения, технологического видения и неделимых подмножеств коммерчески выгодных функциональностей продуктов к портфелю проектов и операционному плану, включающему в себя усиление возможностей компании для достижения требуемых конкурентных преимуществ.

## Стратегическое планирование

Раз в несколько месяцев менеджер по маркетингу продукта управляет созданием маркетинговой стратегии компании. Горизонт планирования маркетинговой стратегии — 18 месяцев.

Цель создания маркетинговой стратегии — определить новые ниши рынка, на которые можно выйти с существующими продуктами, или выпустить новые продукты, а также определить нужные рынку новые направления развития уже существующих продуктов.

Собранная информация преобразуется с участием бизнес-аналитика в набор требуемых (желаемых) характеристик продукта (CHR) в заключительном разделе маркетинговой стратегии.



---

На этапе стратегического планирования **Product Group Manager** создает стратегию развития продуктов для каждого продукта, за развитие которого он отвечает.

Цель создания стратегии развития продуктов — определить цели и направления развития продуктов с точки зрения потребностей клиента, а также требований закона или изменившегося бизнеса клиента.

---

Стратегия развития продукта включает в себя бизнес-требования, бизнес-правила и запросы на расширение/создание новой функциональности от клиентов компании. Основа для создания стратегии развития продуктов — история общения с клиентами компании, поддержка продуктов у клиентов, анализ запросов на изменение продуктов от клиентов.

Для выявления и формирования бизнес-требований и бизнес-правил бизнес-аналитик совместно с юристом и менеджером по маркетингу продукта проводит анализ сегмента рынка и бизнесов клиентов, а также законов, регламентирующих бизнес клиентов и собственную деятельность компании. Эта работа выполняется при подготовке стратегии развития продукта. В результате формируется перечень запросов на расширение функциональности (Business Rules (BRULE) and Business Requirements — (BREQ)).

Еще одна важная часть стратегического планирования — создание архитектурной стратегии.



---

Цель архитектурной стратегии — определить цели и направления развития продуктов с точки зрения новых технологий производства ПО, появившихся на рынке.

Архитектурную стратегию создает главный архитектор (**Lead Architect**) компании с участием менеджеров по продуктам и архитекторов продуктов.

При создании архитектурной стратегии проводится анализ существующей архитектуры продуктов на предмет желаемых изменений архитектуры с точки зрения ее совершенствования и/или использования новых технологий разработки ПО.

---

В процессе создания все эти стратегии итеративно согласуются между собой, цели на первые полгода максимально детализируются.

## Тактическое планирование

На этапе тактического планирования создаются/актуализируются концепция создания и развития продукта менеджером продукта и концепция развития архитектуры продукта архитектором продукта.

Эти концепции взаимно влияют друг на друга, разрабатываются параллельно и итеративно, взаимно уточняя и корректируя содержание, предложения по планированию и варианты построения тактики для достижения стратегических целей.

Концепция создания и развития продукта содержит требования типа «бизнес-видение» (BVISION), сформулированные в результате анализа запросов на расширение функциональности от клиентов, изменившихся требований их бизнеса и бизнес-правил, а также на основании результатов маркетингового анализа (выявленных желаемых характеристик системы). Концепция создания и развития продуктов через набор уникальных и неповторяющихся BVISION предлагает варианты по выполнению стратегических целей в маркетинговой стратегии и стратегии развития продуктов.



Концепция развития архитектуры компонентов содержит TVISION, сформулированные в результате анализа технологических трендов и обработанных запросов на создание и развитие продуктов. Кроме того, она включает предложения: как и когда реализовывать TVISION, **какие BVISION связаны с реализацией** этих TVISION и к чему приведет реализация TVISION с точки зрения выполнения стратегических целей в архитектурной стратегии, то есть оптимального развития архитектуры всех компонентов, из которых состоят продукты.

Концепция создания и развития продукта также содержит Feature [Request]s, которые служат основой для дальнейшего планирования и увязывают BVISION и TVISION в единое целое.

В результате тактического планирования создан следующий набор артефактов:

- ◆ концепция создания и развития продукта (с набором Features для реализации);
- ◆ концепция развития архитектуры продукта.

Тактические мероприятия на первые полгода максимально детализируются, на вторые — не детализируются и остаются на высоком уровне.

## Операционное планирование (планирование производства и мощностей)

Операционная стратегия служит для согласования спроса и возможностей компании на стратегическом уровне. Со стороны спроса выступают факторы конкурентоспособности компании: качество, скорость, уровень обслуживания, гибкость, стоимость. Со стороны возможностей — мощность, сеть поставок, процессы и технология, развитие организации.

Со стороны спроса операционная стратегия базируется на созданных на этапе тактического планирования концепциях создания и развития продуктов и развития архитектуры продуктов и набора Features как результата тактического планирования.

Со стороны возможностей операционная стратегия описывает потребности реализации Features в ресурсах компании, мощностях, технологиях и методологиях, включает в себя план по привлечению и развитию нужных навыков персонала и план инвестиций в материально-техническую базу.



Операционное планирование балансирует спрос и возможности компании. В результате одни проекты могут быть урезаны по границам, другие расширены, для третьих проектов может потребоваться значительное расширение возможностей компании, и эти мероприятия также должны быть приняты во внимание исходя из желаемых сроков окончания проекта, что, в свою очередь, может привести к изменению границ проекта. В ходе операционного планирования состав Features может измениться, некоторые BVISION и TVISION могут быть пересмотрены и, возможно, даже отменены. Все эти коррективы вносятся только с разрешения продуктового комитета.

Продуктовый комитет — кросс-функциональная команда из специалистов продуктового дивизиона и департамента разработки ПО, отвечающая за утверждение и окончательное формирование портфеля проектов по созданию/развитию продуктов.

В результате итеративного операционного планирования создаются артефакты:

- ◆ операционная стратегия (включая план по привлечению и развитию нужных навыков персонала и план инвестиций в материально-техническую базу);
- ◆ профиль проекта с перечнем Features и целей проекта;
- ◆ портфель проектов, состоящий из перечня профилей проектов и интегральной оценки рентабельности портфеля проектов;
- ◆ первые версии планов-расписаний проектов.

Горизонт операционного планирования скользящий, на 12 месяцев, план пересматривается и при необходимости обновляется каждый раз при изменении любой из концепций тактического уровня.

Мероприятия операционного плана на первые полгода максимально детализируются, на вторые — не детализируются и остаются на высоком уровне. Высокоуровневый план-расписание уточняется по принципу набегающей волны при следующем выполнении операционного планирования.

Через выполнение указанных мероприятий реализуется принцип, описанный в разделе «Оценка требования с разных точек зрения»: «Анализ влияний базируется на требованиях, а также на определении приоритетов и управлении портфелем — динамическом процессе принятия решения, направленном на создание правильного сочетания продуктов и выбор правильных проектов для реализации данной стратегии»; а также принцип, описанный в разделе «...И требований к продуктам»: «Необходимо разработать план проекта, который непосредственно привязан к требованиям. Если план или состав продукта меняются, вся группа должна одобрить изменения и синхронизировать их с остальной частью проекта. Требования, информация о проекте, а также данные о других решениях, связанных с текущим продуктом, должны быть доступны интерактивно».

Самое важное заключается в том, что созданный таким образом портфель проектов балансирует бизнес-потребности и технологические потребности развития продуктов.

## 5.2.7. Основы управления

В заключение раздела «Профессиональные и специальные навыки» рассмотрим простейший, но очень эффективный механизм управления — реестры.



Реестр — это табличное представление информации, созданное для управления чем-либо.

Наиболее важные реестры: реестр работ, реестр открытых вопросов и проблем, реестр рисков, реестр выученных уроков. Исходя из основной цели реестра — управление — каждая запись реестра имеет атрибут «состояние». В остальном состав атрибутов реестров может значительно отличаться друг от друга в различных реестрах и проектах.



Рассмотрим возможную структуру и назначение каждого из перечисленных реестров в виде сводной таблицы.

Название реестра	Назначение реестра	Состав реестра	Комментарий по работе с реестром
Реестр работ	Управление работами аналитической группы, собственными работами	Порядковый номер. Наименование работы. Описание работы. Ответственный. Состояние. Основание авторизации работы. Плановая дата завершения. Фактическая дата окончания. Дата создания. Приоритет. Отчет о ходе выполнения. Номер связанной работы	Описание работы и отчет о ходе выполнения работы — поля, в которых в свободной форме, начиная с более поздней даты, вносятся комментарии по выполнению работы. При занесении нового комментария рекомендуется всегда указывать дату и с красной строки начинать сам комментарий. Это позволит проводить ретроспективный анализ. Данный реестр может вести любой участник рабочей группы, но никто не имеет права из него удалять информацию. Неактуальные работы просто переводятся в статус «прекращена»

Продолжение ⇨

(Продолжение)

Название реестра	Назначение реестра	Состав реестра	Комментарий по работе с реестром
Реестр открытых вопросов и проблем	Управление открытыми вопросами и проблемами, возникающими в ходе разработки	<p>Порядковый номер. Дата возникновения вопроса. Инициатор вопроса. Подсистема. Вопрос. Комментарий. Статус вопроса. Этап проекта</p>	<p>Атрибут «Подсистема» здесь рекомендуется использовать для группировки вопросов. «Этап проекта» – этап проекта для разрешения вопроса. Этот атрибут актуален, только если в проекте выделяются этапы. Данный реестр может вести любой участник рабочей группы. Вопросы и проблемы не удаляются из реестра, а помечаются как «неактуальные»</p>
Реестр рисков	Управление рисками проекта или выполнения каких-то работ	<p>Порядковый номер. Риск (название). Последствия наступления риска. Область риска. Вероятность возникновения риска. Влияние риска. Импакт. План предупреждения. План реагирования. Дата наступления риска. Последствия наступления риска</p>	<p>Здесь под областью риска понимается архитектура, управление, обеспечение качества, требования, коммуникации. «Импакт» – это произведение вероятности возникновения на оценку влияния риска на результат работы/на проект. Оценку вероятности и влияния риска дают члены проектной команды совместно. План реагирования, может быть, единственно возможный план реагирования и предотвращения риска – его принятие, тем не менее такие риски должны выявляться командой и управляться через реестр рисков. Дата наступления риска и последствия наступления риска используются только в том случае, если риск наступил, и служат источником информации для выученных уроков. Данный реестр обычно ведет менеджер проекта</p>

Название реестра	Назначение реестра	Состав реестра	Комментарий по работе с реестром
Реестр выученных уроков	Управление выявленными в ходе проекта проблемами, плохими и хорошими практиками, которые должны стать частью выученных уроков	Порядковый номер. Описание. Тип. Функциональная область. Состояние. Дата занесения. Кто занес	Описание — текстовое описание проблемы или хорошей практики. Тип — проблема/хорошая практика/плохая практика. Здесь под функциональной областью понимается архитектура, управление, обеспечение качества, требования, коммуникации. Состояние: выявлена, обсуждена, включена в ВУ проекта. Данный реестр ведет вся проектная команда

Таковы основные реестры, с которыми вам нужно уметь работать и использовать в проектах. Однако не стоит забывать, что основная цель — не форма (реестр или специальные программы), а прозрачность и управляемость, поэтому в проекте вполне можно применять другие инструменты.

Как видите, реестры просты в понимании и использовании. Чтобы их создать и работать с ними, достаточно обычного инструмента **Microsoft Excel**. Если необходимо обеспечить прозрачность в выполнении работ, управлении рисками и т. д., а в компании нет автоматизированного инструмента для этих целей, я создаю реестр. Главное — сдвинуть дело с мертвой точки и развивать более совершенные инструменты параллельно, при наличии возможностей и инструмента для текущего управления.

### 5.2.8. Минутка расслабления

К вопросу о том, стоит ли бояться изобретать собственные нотации, инструменты и методологии. Посмотрите, как оригинально выкрутилась из сложной ситуации одна проектная команда:

*Работал я как-то в восточноевропейском филиале одного ныне крупного разработчика. Однажды из головной конторы нам передали заказ от некоей фирмы на разработку софта под сенсорные экраны вроде тех, что ныне установлены на платежных терминалах, стоящих чуть ли не на каждом углу. Заказчик оказался то ли с юмором, то ли с придурью, то ли с тем и другим вместе: в списке требований, помимо прочего, значилась «проработка эргономики и юзабилити», причем этот пункт был основополагающим. Вот только сам экран нам не прислали — дескать, новая разработка, проходит испытания, не можем прислать (из головной конторы*

по секрету сообщили, что экран, как вторая Звезда Смерти, «not fully operational»). На наш вопрос «А как мы все это будем тестировать?» представитель головной конторы придал лицу выражение, присущее обычно Стивену Сигалу, и выразительно промолчал.

Вариант тестирования «щелкни мышкой» отпал сразу по ряду причин. Решено было нарисовать кучу схем на листах бумаги и постепенно класть один на другой в целях имитации перехода с одного экрана на другой. С учетом того, что экран должен был устанавливаться на уровне груди, можно было наклеить эти листочки на кульман, но с имитацией перехода с экрана на экран по-прежнему возникали определенные проблемы.

Теперь небольшое лирическое отступление: руководителем нашего филиала был поляк, похожий одновременно на Ламберта из недавней игры «Ведьмак» и Янека из бессмертного сериала «Четыре танкиста и собака». Он был рыж и вспыльчив; и в глаза, и за глаза его все звали Руди, прямо как тот танк.

Этот Руди со своим темпераментом явно не мог вытерпеть методичного перекладывания листочков, поэтому собрал консилиум. То, что родилось в подсобке нашей конторки, издали очень напоминало инсталляции многих современных художников а-ля «я его слепила из того, что было». Из огрызков деревянных штакетин, брусков и досок мы сколотили тумбу, к ней приспособили несколько картонных валиков, через которые пустили целлофановую пищевую пленку. К пленке были приклеены вышеупомянутые листочки с эскизами интерфейса.

Со стороны это выглядело так: живчик Руди, стоя около деревянного нечто, крутил большой картонный «штурвал», вращавший валик, на который наматывалась пленка с листами, изображающими разные стадии работы интерфейса. Чтобы не отматывать валик назад, «главное меню» было нарисовано несколько раз, а дальше менялись лишь последующие стадии.

За время работы мы извели несколько десятков пачек бумаги и рулонов пищевой пленки, опустошив полки всех окрестных магазинчиков и вызвав этим недоумение продавцов. Счет за «производственные расходы» был отправлен в головной офис. Жаль, не удалось увидеть лиц руководства в момент получения платежки «на километр целлофана».

Единственным побочным эффектом нашего ударного капиталистического труда стало то, что некоторые коллеги стали на нас косо поглядывать. Однако заказ был сдан и принят в срок, заказчик остался доволен. Не знаю, насколько сильно изменился интерфейс того софта, что мы сдали, — я ни разу не видел его «вживую», но встречал очень похожие решения. Тем не менее теперь я никогда не ругаюсь на глюкающий софт на терминале с сенсорным экраном. Вполне может оказаться, что именно его клепали четыре поляка, пара немцев и по одному венгру и русскому, не имея этого самого экрана.



## 5.3. Типичные проблемы и вопросы

В этом разделе приведены наиболее типичные проблемы и вопросы, с которыми сталкивается ведущий аналитик.

Вопрос/проблема	Рекомендация/комментарий
Кому нужны результаты работы системного аналитика?	<p>Как ведущему аналитику вам придется активно участвовать в налаживании проектных коммуникаций и процессов взаимодействия.</p> <p>Используйте хорошие практики и инструменты, зарекомендовавшие себя для достижения этих целей, — создавайте ПУТ и ПУД в проектах, согласуйте их с проектной командой.</p> <p>Не создавайте требований и спецификаций, которые не нужны архитектору, тест-менеджеру или заинтересованным лицам или которые не снимают никаких рисков достижения требуемого качества продуктов</p>
Планировать или не планировать?	<p>На то, чтобы создать ПУТ и ПУД, иногда требуется довольно значительное время. Но эти трудозатраты с лихвой окупаются в больших и сложных проектах.</p> <p>Я считаю, что планировать нужно всегда. Другой вопрос, насколько детальным должно быть это планирование. Например, для небольших проектов достаточно не создавать полноценный ПУТ, а провести проблемное совещание по теме разработки и управления требованиями и протокол этого совещания считать ПУТ.</p> <p>Очень хорошо, если в компании существуют типизированные шаблоны этих артефактов, которые выбираются в зависимости от характеристик проектов и адаптируются под нужды каждого конкретного проекта</p>
На эти грабли мы уже наступали!	<p>Проводите выученные уроки в проектах всегда. Используйте инструмент «Реестр выученных уроков».</p> <p>Даже если менеджер проекта не выделяет вам проектное время на выполнение этих работ, старайтесь их выполнять за счет операционной деятельности в отделе.</p> <p>Начальник отдела анализа вряд ли будет возражать против проведения анализа и выученных уроков по анализу и разработке требований в проекте.</p> <p>Постарайтесь приблизить ваш стиль работы к циклу качества Деминга: Plan — Do — Check — Act. Это отличный паттерн для постоянного повышения качества выполнения своей работы</p>

Продолжение ⇨

(Продолжение)

Вопрос/проблема	Рекомендация/комментарий
Предупрежден – значит вооружен!	<p>Старайтесь активно управлять рисками достижения качества продукта с точки зрения анализа, моделирования, разработки и управления требованиями.</p> <p>Доводите риски до проектной команды, менеджеров проекта и продукта. Используйте инструмент «Реестр рисков».</p> <p>Помните, что задача менеджера продукта – принять решение о приоритизации рисков, но управляет рисками в проекте и несет ответственность за успех проекта в целом менеджер проектов</p>
Что дальше с точки зрения Пути аналитика?	<p>На этом этапе вашей карьеры вы можете перейти либо в следующую категорию – «начальник отдела» и остаться на Пути аналитика, либо в область разработки и стать системным архитектором, либо в область управления и постараться стать менеджером проекта или менеджером продукта</p>

## 5.4. Заключение

Мы рассмотрели направления развития личностных навыков, узнали, как проводить реверс-инжиниринг требований, что такое требования к качеству продукта, риски достижения качества и как ими управлять, как может быть построен SDLC проекта, коснулись вопросов важности управления требованиями к продукту в SDLC продукта и поговорили об использовании реестров для обеспечения прозрачности и управляемости.



Эта глава содержала сложный для понимания и очень важный материал. Я считаю, что после приобретения профессиональных и специальных навыков, которые мы здесь обсудили, ваше становление в качестве профессионального аналитика завершено.

Для более глубокого усвоения информации рекомендую вам прочитать все источники (книги и статьи), на которые я ссылался в этой главе, а также ознакомиться с другими материалами, указанными в квалификационной таблице навыков (табл. 9).

**Таблица 9.** Профессиональные и специальные навыки ведущего аналитика

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
Старший аналитик	<p>Все теоретические знания аналитика, а также книги: Данилин А., Слюсаренко А. Архитектура и стратегия. «Инь» и «Янь» информационных технологий предприятия»;</p>	<p>Все навыки аналитика, а также:</p> <ul style="list-style-type: none"> <li>• иметь детальное представление о ЖЦ проекта и продукта;</li> <li>• знать, что такое ПУД, и уметь его создавать;</li> </ul>

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
	<p><i>Boehm Barry W.</i> A Spiral Model of Software Development and Enhancement;</p> <p>A Guide to the Project Management Body of Knowledge. ANSI/PMI.</p> <p>Статья Кристофера Эберта Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques.</p> <p>Рекомендую регулярно посещать <a href="http://msdn.microsoft.com/ru-ru/default.aspx">http://msdn.microsoft.com/ru-ru/default.aspx</a> и <a href="http://www-01.ibm.com/software/success/cssdb.nsf/topstoriesFM?OpenForm&amp;Site=rational&amp;cty=en_us">http://www-01.ibm.com/software/success/cssdb.nsf/topstoriesFM?OpenForm&amp;Site=rational&amp;cty=en_us</a></p>	<ul style="list-style-type: none"> <li>• уметь создавать логическую модель и модель данных;</li> <li>• уметь создавать простой программный код (этот навык необходим для закрепления навыка чтения программного кода, никто так не поймет разработчика, как другой разработчик, и, если вы хотите говорить с разработчиками на одном языке, вам придется освоить программирование в минимальном объеме);</li> <li>• уметь профессионально проводить презентации;</li> <li>• проводить выученные уроки по практикам разработки и управления требованиями;</li> <li>• быть наставником для аналитиков;</li> <li>• уметь предотвращать и разрешать конфликты в проектах;</li> <li>• уметь выявлять риски и управлять ими</li> </ul>

# 6. Начальник отдела анализа

---

---

Это последняя глава книги «Путь аналитика». Вы прошли большой путь в своем профессиональном и карьерном развитии и вряд ли нуждаетесь в каком-то обучении. Здесь я хочу просто поделиться своими мыслями, собственными и чужими лучшими практиками, а также постараюсь представить информацию, которая в свое время очень впечатлила меня самого и которую я взял в свой арсенал. Надеюсь, что она будет полезна и вам.

## 6.1. Личностные навыки

### 6.1.1. Самомотивация

Для вас как для начальника отдела очень важно с самого начала четко определить цели вашей деятельности в компании. Скорее всего, вы будете входить в исполнительный комитет, рабочую группу по совершенствованию процессов. Ваше участие во всех этих разноплановых деятельности будет преследовать определенные цели, которые будут противоречить целям других функциональных руководителей.

Чтобы успешно достигать своих целей, в первую очередь необходимо правильно их формулировать. Для этого существует ряд методик, с помощью которых можно ставить действительно «качественные» цели. Один из самых распространенных и действенных способов — постановка «умных» (от англ. *smart*) целей, то есть целей с использованием S.M.A.R.T.-критериев.

Цели должны быть:

- ◆ **конкретными** (Specific).

Цель должна быть четко сформулирована. Иначе может быть достигнут результат, отличающийся от запланированного;

- ◆ **измеримыми** (Measurable).

Если у цели нет каких-либо измеримых параметров, то будет невозможно определить, достигнут ли результат;

◆ **достижимыми (Achievable).**

Нужно ставить сложные цели, предполагающие значительные усилия, но при этом они должны быть достижимыми. Не следует ставить такие цели, которые приводили бы к увеличению стрессов в вашей жизни или в жизни ваших подчиненных. В идеале S.M.A.R.T.-цели используются в качестве стимула для успешного развития в целом, через цепочку успехов в достижении конкретных целей;

◆ **ориентированными на результат (Result oriented).**

Цели должны характеризоваться исходя из результата, а не выполняемой работы. Так обеспечивается эффективность. Можно поставить себе цель работать на два часа больше, но если не определен конкретный ожидаемый результат такой «цели», то эффективность этого дополнительного времени вызывает большие сомнения;

◆ **имеющими конкретный срок достижения (Timed).**

Любая цель должна быть выполнима в определенном временном измерении.



Крайне важно, чтобы ваши отношения с топ-менеджерами строились на основе S.M.A.R.T.-целей. Я стараюсь строить отношения в ключе «Я не имею права принимать всю ответственность только на себя, так как за бизнес в целом отвечает топ-менеджмент, но я надеюсь на помощь и поддержку остальных топ-менеджеров в определении стратегических целей и утверждении разработанной мной тактики их достижения». При таком подходе к работе вы можете быть уверены, что деятельность вашего подразделения согласована с общей стратегией развития компании и ее бизнеса. А значит, есть вероятность и вашего персонального успеха в этой компании.

В работе начальника отдела неизбежны производственные конфликты. Здесь я порекомендую вам один из уроков К. Фиорины: «Не стоит переживать из-за чьей-то узколобости или заблуждений. Этот мир несправедлив. Эту реальность следует принять и не позволять себе переживать из-за нее. Ваше сердце и нервы принадлежат только вам и никому больше. Берегите их».

Когда вы были лидером и ведущим аналитиком, то были тесно интегрированы в команду. Теперь, находясь на позиции начальника отдела, вам придется скорректировать свою позицию внутри коллектива. Чтобы принимать трудные решения, следует дистанцироваться от него. Однако при этом вы должны оставаться лидером команды и не переставать быть ее частью.



Если хотите, это своего рода «раздвоение сознания». В какой-то момент (например, в процессе «мозговых штурмов») вы выступаете как обычный член команды, только изредка пользуясь полномочиями вашей позиции (например, для принятия окончательного решения или для модерирования). В другое время вы исключительно функциональный менеджер, перед которым стоят задачи и цели и который требует выполнения заданий и поручений от своих подчиненных для того, чтобы эти цели и задачи были достигнуты. Я сторонник прозрачных и честных отношений в команде, во многом это обеспечивается за счет того, что каждый член команды понимает, как его

персональный вклад влияет на достижение целей, стоящих перед отделом в целом. Команда должна осознавать, с какой именно из ваших личностей они говорят в каждый конкретный момент. Во избежание недоразумений рекомендую вам создавать базовые правила отдела, в которых определять основные приемы выполнения работ, обязательные для исполнения всей командой (и вами в том числе). Это поможет команде ощущать и себя как единое целое, и вас как непосредственного участника команды. Более подробно см. в пункте «Базовые правила работы отдела» подраздела «Корпоративная культура» раздела «Профессиональные и специальные навыки».

Формируйте личный бренд. Он начинается с вашего авторитета в компании, где вы работаете. Это не только сотрудники вашего отдела, но и сотрудники отделов, с которыми вы взаимодействуете в вашей производственной деятельности, и ваши коллеги по компании. Я неоднократно советовал «делать добро и бросать его в воду». Если вы выросли в этой компании, вам будет проще, так как определенный авторитет у вас уже есть.



Если же вы новый человек в компании, то рекомендую начать ваш путь с проведения встреч с командой и коллегами из смежных отделов по выявлению существующих проблем и обсуждению принципов и подходов в методологии анализа и разработки требований «как есть» и «как должно быть». Собирайте ожидания, приоритезируйте их, управляйте ими и учитывайте наиболее приоритетные ожидания в первую очередь. Проводите презентации и обучения. Живое общение и управляемые изменения помогут вам сформировать авторитет в производственной среде. Кроме того, рекомендую вам начать выступать на публичных профильных конференциях, писать статьи в специализированные журналы и участвовать в профильных форумах.

В заключение хочу напомнить о том, что ваше сердце, ваши нервы и ваше здоровье принадлежат только вам. Прежде чем принять какое-то решение, оцените все за и против, а затем следуйте зову своего сердца.

## 6.1.2. Лидерство

Лучший лидер — тот, о существовании которого люди почти не знают. Когда работа сделана, они говорят: мы сделали это сами.

*Лао Цзы*

Лидерство — это искусство побуждать людей делать то, что тебе нужно, причем делать это с желанием.

*Дуайт Д. Эйзенхауэр*



По моему мнению, хороший руководитель — это лидер, профессионал в своей области, владеющий мировыми методологиями и стандартами в своей области и следящий за новыми веяниями, имеющий значительный практи-

ческий опыт, умеющий сформировать свое видение, согласованное с видением компании, увлечь этим видением своих подчиненных, создать команду и добиться достижения поставленных целей.

Я сторонник управления, основанного на лидерстве при четко определенных базовых правилах работы. Считаю важным установление прозрачных и честных отношений с подчиненными и коллегами на основе четко обозначенных целей, оценки трудоемкости работ со стороны подчиненных (при постановке задач) и совместного выявления рисков и управления ими.

В этой главе мы рассмотрим некоторые практики лидерства.

Общекорпоративные ценности — это не что иное, как своеобразные «маячки», определяющие принятые в компании хорошие практики и правила хорошего тона, сигнализирующие сотрудникам, как им поступать, если инструкции на данный конкретный случай еще не придумано. Основные ценности, лежащие в основе деятельности любой компании, — это взаимоуважение, приверженность своим потребителям, командная игра и сотрудничество, доверие к профессионализму коллег, мотивация к инновациям, оценка вклада каждого, а также быстрота действий и мысли. Одна из задач лидера — определять такие ценности своим примером, через практики собственного участия в проектах и коммуникациях, через создание базы знаний и механизмы лучших практик, закрепляя результат в базовых правилах работы.

Дуг Де Карло в своей книге **eXtreme Project Management** определяет экстремальное управление проектами как управление, построенное вокруг того, **как люди думают, что они чувствуют и как общаются между собой**, а управление проектами — как науку и искусство направлять потоки мыслей, эмоций и отношений на достижение значимых результатов. Я считаю, что управление проектами в такой трактовке отлично подходит и для управления вообще, и для управления отделом в частности.



Руководителю проекта придется стать управляющим энергетическим полем проекта и *взять на себя обязанности лидера*, направляющего потоки мыслей, эмоций и отношений на достижение значимых результатов.

Приведу еще несколько выводов из этой книги, связанных с лидерством, к которым я пришел самостоятельно, путем собственных проб и ошибок, и которые целиком и полностью поддерживаю и разделяю.



Слово «проект» содержит в себе некий угнетающий подтекст. Люди будут работать с большим энтузиазмом, если будут знать, что *выполняют определенную миссию*; если они будут рассматривать проект не как «проект», а как причину их действий. Применение на практике этого принципа означает, что *вы должны показать людям, что их работа является частью чего-то большего, дав им четкое представление о целях и средствах*.

*Сделайте людей хозяевами достигнутых результатов.* Люди заботятся о том, что они создают. Вы должны *доверять профессиональным способностям* людей и *дать им возможность повлиять на достижение успеха всего проекта в целом*, включая измерение производительности проекта.

Воспитывайте и развивайте *систему ценностей*, основанную на непоколебимой вере участников проекта в то, что, работая сообща, они обязательно добьются успеха, даже в неблагоприятных изменчивых условиях.

Развивайте *человеческие ценности*:

- ♦ *свободное общение* — действуйте последовательно и открыто, говорите правду о хорошем, плохом и ужасном, не боясь наказания;
- ♦ *главное — люди*: устраните все препятствия и дайте людям возможность качественно выполнять свою работу;
- ♦ *качество жизни* — убедитесь в том, что соблюден необходимый баланс между работой и личной жизнью;
- ♦ *отвага* — действуйте несмотря на свой страх; действуйте наперекор страху, если вы знаете, что поступаете правильно;
- ♦ *прозрачность* — держите все, что можно, на виду у всех: проектные планы, достигнутый прогресс, разработанные продукты, отчеты, ответственность за выполненную работу.

Люди должны обмениваться проектной документацией и иметь постоянный доступ к инструментам управления проектами. Участники проекта должны быть в курсе последних событий.

Общение в реальном времени подразумевает возможность *создания проектной инфраструктуры, которая обеспечит доступ к информации всем участникам проекта*, что позволит ускорить поток мыслей и процесс принятия решений.

Проекты подобны цветам. Если почва отравлена, один или два цветка прорастут, но рано или поздно весь урожай погибнет.



В гибкой организации развивается устойчивая к изменениям, поддерживающая проект культура, которая оказывает содействие нуждам различных проектов, от традиционных до экстремальных.

*Целью устойчивой к изменениям организации является не столько соблюдение временных, целевых и бюджетных требований, сколько создание условий, способствующих получению желаемого результата.*

Во многом это обеспечивается за счет *эффективного лидерства*.

В традиционной модели управления руководитель проекта пытается выжать из людей все, что возможно. В модели управления экстремальным проектом его роль заключается в такой *организации работы, при которой люди смогут достичь большего*.



Вы можете добиться этого путем внедрения соответствующих процессов и практик (например, грамотной организации встреч) и создания продуктивной рабочей среды, которые в совокупности направляют потоки мыслей,

эмоций и отношений на получение значимых результатов и ускоряют весь процесс.

В хорошем экстремальном проекте никто не находится под контролем проекта. Наоборот, все контролируют проект. Лидерство через приверженность вдохновляет. Лидерство через контроль подавляет. К счастью, ньютоновский тип установления контроля редко имеет место в экстремальном управлении проектами. Как правило, вы не сможете оказывать существенного влияния только на основании занимаемой должности в организации. Но как же тогда вы будете управлять группой людей (проектной командой), которые не отчитываются перед вами в привычном смысле этого слова, подчиняются другим руководителям и которым в принципе не нравится словосочетание «управление проектами»?



В мире экстремальных проектов *ваша сила заключена в процессе, который вы используете для стимулирования людей на выполнение работы, а не в руководстве этими людьми напрямую.*



Используйте этот список в качестве шаблона и спросите себя, какие из этих демотиваторов присутствуют в вашем проекте. Вы можете с уверенностью сказать, на какие из этих факторов вы способны повлиять?

Отсутствие уверенности	Отсутствие страсти
<p>Члены команды ничего не умеют.</p> <p>Слишком много людей; постоянные споры.</p> <p>Руководитель сомнительным образом регулирует ход выполнения работ.</p> <p>Нереальные цели.</p> <p>Слишком сложный проект.</p> <p>Проект политически некорректен.</p> <p>Высокая вероятность провала проекта.</p> <p>Наш спонсор — слабак, а не чемпион.</p> <p>Заказчики не испытывают приверженности к проекту.</p> <p>Нет надежды, что поставщики или другие отделы дадут то, что нам нужно.</p> <p>От совещаний мало толку.</p> <p>В ключевых решениях и положениях нет целостности.</p> <p>Слабая система поддержки.</p> <p>Условия работы оставляют желать лучшего</p>	<p>Был, сделал.</p> <p>Работаю по 60 часов в неделю.</p> <p>Не вижу ценностей проекта.</p> <p>Не понимаю сути моего участия в проекте.</p> <p>Непонятно, как осуществляется проект.</p> <p>Это большой риск для моей карьеры.</p> <p>Мне не нравятся члены проектной команды.</p> <p>Слишком много бюрократии.</p> <p>Хорошие начинания рубятся на корню.</p> <p>Я ничего не умею</p>



Непозволительной ошибкой для руководителя любого уровня я считаю тоталитаризм (или шаблонный менеджмент), когда руководитель проекта полагает, что сможет управлять динамикой проекта, заставляя людей заполнять формы отчетности вместо того, чтобы сосредоточиться на раскрытии мотивации, создании инноваций и установлении доверительных отношений, что требует реализации управления, основанного на ценностях и принципах.

К сожалению, таких руководителей пока еще довольно много, хотя, по-хорошему, их топ-менеджерам стоит серьезно задуматься о том, чтобы прекратить сотрудничать с подобными менеджерами.

На сайте [http://www.cecsi.ru/coach/mgmt\\_top.html](http://www.cecsi.ru/coach/mgmt_top.html) вы найдете еще один интересный взгляд на лидерство: концепция С.М.А.Р.Т. (Синергичный + Мобильный + Активный + Рациональный + Творческий) бизнес-лидера и мастера бизнес-синергии.



Основная идея заключается в том, что новой экономике нужны новые менеджеры. Сегодня, в условиях постоянно усложняющихся технологий и бизнес-систем, основной рост достигается не благодаря улучшению каких-то отдельных операций, технологий, процессов или бизнес-единиц, а за счет нахождения возможностей эффективного использования «белых пространств» между бизнес-единицами и построения инновационных синергичных комбинаций.

**Синергия лидерской и менеджерской ролей** заключается в том, что лидер делает правильные вещи, менеджер делает вещи правильно, а вы должны делать правильно правильные вещи.

- ◆ Лидерская роль: вдохновлять, создавать возможности, заряжать людей энергией, вводить перемены и инновации.
- ◆ Менеджерская роль: реализовывать планы и следить за тем, чтобы задания выполнялись вовремя, влезать в детали и заниматься рутинной работой.



Вы должны хорошо понимать основные различия между менеджментом и лидерством и гармонично соединять в себе обе эти роли, чтобы вести свою команду к успеху. Как менеджер вы должны следить за тем, чтобы работа была сделана, а как лидер — заботиться о людях, которые ее выполняют.

Таким образом, чтобы успешно интегрировать лидерские и менеджерские функции, вы должны найти равновесие между выверенным и логическим подходом к организационным процессам (менеджмент) и вдохновляющим, заряжающим сотрудников энергией и проникнутым искренней заботой о людях (лидерство).

В современной динамичной экономике и быстро меняющейся бизнес-среде руководителю уже недостаточно быть просто менеджером. Чтобы достичь более заметных успехов, тактических и стратегических, старайтесь быть одновременно и менеджером, и лидером, синергично переплетая их функции.

Непрерывно заряжайте своих сотрудников энергией. Что в современной бизнес-среде заряжает людей энергией, так это широкие горизонты, возбуждающие новые вызовы и большие возможности. Когда лидеры могут им это дать — люди оживают.

Индивидуальные отношения сотрудников с менеджерами, доверие, уважение и ежедневное внимание со стороны менеджеров являются базисом для создания заряженной энергией рабочей силы. Получение наивысших результатов от сотрудников — прежде всего результат «мягкой» стороны менеджмента: как менеджер обращается с индивидуальными сотрудниками, создает дух соревнования и вдохновляет их на достижение наилучших результатов. Другой стороной медали являются поддержка, ресурсы и руководство, которые менеджеры предоставляют сотрудникам, чтобы сделать их исключительную продуктивность реальностью.

Одна из основ успешной работы команд в современном бизнесе — создание инновационного управления процессами через создание и управление кросс-функциональными командами. На рис. 64 приведена диаграмма с сайта Вадима Котельникова: [http://www.cecsi.ru/coach/process\\_mgmt.html](http://www.cecsi.ru/coach/process_mgmt.html):



Рис. 64. Иновационное управление процессами



**Кросс-функциональная команда** — это группа сотрудников различных функциональных департаментов организации, таких как научные исследования, инжиниринг, маркетинг, финансы, развитие людских ресурсов и управление операциями, сконцентрированных на решении конкретной задачи и работающих как команда ради улучшения координации работы департаментов, системных инноваций, решения общих проблем и создания синергии в бизнесе.

Особенно хорошо эта концепция работает в организациях, которые стараются быть восприимчивыми к инновациям. Но даже если ваша организация не такая, идеи, представленные на рис. 65 ([http://www.cecsi.ru/coach/innovation\\_org.html](http://www.cecsi.ru/coach/innovation_org.html)), все равно заслуживают внимания для внедрения хотя бы на уровне подразделения.



**Рис. 65.** Организация, восприимчивая к инновациям

Интересна также концепция «Дао в бизнесе». Рекомендую вам посмотреть эти материалы самостоятельно. Здесь я приведу несколько высказываний и даосскую медитацию, которая помогает посмотреть на любую ситуацию с двух сторон — весьма полезный навык для начальника отдела. На рис. 66 приводится диаграмма с сайта Вадима Котельникова: [http://www.cecsi.ru/coach/leadership\\_tao.html](http://www.cecsi.ru/coach/leadership_tao.html).

Древняя даосская медитация «Будь в гармонии с реальностью»:

Я закрываю глаза... и все ясно вижу...

Я перестаю пытаться слушать... и слышу правду...

Я безмолвен... и мое сердце поет...

Я не ищу контакта... и нахожу единение...

Я неподвижен... и я двигаюсь вперед...

Я податлив... и мне не нужна сила...

Я покорен... и остаюсь целостным...

Не менее интересны с практической точки зрения и 25 уроков Джека Уэлша.

## Дао лидерства

Незаметное, но очень эффективное лидерство

Лао Цзы



Лао Цзы род. ок. 500 г. до н.э. Он считается основателем **даосизма** и автором **Дао Де Цзин**, библии даосизма.

Лучшим лидером является тот,  
О существовании которого люди едва догадываются.  
Плохо, когда люди хвалят управителей.  
Еще хуже, когда они их боятся.  
Хуже всего, когда они их презирают.  
Если ты не уважаешь людей, то и они тебя уважать не будут.  
Хороший же лидер говорит мало,  
А когда его работа сделана, и цель достигнута,  
Люди говорят: «Мы сделали это сами».

Рис. 66. Дао лидерства



Джек Уэлш, легендарный бывший руководитель General Electric, поставил себе целью сделать свою корпорацию «самым конкурентоспособным предприятием в мире». «В наше время назначение на должность — лишь начало карьеры, когда главные битвы еще впереди. Нельзя приходить на работу и сидеть сиднем в своем кабинете. Нельзя думать лишь о политике. Ты должен быть в действии постоянно, день за днем. И ты должен уметь заряжать своей энергией других». Джек Уэлш дает революционное определение нового типа менеджера XXI столетия: они «забывают о своем эго, растворяют свою индивидуальность в команде и работают на благо компании». На рис. 67 приведена диаграмма с сайта Вадима Котельникова: [http://www.cecsi.ru/coach/mgmt\\_25lessons\\_welch.html](http://www.cecsi.ru/coach/mgmt_25lessons_welch.html).

Приведу информацию только из первого урока, потому что считаю его уместным в этом разделе. Остальные уроки рекомендую вам изучить самостоятельно.

*Менеджеры копаются в мелочах, а лидеры вдохновляют. «Кого мы ищем, так это лидеров на каждом уровне, способных заряжать людей энергией, возбуждать их интерес и вдохновлять, а не нервировать, подавлять и контролировать», — говорил Уэлш.*

*Создай видение и вдохни энтузиазм в свою компанию, чтобы превратить это видение в реальность. Люди должны быть искренне увлечены своим делом так, чтобы они с трудом дожидались момента, когда можно будет привести планы в действие. Ты должен обладать огромной энергией, духом конкурентной борьбы и способностью зажечь других людей и добиться результатов. Ищи лидеров, обладающих такими же качествами.*

*Сосредоточься на стратегических вопросах. Твоя задача — задать ключевые вопросы и получить на них ответы, чтобы понять, каковы стратегические проблемы каждого подразделения и в каком направлении эти подразделения развиваются. Пойми, какие таланты и капиталы им нужны, чтобы выиграть на своих целевых рынках, и сделай свою ставку.*

**Управление путем лидерства  
25 уроков Джека Уэлча**

Уроки лидеров

**Будь больше лидером и меньше управленцем**

1. Будь лидером
2. Меньше управляй
3. Ясно сформулируй свое видение
4. Упрощай
5. Будь менее формальным
6. Заряжай других энергией
7. Смотри реальности в глаза
8. Смотри на перемены как на возможности
9. Ищи хорошие идеи повсюду
10. Доводи начатое до конца

**Создай организацию-лидера**

11. Уничтожь бюрократию
12. Разрушь границы
13. Поставь на первое место ценности
14. Выращивай лидеров
15. Нацель культуру фирмы на обучение

**Раскрой потенциал людей**

16. Вовлеки каждого
17. Сделай каждого игроком команды
18. Дерзай
19. Вселяй в людей уверенность
20. Преврати работу в удовольствие

**Создай компанию, лидирующую на рынке**

21. Будь № 1 или № 2
22. Сделай качество стилем жизни
23. Постоянно внедряй инновации
24. Преврати скорость в норму
25. Создай на фирме дух маленькой компании

© Вадим Котельников cecsi.ru

**Рис. 67.** Уроки Джека Уэлша

*Не копайся в мелочах. Твоя обязанность — видеть картину в целом. Не управляй каждой деталью. Не копайся в мелочах и не трать все свое время на них, а, наоборот, вдохновляй людей на исполнение твоего видения. Окружи себя отличными сотрудниками и доверяй им, чтобы они выполняли свою работу самостоятельно на благо всей компании.*

*Вовлекай всех и приветствуй хорошие идеи из любых источников. Каждый может быть лидером, если он тем самым вносит свой вклад в общее дело. И самый эффективный способ для каждого сделать свой вклад — родить хорошую идею. Вся суть бизнеса состоит в привлечении хороших идей отовсюду. Новые идеи — это кровь компании, горячее, которое заставляет ее двигаться. «Герой — это человек с новой идеей». Попросту говоря, нет ничего более важного для бизнеса, чем создание четкого видения и генерация идей.*

*Увлекай людей не только словами, но и личным примером. Чтобы другие люди искренне стремились побить все рекорды, ты должен постоянно заражать их личным примером. Джек Уэлш всегда был образцом исполнения собственной формулы лидерства — «обладай огромной энергией, заряжай энергией других, имей конкурентное преимущество и доводи дело до конца». Его энергия была огромна, он зажигал других, обладал необыкновенным духом конкурентной борьбы и был лучше всех в претворении планов в жизнь. Это и есть ключ к успеху Джека Уэлша. Если бы у него отсутствовала какая-то из черт лидера, которые он сам пропагандировал, Уэлш не добился бы такого успеха.*



В конце этого раздела я приведу несколько собственных выученных уроков и уроков из книги К. Фиорины.

- ◆ Чем меньше требуешь от человека, тем меньше он и достигнет.
- ◆ Поговорить с каждым из сотрудников: что думают о своей работе, что мы делаем правильно, а что — нет, охарактеризовать их возможный вклад в общее дело, чего они ждут от меня:

- чем именно они занимаются;
- зачем это нужно;
- ни один из присутствующих менеджеров не будет через три месяца занимать ту же должность, что сейчас, — тем самым каждому гарантировано существенное изменение привычного круга обязанностей.
- ◆ Люди есть люди — неважно, какую должность и в какой компании они занимают, одни находятся на своем месте, другие — нет.
- ◆ Достоинство человека не измеряется его должностью или званием, но зависит от характера, способностей и умения применять эти способности.
- ◆ Когда люди напуганы, на первый план выходят их личные проблемы, стремление обезопасить себя.
- ◆ Для многих людей даже разочаровывающее, но известное настоящее гораздо предпочтительнее неизвестного будущего.
- ◆ Избавляться вовремя от наиболее «проблемных» членов команды, которые «плывут против течения» и мешают команде достигать поставленных целей.
- ◆ Часто нет возможности повышать людям зарплату. Каким же образом можно вознаградить людей за самоотверженный труд? Нужно сделать работу увлекательной:
  - организовать оповещение об успехах коллег — маленьких или больших — все равно;
  - каждый должен иметь возможность услышать свою фамилию во время очередной телеконференции или увидеть ее в итоговых отчетах/ протоколах.

## 6.2. Профессиональные и специальные навыки

Как начальник отдела вы должны обладать следующими квалификационными навыками (табл. 10).

**Таблица 10.** Профессиональные и специальные навыки начальника отдела

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
Начальник отдела	Книги: Capability Maturity Model for Software. SEI; <i>Mulcahy Rita</i> . PMP® Exam Prep;	Все навыки аналитика, а также: <ul style="list-style-type: none"> <li>• знать модель зрелости процессов компании CMMI в областях: RD, REQM, DAR, TS, PI, VER, RSKM, PP, PMC, IPM, VAL, QPM, SAM;</li> </ul>

Продолжение ⇨

(Продолжение)

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
	<p><i>De Carlo Doug.</i> eXtreme Project Management;</p> <p><i>Мициберг Г.</i> Структура в кулаке: создание эффективной организации;</p> <p><i>Адизис Ицхак.</i> Идеальный руководитель;</p> <p><i>Фиорина Карли.</i> Трудный выбор. Уроки бескомпромиссного лидерства в сложных ситуациях от экс-главы Hewlett-Packard.</p> <p>Рекомендую регулярно посещать сайты: <a href="http://www.iteam.ru/">http://www.iteam.ru/</a>; <a href="http://msdn.microsoft.com/en-us/architecture/default.aspx">http://msdn.microsoft.com/en-us/architecture/default.aspx</a>; <a href="http://www.cecsi.ru">http://www.cecsi.ru</a>, а также зарегистрироваться в сети профессиональных контактов (например, <a href="http://moikrug.ru">moikrug.ru</a>, <a href="http://linkedin.com">linkedin.com</a>), заниматься просветительской деятельностью, участвовать в конференциях, форумах</p>	<ul style="list-style-type: none"> <li>• уметь анализировать эти области на предмет требуемых улучшений и несоответствий с моделью;</li> <li>• разрабатывать методологию системного анализа;</li> <li>• проводить тренинги и семинары;</li> <li>• иметь четкое представление об управлении проектом/ программой проектов;</li> <li>• уметь строить и развивать команду аналитиков в проектах;</li> <li>• уметь предотвращать и разрешать конфликты в проектах;</li> <li>• уметь выявлять аналитические риски и управлять ими;</li> <li>• проводить выученные уроки по результатам выполненных работ/проектов;</li> <li>• участвовать в совершенствовании процессов;</li> <li>• разрабатывать процедуры, регламенты, рабочие инструкции;</li> <li>• создавать функциональную стратегию своего направления;</li> <li>• планировать развитие отдела, вовлекать топ-менеджеров в решение стратегических и тактических вопросов;</li> <li>• уметь строить и развивать команду;</li> <li>• строить эффективное взаимодействие с другими подразделениями;</li> <li>• разрешать конфликты на всех уровнях;</li> <li>• уметь управлять проектом;</li> <li>• профессионально развивать подчиненных;</li> <li>• уметь проводить аттестацию сотрудника;</li> <li>• курировать создание базы знаний отдела;</li> <li>• управлять совершенствованием процессов в области анализа и разработки и управления требованиями;</li> <li>• управлять формализацией процессов и созданием системы менеджмента качества отдела</li> </ul>

## 6.2.1. Командная работа



---

Я стараюсь обсуждать альтернативные пути решения проблем и открытых вопросов в пользу выбора обоснованного оптимального решения. В одном случае надо идти на компромисс, в другом — жестко отстаивать свою позицию.

Если у моих коллег существует четкая, сформулированная позиция, которую они могут аргументировать, я обязательно выслушаю все мнения. Поскольку я стремлюсь создать команду, то, как правило, стимулирую именно команду принимать текущие решения. Я выступаю в роли модератора и принимаю окончательное решение, когда команда не может его принять. Это обеспечивается путем реализации механизма оценки предложений и альтернативных вариантов в соответствии с заранее заданными критериями.

Стратегические вопросы остаются в зоне исключительно моей ответственности и обязательно согласуются мной с **executive board**. **Но команда должна** понимать контекст моих требований к ним — стратегические цели и место в их достижении.

---



---

Если рассматривать командную работу в контексте особенностей менталитета разных наций, то мне ближе всего японский подход к данному вопросу. Японцы используют особую систему принятия решений, в которых участвуют все — от руководителя до рядовых сотрудников, поэтому предложения японской стороны всегда реализуемы и эффективны. Команда японцев на переговорах формирует свое преимущество за счет того, что каждый задает вопрос в сфере, где является специалистом. Собрав информацию, японская команда будет вырабатывать решение в группе. В команде переговорщиков со стороны японских партнеров возможны замены на лету, чтобы как можно больше людей узнали вас и вашу команду. Японцы очень осторожны, они искусно затягивают переговоры, чтобы достичь консенсуса внутри команды. При этом команда всегда старается действовать в соответствии с генеральной линией, для чего поддерживает постоянную связь с центральным офисом и запрашивает оттуда инструкции. Чтобы сохранить гармоничность и возможность завершить переговоры, если эмоции сторон берут верх, японцы готовы пригласить посредника, которому доверяют обе стороны.

---

По моему мнению, почти все подходы из перечисленных можно смело брать в свой портфель и применять на практике.



---

Рассмотрим пример командной работы уже из моей практики. Перед командой из пяти менеджеров нашей компании стояла задача провести сложное совещание с внешним аудитором процессов, которого компания наняла на основе Time and Material для проведения гар-анализа процессов разработки ПО в компании. Так как аудитор был из компании, сертифицированной SEI и занимающейся аудитом и оценкой зрелости процессов компании по международной модели CMMI, **тариф этого специалиста был достаточно высоким**. Перед командой стояла задача — провести совещание по обсуждению

примерно 80 вопросов, постаравшись уложиться в два часа рабочего времени, то есть на каждый вопрос в среднем приходилось две минуты. Мы прекрасно понимали, что обычные методы проведения совещания растянут эту встречу на 4–6 часов, а значит, на несколько сессий, поскольку работать эффективно на таких совещаниях больше двух часов невозможно. Не помогут здесь и методы проведения совещания по схеме «мозгового штурма». В итоге была придумана стратегия проведения командного совещания. Эта стратегия заключалась в том, что команда заранее готовит список вопросов, выделяет в своем составе роли с четкими обязанностями, тщательно следит за временем проведения совещания, не ведет никаких обсуждений ни с аудитором, ни между собой непосредственно на совещании и задает уточняющие вопросы только для того, чтобы однозначно понять мнение аудитора. Список вопросов был заблаговременно подготовлен и обсужден на отдельных совещаниях команды, в результате остались только самые важные вопросы с согласованными в команде формулировками. Были распределены роли:

- ♦ ведущий — вводное слово, описание повестки встречи и ее регламента, озвучивание заранее подготовленных вопросов;
- ♦ Time Line Controller — человек, обязанностью которого было раз в пять минут молча подходить к флипчарту и писать на нем, сколько времени осталось на текущий момент;
- ♦ персональный секретарь аудитора — нес ответственность за фиксацию непосредственных ответов аудитора на заданные вопросы как можно ближе к тексту;
- ♦ секретарь — отвечал за фиксацию обсуждений рекомендаций со стороны команды и дополнительных комментариев аудитора как можно ближе к тексту.

Перед началом совещания все получили специальную таблицу с вопросами, согласованными с командой, вариантами ответов на эти вопросы и двумя пустыми графами для записи рекомендации аудитора и собственных мыслей и комментариев. В результате проведения совещания в соответствии с этой стратегией команда успела в запланированное время не только получить и проанализировать рекомендации аудитора по всем своим вопросам, но и в оставшееся время обсудить дополнительные вопросы в неформальной обстановке.

Далее предлагаю вашему вниманию выученные уроки из собственной базы знаний выученных уроков и интересные практики из книги К. Фиорины.



### **Методика «переворачивания страниц»**

Обсуждаем страницу (пункт повестки) до тех пор, пока:

- 1) все участники не придут к какому-то соглашению;
- 2) не признают, что это пока невозможно;
- 3) если нет лиц, полномочных принимать решения, страница откладывается.

Когда все вопросы, связанные с очередной страницей проспекта, обсуждены и отнесены к первой из трех категорий, страница переворачивается и мы больше к ней не возвращаемся.

### **Согласование действий основывается на совместно разделяемых целях**

Сотрудничество имеет в своей основе общие представления об успехе и методы его оценки.

- ♦ Все должны понимать, что *другого* пути к достижению этих целей нет.
- ♦ Сотрудничество требует принятия персональной ответственности.
- ♦ Сотрудничество предполагает доверие к профессионализму и добросовестности партнеров.
- ♦ Управление на основе сотрудничества определяется процессами и их результатами, передаваемыми от одной группы сотрудников другой. Лучше всего систему можно понять, проанализировав, кто, с кем и по каким поводам общается — на совещаниях, посредством электронной почты или по телефону.
- ♦ Следует доводить одну и ту же информацию до всех сотрудников, причем желательно одинаковым способом и в одно и то же время.
- ♦ Устные сообщения всегда нужно подкреплять изложенными на бумаге соображениями.

### **«Я не могу выполнить это задание»**

Не знает, как приступить к выполнению задачи. Поговорить о возможных вариантах реализации задачи.

- ♦ Что нужно для решения задачи? Новые навыки, углубленный анализ возможностей или творческий подход к нетривиальным способам решения?
- ♦ Разговор должен быть открытым и взаимно полезным. Задумывались ли мы над этим? А может, стоит попробовать вот это?
- ♦ Кого еще из сотрудников следует привлечь для решения задачи?

Это самый легкий выход из положения. Скромные результаты требуют гораздо меньше сил и энергии для их достижения.

- ♦ Следует подчеркнуть реальность выдвигаемых требований и правильность предлагаемых методов действий.
- ♦ Дополнительно мотивировать людей, чтобы переключить их с поиска самого легкого пути на поиск самого эффективного.

Действительно не могут выполнить это задание.

- ♦ Убедиться в том, что все участники одинаково оценивают ситуацию.
- ♦ Почему участники рассматривают ситуацию с разных точек зрения? Можно ли найти какую-то общую почву для согласования взглядов?

Если выполнение поставленной задачи нереально, может, следует ее переформулировать?

---

Таким образом, успешная команда — это взаимосвязанный комплекс таких компонентов, как:

- ◆ **отлаженные инструменты;**
- ◆ **наработанные стандарты и библиотека;**
- ◆ низкий коммуникационный барьер;
- ◆ хорошее общее понимание;
- ◆ высокий кредит доверия;
- ◆ устоявшиеся традиции.

Первые два компонента могут (и должны) быть определены в системе менеджмента качества (СМК) компании, но самое главное, что делает вашу команду эффективной, — тот путь, который ее члены прошли вместе.

## 6.2.2. Политика. Баланс интересов. Переговоры

Занимая позицию начальника отдела, вы уже не можете дистанцироваться от политики. Как говорится, «если вы не занимаетесь политикой, то политика займется вами».

### Конфигурация организации

В основе понимания политической жизни организации лежит понятие «организационная конфигурация». Если вы сумеете правильно определить конфигурацию вашей организации, то сможете выработать оптимальную линию поведения. Невозможно, например, в компании с прямым управлением строить какие-то эффективные процессы без прямого участия главного лица компании. Эти действия заранее обречены на провал. С другой стороны, нет смысла беспокоить главное лицо компании с конфигурацией «механистическая бюрократия». С большой долей вероятности в СМК такой компании вы найдете ответы на большинство своих вопросов.

Генри Минцберг — признанный мировой авторитет в области организационного управления и проектирования в своей книге «Структура в кулаке: создание эффективной организации» рассматривает организацию как совокупность пяти основных частей (рис. 68) [16].

Эти пять частей, согласно Г. Минцбергу, — «операционное ядро организации, куда входят ее члены (операторы), выполняющие связанную с производством товаров и услуг основную деятельность. Стратегическая вершина, которая обязана обеспечить эффективное выполнение организацией ее миссии, а также обслуживание потребностей тех, кто контролирует организацию или обладает иной властью над ней (собственники, государственные органы, профессиональные объединения, группы влияния). Стратегический апекс и операционное ядро соединяет цепь наделенных формальными полномочиями менеджеров



**Рис. 68.** Основные части организации

срединной линии. Аналитики техноструктуры, специализирующиеся на вопросах управления, определяют формы стандартизации в организации. На схеме практически любой современной крупной организации мы видим множество организационных единиц (все они специализированные), обеспечивающих поддержку организации за рамками текущего рабочего процесса. Эти единицы образуют вспомогательный персонал».

Если рассмотреть функционирование организации, то можно выделить некоторые основные функциональные потоки (рис. 69). Каждый такой функциональный поток наилучшим образом сопоставим с соответствующим механизмом конфигурации.

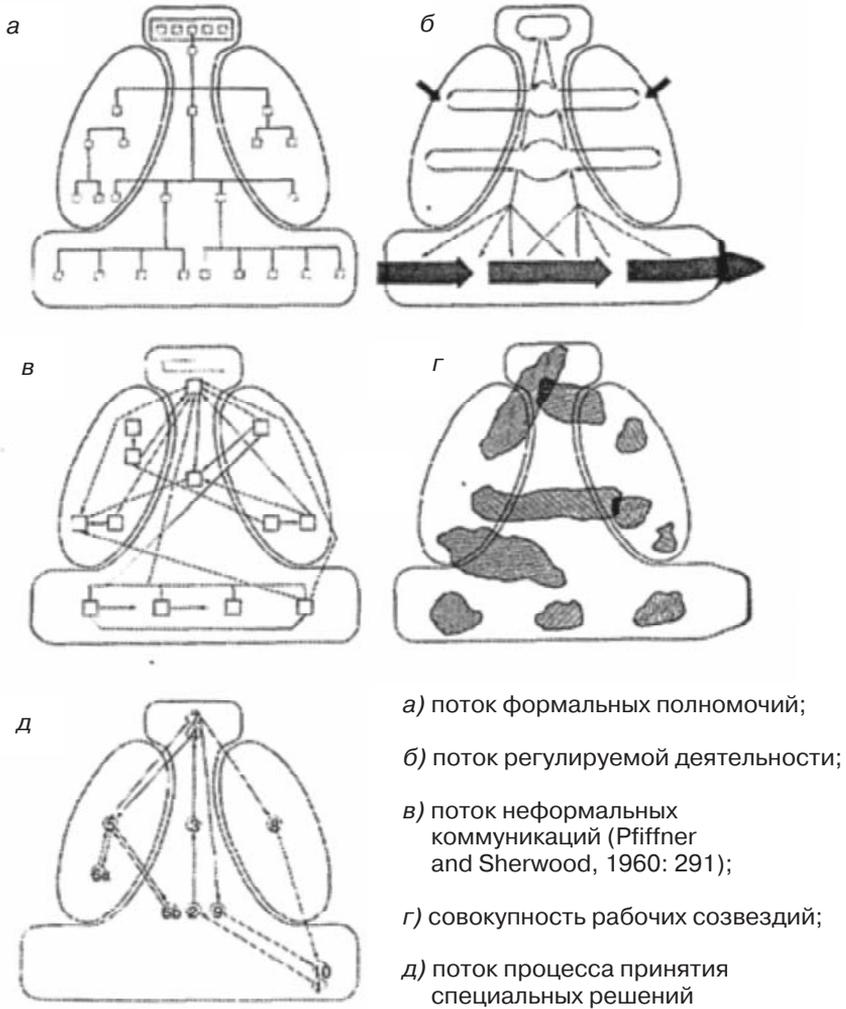
Минцберг выделяет пять координационных механизмов, которые «...раскрывают способы, посредством которых организации координируют свою деятельность: взаимное согласование, прямой контроль, стандартизация рабочих процессов, стандартизация выпуска и стандартизация навыков и знаний (квалификации)».



Для эффективного управления организацией потоки функционирования должны использовать оптимальные механизмы координации:

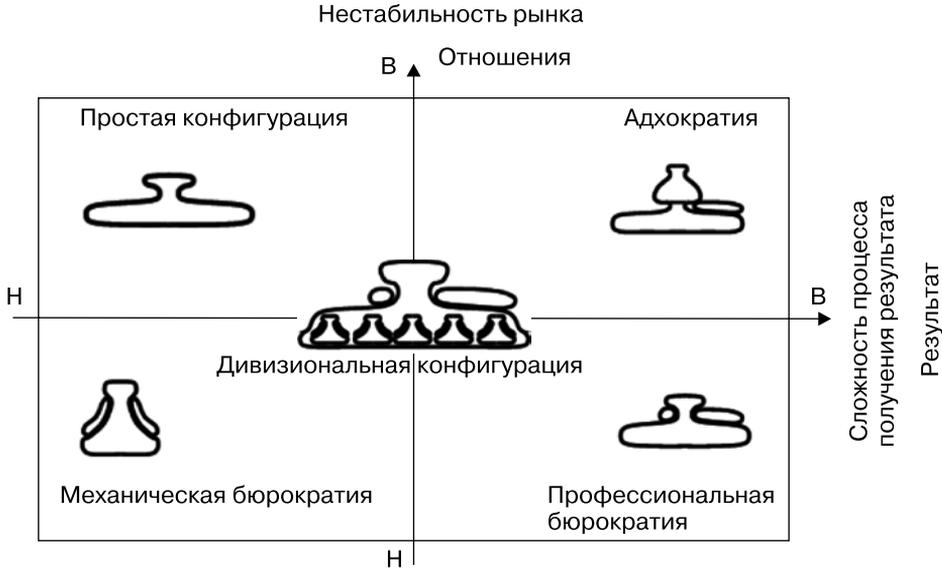
- ♦ поток формальных полномочий — прямой контроль;
- ♦ поток регулируемой деятельности — стандартизация рабочих процессов и операций;
- ♦ поток неформальных коммуникаций — взаимное согласование, корпоративная культура и культура коммуникаций;

- ♦ совокупность рабочих созвездий — стандартизация навыков и знаний (квалификации);
- ♦ поток процесса принятия специальных решений — стандартизация выпуска.



**Рис. 69.** Потоки функционирования организации

Минцберг также выделяет несколько типовых организационных структур: простую конфигурацию, механистическую бюрократию, адхократию, профессиональную бюрократию и дивизионную конфигурацию в зависимости от различных условий бизнеса (рис. 70).



**Рис. 70.** Соответствие конфигураций компании различным условиям бизнеса

В зависимости от условий бизнеса конфигурации претерпевают изменения, одни части усиливаются, другие упраздняются совсем или атрофируются, соответственно изменяются и механизмы координации внутри компании (см. рис. 70).

Конфигурация организации влияет на все процессы компании, в том числе на процессы управления проектами, и является портретом организации.

Для управления проектами наиболее критичны построение эффективных проектных коммуникаций и управление рисками. Следовательно, важно понимать, какие потоки функциональности в компании являются доминирующими и какие механизмы координации следует стандартизировать в первую очередь (рис. 71).



Например, компания по разработке «коробочного» и заказного ПО не может «в чистом виде» быть отнесена ни к одному из классических типов конфигурации по Минцбергу. Конфигурация компании может быть определена как:

- ♦ механистическая бюрократия в области производства «коробочного» ПО — производство «по конвейеру»;
- ♦ профессиональная бюрократия в области адаптации «коробочных» продуктов в конкретные решения под индивидуального заказчика;
- ♦ адхократия в области заказных разработок;
- ♦ дивизиональная конфигурация с точки зрения балансирования портфеля проектов и потребностей в бизнесе и технологического развития продуктов компании.



**Рис. 71.** Рекомендации по стандартизации компании в зависимости от ее конфигурации



Для каждого типа конфигураций на основании анализа доминирующих функциональных потоков можно выработать рекомендации по основным механизмам координации, которые должны быть стандартизированы в первую очередь:

- ♦ простая конфигурация — прямой контроль;
- ♦ механистическая бюрократия — стандартизация рабочих процессов и операций;
- ♦ дивизиональная конфигурация — взаимное согласование, корпоративная культура и культура коммуникаций;
- ♦ профессиональная бюрократия — стандартизация навыков и знаний (квалификации);
- ♦ адхократия — стандартизация выпуска.

## Баланс интересов

При работе на позиции начальника отдела вам придется участвовать во многих совещаниях, посвященных как сиюминутным проблемам, так и стратегическим целям и тактическому планированию.

Главная рекомендация здесь — «гасить личность». На сложных переговорах я часто употребляю фразы: «Я уважаю ваш профессионализм и вижу, как мно-

го сил и стараний вы прикладываете. Мои замечания относятся к процессу, а ни в коем случае не к вам лично и не к вашей команде. Уверен, то, что я сообщу, вы и без меня знаете, но выскажу свои ожидания от нашей совместной работы». Очень важно постараться с самого начала задать конструктивный тон переговоров, дистанцируя обсуждаемые проблемы от личностей присутствующих.

Далее приведены выдержки из книги К. Фиорины, которые я часто использую в своей работе.



---

«Старички» могут обидеться, если обнаружить что-то неизвестное им. Важно не обрушиваться на них с критикой их работы, а мотивировать на то, что еще надо сделать.

Никогда не угрожай, если не готов выполнить свою угрозу. Никогда не угрожай, если есть шанс добиться успеха логикой и убеждением. Но если другого выхода нет, угроза должна касаться чего-то важного для оппонента. И если она прозвучала, нужно стоять на своем во что бы то ни стало.

Чтобы провести эффективные переговоры, сначала надо узнать все, что только возможно, о вашем партнере. Проявите к нему почтение, продемонстрировав уважение к вещам, имеющим большое значение для него, не ленитесь выделить время на создание атмосферы доверия и взаимопонимания. Уважение и доверие лежат в основе всех успешных переговоров, а также помогают сохранить контакты между сторонами, если возникнут разногласия.

Участие в культурных обычаях других народов (здесь можно применить к культуре и обычаям других подразделений. — *Примеч. А. П.*) прокладывает путь к общему взаимопониманию.

Проверять убеждения или мнения других людей, опровергая их аргументы и наблюдая за тем, насколько упорно и логично они их отстаивают.

Чтобы принять правильное решение, мне нужно знать ваше настоящее мнение. Если я спорю с вами, или нажимаю на вас, или задаю встречный вопрос, я вовсе не обязательно не согласна с вами. Это просто размышления вслух. Таким образом, я стараюсь убедиться в том, что мы обсудили все аспекты проблемы и ничего не упустили.

Выслушать точку зрения каждого и затем принять решение.

---



---

Если в компании не существует системы сбалансированных показателей (BSC), **рекомендую попытаться построить ее хотя бы частично с теми подразделениями, с которыми вы непосредственно контактируете. BSC состоит из KGI, KPI и CSF.**

---

## KGI (Key Goals Indicators)

KGI предназначены для контроля достижения целей процессов, описывают результат исполнения процесса, измеряют, «что должно быть достигнуто»,

и являются индикаторами успешности процесса. KGI сфокусированы на измерениях BSC «Финансы» и «Клиенты».

Иначе говоря, это измеримые параметры, которые характеризуют результаты исполнения бизнес-процессов по отношению к финансам или клиентам. Как правило, анализ этих индикаторов выполняется уже после реализации проекта, то есть эти параметры отвечают на вопросы: насколько успешно все прошло? Попали ли мы в те границы по параметрам, которые сами себе установили? Почему нет, если не попали?



Например:

**итоги операционной деятельности:**

- ◆ сумма операционной прибыли (ЕВITDA) за период времени по клиентам, по проектам и суммарно;

**эффективность работы:**

- ◆ внутренний тариф в месяц = зарплата сотрудника/количество оплаченных часов — по брендам, по проекту, в общем;
- ◆ внешний тариф в месяц = стоимость работ сотрудника, оплаченных клиентом/количество оплаченных часов — по бренду, по проекту, в общем;

**узнаваемость:**

- ◆ доля выигранных фестивалей = N выигранных фестивалей/N фестивалей, в которых участвовали.

## KPI (Key Performance Indicators)

KPI предназначены для контроля качества результатов процессов, определяют, насколько хорошо процессы достигают поставленных целей, прогнозируют возможность успехов или неудач процессов, сфокусированы на измерениях BSC «Персонал» и «Внутренние процессы», позволяют улучшать процессы в ходе измерения и воздействия на них.

Иначе говоря, это измеримые параметры, которые, как приборы самолета, показывают, «как мы летим», позволяют вмешаться и что-то поменять в процессе реализации проекта.



Например:

**качество работы:**

- ◆ количество ошибок на каждой итерации на создание артефактов;
- ◆ количество итераций на создание артефактов;
- ◆ плотность дефектов;
- ◆ количество post-production-дефектов;
- ◆ количество критических post-production-дефектов;

**эффективность проекта/клиента:**

- ♦ доля типа работ в проекте (по каждому типу проектов);
- ♦ доля затрат на клиента = время, потраченное сотрудником (ролью) на артефакт, проект, бренд, клиента;

**финансовые процессы:**

- ♦ долевое соотношение суммы задолженности аутсорсерам, ФОТ и плановых инвестиций;

**развитие клиентов:**

- ♦ количество входных контактов (брендов);
- ♦ эффективность освоения бюджета = процент освоения годового бюджета по клиенту/плановый процент освоения бюджета за период времени (месяц, квартал).

## CSF (Critical Success Factors) процессов

Критические факторы успеха процессов — это то, что может помешать эффективному выполнению производственных процессов.



Например:

- ♦ наличие ясных процессных коммуникаций;
- ♦ набор стандартов ограничений от ИТ, Client Service и ограничений бюджета;
- ♦ стоимость примерных вариантов различных решений;
- ♦ владение новыми технологиями и методами разработки ПО — C#, .asp, SQL;
- ♦ наличие шаблонов ТЗ и спецификаций;
- ♦ ключевые компетенции по возможности отторгаются и формализуются в виде презентаций по передаче знаний;
- ♦ существует и накапливается история отношений с клиентом, формируются портрет клиента и рекомендации по коммуникациям с ним, эта информация учитывается при определении рисков проектов;
- ♦ осуществляется эффективный контроль задач, планирования времени и загрузки сотрудников;
- ♦ выработаны и признаются правила коммуникаций и области ответственности между участниками процессов.

Сформированные KPI, KGI и CSF должны быть согласованы между собой, после чего для каждого из этих параметров должен быть определен перечень инициатив по выполнению целей/достижению параметра в заданных границах. В ходе такого планирования вы со своими коллегами выработаете единое представление о том, какие инициативы первоочередные, лучше поймете зоны ответственности друг друга, как ваши инициативы связаны с инициативами

других подразделений и как ваши цели влияют на цели других подразделений. При условии конструктивного взаимодействия эта работа в конечном итоге приведет к эффекту бизнес-синергии, о которой мы писали выше.

## Механизм достижения согласованных позиций



Я хочу поделиться с вами неоднократно проверенной на практике, обобщенной WBS по согласованию позиций и ожиданий.

1. Формально определяйте цели согласования позиций.
2. Проводите эту работу в форме программы проектов.
3. Определяйте устав данной программы проектов хотя бы в форме протокола совещания.
4. Определите перечень областей, в которых вы должны достичь согласованных позиций.
5. Определите по одному ответственному за каждую область.
6. Определите перечень ЗЛ, которые должны быть опрошены каждым ответственным.
  - А. Могут появиться «виртуальные ЗЛ», например методологии или стандарты. В данном случае в качестве ожиданий берутся положения этих материалов.
7. Ответственный за область должен встретиться с каждым из ЗЛ и сформировать единый и непротиворечивый перечень ожиданий ЗЛ от области, за которую он отвечает.
  - А. Ни одно из выявленных ожиданий не удаляется.
  - Б. Если есть спорные ожидания, они помечаются как спорные, фиксируются обе точки зрения.
8. На общем собрании рассматриваются сформированные реестры ожиданий.
  - А. В первую очередь рассматриваются спорные ожидания. Процесс может потребовать много времени. Лучше выполнять эту деятельность итеративно.
  - Б. Во вторую очередь просматриваются все выявленные ожидания, приоритизируются и утверждаются.
9. Наметить инициативы для выполнения ожиданий в соответствии с приоритетами. Повторить циклы 5–7 для инициатив. Провести трассировки от ожиданий к инициативам.
10. Сформировать портфель проектов по реализации согласованных инициатив.
11. Передать выполнение проектов в РМО.
12. Сформировать реестр рисков уровня программы проектов и управлять им на периодической основе в тесном контакте с менеджерами проектов.

Очень часто компании начинают серьезные преобразования, не разграничив зоны ответственности, не выработав никакого подхода к выполнению этой деятельности, положившись на то, что менеджеры достаточно квалифицированы, чтобы достигнуть соглашения естественным путем. Это не работает. Каждый раз, когда я или мои коллеги попадали в проекты, в которых не было построено управление ожиданиями, возникало огромное количество конфликтов, «топтания на месте», переделок и огромных затрат денег без достижения каких-то серьезных результатов. И только после того, как такой деятельностью начинали управлять в соответствии с WBS, приведенной выше или аналогичной, наблюдался прогресс и через некоторое время программа становилась прозрачной и управляемой.

Не надейтесь, что проблемы и открытые вопросы будут разрешаться сами собой, будьте проактивными и заранее организуйте работу, используя описанный механизм как шаблон.



---

Далее перечислены основные риски позиции «начальник отдела», для которых в первую очередь рекомендуется применять механизм достижения согласования позиций.

---

#### Основные риски начальника отдела анализа:

- ◆ отсутствие вовлеченности топ-менеджеров;
- ◆ отсутствие согласованного видения развития компании/департамента/отдела;
- ◆ отсутствие утвержденных процедур разработки ПО и управления разработкой;
- ◆ отсутствие четко определенных процедур коммуникаций;
- ◆ отсутствие четкого разграничения полномочий и функций между ролями;
- ◆ недостаточно высокий уровень корпоративной культуры и этики.

## Переговоры/координационные совещания руководителей

В этом разделе я приведу свои персональные выученные уроки по участию в переговорах и координационных совещаниях со средним и высшим менеджментом компании.



---

Нужно не доносить свои мысли, а строить беседу через поощрение к высказыванию своего мнения каждым участником собрания.

*«Я высказал свое мнение, по этому вопросу мне на данный момент добавить нечего. Давайте послушаем и другие мнения, чтобы составить полную картину», — не вдавайтесь в длинные дискуссии и диспуты.*

Скорость предоставления замечаний и конструктивной критики должна быть ниже, чем скорость предоставления замечаний со стороны отвечающего

за направление руководителя. Иными словами, «не лезь поперек батьки в пекло».

Поэтапное усложнение и усиление требований при согласовании «сложных» документов.

*«Мы развиваемся, компания развивается, процессы изменяются и развиваются — наши представления и позиция по тем вопросам, которые мы обсуждали ранее, также могут измениться. Это нормально. Нужно обсуждать новую актуальную позицию на сегодняшний день, не вспоминая, что и кто говорил раньше».*

*«Документ (первый драфт документа) я расцениваю как подготовительный, концептуальный. При ревью этого документа я сознательно не обращаю внимания коллег и не делаю мелких замечаний, чтобы иметь возможность решить общие, концептуальные вопросы и двинуться дальше».*

## RACIchart

Эффективным инструментом для разграничения деятельности и фиксирования достигнутых договоренностей и согласованных позиций по ответственностям разных ролей в процессе разработки является RACI-таблица (рис. 72), где:

- ◆ **Responsible** — ответственный, роль, которая непосредственно работает над задачей;
- ◆ **Accountable (Approver)** — принимающий решения;
- ◆ **Consult** — вовлекается для принятия решений или для начала работы;
- ◆ **Inform** — должен быть информирован о решении или выполнении работы.

Roles and Responsibilities Analysis

Business Processes	Functional Roles								
	R		A		C		I	C	
	A	R		R	C	C	I		I
Decisions/ Functions/ Activities	C		R		C	C		R	A
	C		A			R		R	
	I	C		R	A		C		R
		I		C	R	A			C

Рис. 72. RACI-таблица



RACIchart строится как для ролей, так и с указанием конкретных исполнителей. В этом разделе я хочу привести рекомендации по анализу RACI-таблицы. Анализ проводится по двум направлениям — по строкам и по столбцам матрицы (таблицы).

Наблюдение	Трактовка / рекомендация
<b>Анализ по строкам</b>	
Отсутствуют «R» или «A»	Критическая ситуация. Не определено, кто выполняет работу, но самое главное — кто несет ответственность за ее выполнение
Более чем одна «A»	Критическая ситуация. А кто именно отвечает за результаты работы?
Слишком много «R» в одной строке	Возможно, вовлечено слишком много исполнителей. Это приводит как к накладным расходам, так и к возрастанию коммуникационных рисков
Много «C»	Действительно ли необходимо столько консультантов, добавляет ли участие каждого из них ценность результату, упрощает ли работу?
Много «I»	Все ли эти лица должны знать о результатах выполнения работы? Определены ли каналы коммуникации и форма отчетности с ними? Это серьезное усложнение коммуникаций в проекте и риски коммуникаций
<b>Анализ по столбцам</b>	
Слишком много «R»	Роль не перегружена?
Нет свободных ячеек	Огромный риск по персоналу. Точно никто больше не может заменить эту роль/человека?
Отсутствуют «R» или «A»	Может, эта функциональная роль вообще не нужна?
Слишком много «A»	Слишком много ответственности на одной роли/человеке. Риски согласований и оперативности принятия решений

### 6.2.3. Принципы и методы принятия управленческих решений

Как начальнику отдела вам придется принимать различные решения. Я рекомендую ознакомиться с известными и хорошо зарекомендовавшими себя принципами и методами принятия управленческих решений. Это поможет вам

в принятии взвешенных и объективных решений, что критично как с точки зрения качества работы, так и с точки зрения выстраивания прозрачных и партнерских отношений с вашими коллегами и подчиненными.

В данном разделе мы рассмотрим одну из практик СММІ, которую я настоятельно рекомендую применять любому руководителю при необходимости выработать какое-то решение. Это группа процессов DAR.

В СММІ существуют общие цели — некоторые общие принципы для всех групп процессов, а также специфические цели каждой процессной группы — основные цели процессов этой группы. Для некоторых процессных групп общие цели дополнительно детализируются с точки зрения того, что конкретно нужно выполнить в рамках совершенствования процессов этой процессной группы для достижения общей цели в рамках компании. Для достижения каждой цели СММІ рекомендует соответствующие практики.

Мой совет — разработать и внедрить в отделе руководство по анализу и принятию решений (возможно, как часть базовых правил).

SG 1		<b>Decisions are based on an evaluation of alternatives using established criteria</b>	<b>Решения принимаются на основе оценки альтернатив, используя заранее определенные критерии</b>
	SP1.1	Establish and maintain guidelines to determine which issues are subject to a formal evaluation process	Разработайте и поддерживайте в актуальном состоянии руководство по определению необходимости решения вопроса/проблемы с использованием механизма формальной оценки альтернатив
	SP1.2	Establish and maintain the criteria for evaluating alternatives, and the relative ranking of these criteria	Разработайте и поддерживайте в актуальном состоянии критерии для оценки альтернатив и относительные веса этих критериев
	SP1.3	Identify alternative solutions to address issues	Идентифицируйте альтернативные варианты решения
	SP1.4	Select the evaluation methods	Выберите метод оценки
	SP1.5	Evaluate alternative solutions using the established criteria and methods	Оцените альтернативные решения, используя заранее определенные критерии и методы
	SP1.6	Select solutions from the alternatives based on the evaluation criteria	Выберите решение из альтернативных вариантов на основании критериев оценки
GG 3		<b>The process is institutionalized as a defined process</b>	<b>Процесс определен и формально описан на уровне организации (в СМК организации)</b>

	GP2.1	Establish and maintain an organizational policy for planning and performing the decision analysis and resolution process	Разработайте и поддерживайте в актуальном состоянии организационную политику для планирования и выполнения процессов анализа и принятия решений
	GP2.2	Establish and maintain the plan for performing the decision analysis and resolution process	Разработайте и поддерживайте в актуальном состоянии план для выполнения процессов анализа и принятия решений
	GP2.3	Provide adequate resources for performing the decision analysis and resolution process, developing the work products, and providing the services of the process	Обеспечивайте процесс анализа и принятия решений правильными и нужными ресурсами при разработке рабочих продуктов и предоставлении сервиса
	GP2.4	Assign responsibility and authority for performing the process, developing the work products, and providing the services of the decision analysis and resolution process.	Назначайте права и ответственность за выполнение процесса при разработке рабочих продуктов и предоставлении сервиса
	GP2.5	Train the people performing or supporting the decision analysis and resolution process as needed	Тренируйте сотрудников, выполняющих или поддерживающих процесс анализа и принятия решений
	GP2.6	Place designated work products of the process under appropriate levels of control	Размещайте разработанные рабочие продукты процессов под соответствующий уровень версионного контроля
	GP2.7	Identify and involve the relevant stakeholders of the decision analysis and resolution process as planned	Идентифицируйте и вовлекайте правильных заинтересованных лиц в процесс анализа и принятия решений в соответствии с планами
	GP2.8	Monitor and control the decision analysis and resolution process against the plan for performing the process and take appropriate corrective action	Наблюдайте и контролируйте выполнение процесса анализа и принятия решений в соответствии с планом для принятия соответствующих корректирующих действий
GG 3		<b>The process is institutionalized as a defined process</b>	<b>Процесс определен и формально описан на уровне организации (в СМК организации)</b>

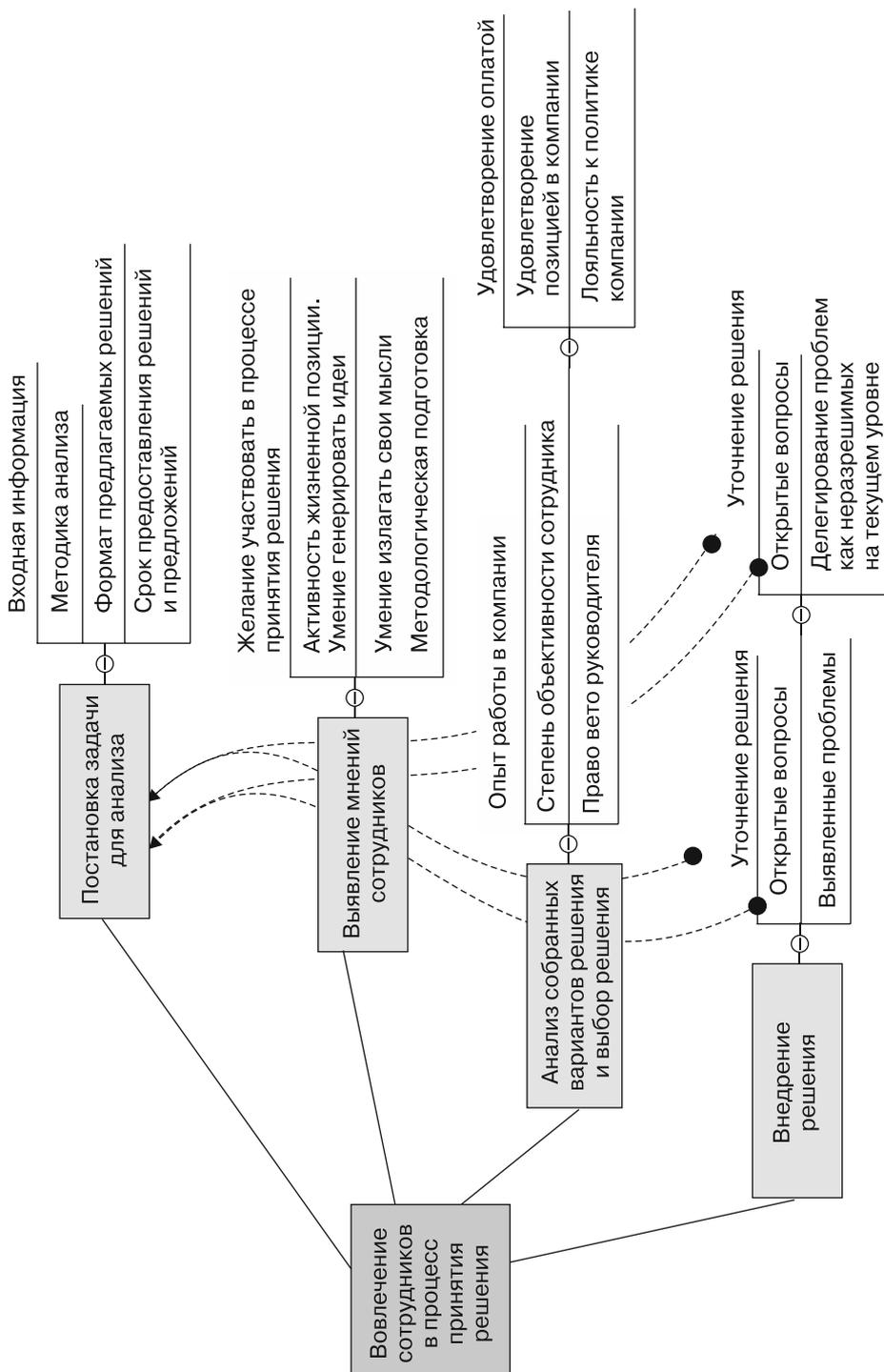
(Продолжение)

	GP2.9	Objectively evaluate adherence of the decision analysis and resolution process against its process description, standards, and procedures, and address noncompliance	Объективно оценивайте строгое соответствие применяемого процесса анализа и принятия решений описанию процессов, процедур, стандартов (в СМК) и выявляйте несоответствия
	GP2.10	Review the activities, status, and results of the decision analysis and resolution process with higher level management and resolve issues	Проверяйте активность, статус и результаты процесса анализа и принятия решений с более высоким уровнем менеджмента и разрешайте проблемы и открытые вопросы
	GP3.1	Establish and maintain the description of a defined decision analysis and resolution process	Установите и поддерживайте в актуальном состоянии описание определенного процесса анализа и принятия решений
	GP3.2	Collect work products, measures, measurement results, and improvement information derived from planning and performing the decision analysis and resolution process to support the future use and improvement of the organization's processes and process assets	Собирайте рабочие продукты, измерения, результаты измерений и информацию для улучшения процесса в ходе планирования, анализа и принятия решений, чтобы развивать и обеспечивать возможность использования процесса в будущем и улучшать процессы организации и организационные активы компании

На мой взгляд, эти рекомендации достаточно полно и всесторонне описывают основные принципы эффективного процесса анализа и принятия решений. Кроме СММІ существуют другие методы и принципы принятия решений. Я рекомендую ознакомиться со следующими: IDEAL, циклическая модель ПРУР (принципов разработки управленческих решений), модель ПРУР по Карпову, модель ПРУР по Минцбергу, а также модель принятия решений Врума – Йеттона 1988 года.

Для наиболее сложных случаев, когда цена ошибки в принятии решений крайне высока, лучше воспользоваться методами с применением взвешивания и нормализации ответов респондентов – «Метод многокритериальных оценок».

В свое время мной был разработан собственный метод вовлечения сотрудников в процесс принятия решений (рис. 73).



**Рис. 73.** Вовлечение сотрудников в процесс принятия решений (пример использования MandMap)

## 6.2.4. Корпоративная культура

### Базовые правила работы отдела

В этом разделе я приведу основные принципы, которые обычно включаются в базовые правила работы отдела или компании.

#### Принцип организованного и управляемого подхода к работе

Основной принцип организованного и управляемого подхода — организация деятельности коллектива сотрудников путем определения четких границ работ, ответственности сотрудников за выполнение конкретных работ, явных целей и критериев успешности выполнения как конкретных процессов, так и всего объема работы.

Данный принцип, как правило, реализуется в форме проектного управления. Однако даже без создания проектов и формальных артефактов управления оптимальный путь — построение упрощенных артефактов управления в свободной и удобной для участников форме с целью более четкой координации, фиксирования договоренностей, упрощения информирования сотрудников, не вовлеченных в процесс конкретной деятельности непосредственно, а также информирования руководства компании. Такой подход часто называют гибким (Agile) проектным управлением.

Принцип организованного и управляемого подхода к работе достигается за счет:

- ◆ выделения главного ответственного за выполнение всего объема работ (менеджера проекта);
- ◆ фиксирования границ работ, целей выполнения работ, критериев успешности работы и проектной команды в формальном документе или официальном письме;
- ◆ планирования и согласования иерархической структуры работ;
- ◆ согласования перечня артефактов (документов), которые будут создаваться в ходе выполнения каждой работы;
- ◆ согласования структуры разрабатываемых артефактов (документов);
- ◆ согласования плана создания, поставки и согласования артефактов (документов) с указанием ответственного за каждый артефакт (документ), списка рецензентов артефакта (документа) и списка утверждающих лиц артефакта (документа);
- ◆ четкого соблюдения обязательств по срокам, зафиксированным в плане создания, поставки и согласования артефактов (документов);
- ◆ планирования коммуникаций до начала выполнения работ. Планирование определяет успешность и своевременность информирования «правильных» людей в «правильное» время о «правильных» вещах в «правильной» форме

с целью наиболее быстрого и аргументированного принятия решений по вопросам в рамках выполнения работ. Существуют пять «золотых» правил коммуникации: первое — «кто», второе — «кого», третье — «о чем», четвертое — «когда» и пятое — «в какой форме» уведомляет. При планировании коммуникаций важно договориться и сформировать эти правила;

- ◆ правильной организации и проведения собраний: собрания должны планироваться, обсуждаемые вопросы — иметь явно обозначенную повестку и строго следовать ей, а принятые решения и дальнейшие шаги должны фиксироваться в виде протокола;
- ◆ информирования всех заинтересованных лиц о результатах выполнения работ на периодической основе.

### **Принцип проактивного построения работы**

Данный принцип заключается:

- ◆ в выполнении принципа организованного и управляемого подхода к работе;
- ◆ в инициативном подходе. Каждый участник команды проявляет максимальную заинтересованность и инициативность в оптимальном достижении качественного результата. Если сотрудник видит возможность улучшения, он открыто и прямо заявляет об этом в дружелюбном и конструктивном тоне;
- ◆ в раннем участии в обсуждении и контроле исполнения поставленной задачи. Для этого надо как можно раньше проверять промежуточные результаты выполнения работы (драфт артефакта/ документа), чтобы уже на ранних стадиях иметь возможность выявить разногласия в понимании сути предмета и устранить их до того, как будет потрачено значительное время на выполнение работ;
- ◆ в продуманном подходе к коммуникациям. Каждый сотрудник компании, начиная процесс взаимодействия, должен сначала определить пять правил коммуникации (см. выше) путем ответа на вопрос «Кого, о чем, когда (с какой периодичностью) и в какой форме я собираюсь проинформировать?» и только после ответа на него приступать к обмену информацией. Если такой информации о правилах в плане коммуникаций нет, сотрудник должен внести ее.

### **Принцип персональной ответственности за участие в результате**

Каждый сотрудник компании думает и ведет себя по принципу: «Если что-то выполнено не так, как я хотел, то в этом есть и моя недоработка, потому что я не обозначил свою позицию и не донес свои требования до других сотрудников».

Поэтому каждый сотрудник стремится донести свою мысль наиболее эффективным способом, предпочитая личное общение длительной переписке.

## Принцип адекватного реагирования и конструктивной критики

Каждый сотрудник компании думает и ведет себя по принципу: «Что сделано, то сделано — давайте подумаем вместе, как можно исправить недоработки».

Суть: «Критикуя — предлагай».

В связи с этим недопустимы замечания в неуважительной и неконструктивной форме.

Недопустимо высказывать замечания общего характера («документ никуда не годится»), без конкретизации, в чем именно и где конкретно допущены ошибки или есть недоработки.

Недопустимо и простое формальное указание на ошибку («раздел 1.22 сделан не по ГОСТу»), без информирования о том, что конкретно сделано не так и что хотелось бы видеть.

## Принцип постоянного улучшения процессов работы

В ходе любых работ рекомендуется вести реестр проблем, по окончании выполнения необходимо провести совещание по обсуждению выявленных проблем и возможных путей их предотвращения в будущем. Протокол собрания в качестве «выученных уроков» должен направляться в соответствующие исполнительные комитеты и рабочие группы (PMO, SEPG, Steering Committee).

Для постоянного улучшения процессов работы крайне важно следовать принципу адекватного реагирования и конструктивной критики.

## Базовые правила

Кроме раздела «принципы работы в организации», в базовые правила работы входят разделы, непосредственно описывающие базовые правила.



Например:

- ◆ форма общения;
- ◆ вопросы разрешения конфликтов и правила эскалации;
- ◆ правила проведения дискуссий и правила принятия решений;
- ◆ правила переписки;
- ◆ обучение и повышение квалификации;
- ◆ график отпусков;
- ◆ учет рабочего времени;
- ◆ контроль выполнения работ:
  - работы, выполняемые в рамках операционной деятельности;
  - работы, выполняемые в рамках проектной деятельности;
- ◆ документальные отчеты об аналитических исследованиях;

- ◆ проведение анализа оценок трудоемкости;
  - ◆ процессы и процедуры:
    - управление документами/артефактами;
    - проведение Peer Review;
    - проведение собраний;
    - рекомендации по подготовке и проведению презентаций;
  - ◆ управление задачами;
  - ◆ рабочее окружение;
  - ◆ создание архивных копий.
- 

## Кодекс руководителя и сотрудника компании

Еще одной неотъемлемой частью корпоративной культуры являются кодекс руководителя компании и кодекс сотрудника компании. Это высокоуровневая декларация о том, какое поведение и нормы считаются общепринятыми в компании по отношению к руководителям и сотрудникам.



В качестве примера я хочу привести кодексы руководителя и сотрудника компании HELIOSIT-SOLUTIONS, на сайт которой ([http://www.hbc.ru/ru/about/career/codex/codex\\_director/](http://www.hbc.ru/ru/about/career/codex/codex_director/)) я натолкнулся в открытом доступе, набрав фразу «*вселять дух оптимизма*» в поисковике. Эти кодексы показались мне близкими к идеалу.

---

## Кодекс руководителя

Семь качеств лидера:

- ◆ командное лидерство;
- ◆ нацеленность на победу;
- ◆ знание всех возможностей компании;
- ◆ стойкость;
- ◆ терпение;
- ◆ смелость;
- ◆ любовь к делу.

Обязанности руководителя:

- ◆ вселять дух оптимизма и победы в своих сотрудников;
- ◆ создание коллектива и взаимопонимания между людьми;
- ◆ публично поощрять успешных;
- ◆ хорошо знать своих людей и доверять им. Поддерживать в трудную минуту, щедро делиться своим опытом;

- ◆ увлекать людей работой на пределе возможностей. Предлагать интересную работу;
- ◆ предлагать к решению задачи минимум на порядок сложнее возможностей своих людей. Постоянно повышать планку;
- ◆ требовать высоких скоростей от подчиненных при реализации задач;
- ◆ терпеливо доводить до всех подчиненных главные цели и задачи подразделений. Объяснять роль каждого в общих задачах;
- ◆ мотивация и вовлечение, а не приказы и распоряжения;
- ◆ личным примером демонстрировать высокую исполнительскую дисциплину;
- ◆ незамедлительно избавляться от безответственных людей;
- ◆ постоянно заниматься развитием своих профессиональных навыков.

Недопустимо для руководителя:

- ◆ унижать подчиненного. Оскорблять, высмеивать слабости, третировать, отчитывать в присутствии других подчиненных;
- ◆ раздражаться в присутствии подчиненных, повышать голос;
- ◆ вести себя неуверенно, беспокойно, озабоченно;
- ◆ проявлять нескромность перед подчиненными, восхвалять собственные достижения, приписывать себе их идеи;
- ◆ подавлять инициативу и нестандартный подход;
- ◆ требовать подобострастного отношения к себе;
- ◆ капризничать и самодурствовать;
- ◆ создавать во вверенном подразделении дух сепаратизма и противопоставления другим подразделениям.

## **Кодекс сотрудника компании**

Пять ценных качеств: «5 × 5».

Внутренняя дисциплина:

- ◆ ответственность;
- ◆ самостоятельность;
- ◆ самоотверженность;
- ◆ требовательность к себе;
- ◆ воля в достижении результатов.

Творческий потенциал:

- ◆ новаторство;
- ◆ оригинальность;
- ◆ предприимчивость;

- ◆ инициативность;
- ◆ развитое воображение.

Боевые качества:

- ◆ оптимизм;
- ◆ стойкость;
- ◆ энергичность;
- ◆ жажда победы;
- ◆ смелость и риск.

Профессиональный рост:

- ◆ личные амбиции;
- ◆ самокритичность и работа над ошибками;
- ◆ самообразование;
- ◆ любознательность;
- ◆ самосовершенствование.

Коммуникабельность:

- ◆ работа в команде;
- ◆ способность найти взаимопонимание;
- ◆ доброжелательность;
- ◆ помощь и обмен опытом;
- ◆ уважение к коллегам.

Эти кодексы показались мне заслуживающими внимания и мотивирующими как руководителя, так и рядовых сотрудников.



---

Даже если в компании не определены такие кодексы, рекомендую создать их в вашем отделе. Наличие четких и прозрачных правил, а также регламентов общепринятых в компании норм и правил поведения является базой для честного и взаимовыгодного сотрудничества, основанного на прозрачности и доверии.

---

## 6.2.5. Системы управления деятельностью сотрудников

Для управления деятельностью сотрудников и контроля выполнения работ используйте инструменты компании, такие как WAS, PMS или ALM-система.

WAS (Work Authorization System) – это система авторизации работ, основная задача которой – быть уверенными в том, что только авторизованные руководителем работы будут выполняться, причем своевременно и в соответствии с выставленными для них приоритетами.

PMS (Project Management System) — система управления проектами, основная задача которой — планирование и контроль выполнения работ проекта. Например, MS project и MS project Server, MTFS (Microsoft Team Foundation Server).

ALM-система — это система управления жизненным циклом приложения (Application Life Time System). Она, как правило, включает в себя все системы, необходимые для обеспечения жизненного цикла продукта. Это и управление требованиями, и управление проектами, и управление задачами.



---

Рекомендую вам с первых дней работы в должности вводить как минимум WAS, которая обеспечит горизонтальную и вертикальную прозрачность работ. В WAS есть две основные сущности — работы и артефакты. Работа имеет состояние и атрибуты, такие как название, описание, история выполнения, плановая дата окончания, фактическая дата окончания, тип работы, ответственный, контролер (менеджер проекта). Кроме того, эта система имеет функционал для краткосрочного планирования выполнения работ в начале недели со стороны исполнителей и заполнения Time Sheet (данных о трудозатратах) в конце каждой недели. Используя подобную систему, вы не только будете владеть ситуацией и иметь возможность проактивно реагировать на намерение сотрудника выполнить «не ту работу не в то время» через механизм авторизации работ, но и получите в руки мощный инструмент для анализа статистической информации и сбора метрик. Эта информация и метрики помогут на основании измерений понять, качественно ли построены процессы, и рекомендовать изменения по улучшению и оптимизации процессов и деятельности компании в целом.

---



---

Если в компании нет таких систем, используйте обычный реестр работ или создайте свой собственный. Я настойчиво рекомендую создать также реестр артефактов отдела и поддерживать его в актуальном состоянии.

---

О реестрах мы с вами говорили в разделе «Основы управления» главы «Старший/ведущий аналитик».

## 6.2.6. Управление персоналом

В этом разделе я опять-таки не буду углубляться в теорию управления персоналом, а приведу конкретные рекомендации из своих персональных выученных уроков и практик.

### Критика

При критике старшего аналитика необходимо особым образом мотивировать его:

- ◆ требования к нему уже не как к новичку, а как к серьезному специалисту;
- ◆ ошибки, которые позволительны новичку, для старшего аналитика уже недопустимы;

- ◆ на него и на его обучение потрачено время дорогостоящих специалистов — кураторов и наставников, и теперь компания рассчитывает на *значимую* отдачу от своих капиталовложений.

Критику слабого сотрудника более сильным нужно пресекать, сильный сотрудник должен помогать слабому, быть ему наставником и куратором.

Цель менеджерского состава (включая старших аналитиков, которые выступают как руководители групп) заключается в том, чтобы каждый сотрудник команды чувствовал себя мотивированным и ощущал свою нужность, полезность и ценность для компании.

## Культура

Создать четкие и ясные должностные инструкции, в которых описать ответственность и обязанности аналитиков, а также профессиональный кодекс аналитика.

Воспитывать в подчиненных уважение и *доверие* к мнению профессионалов в других проектных и производственных ролях. Не считать других менее знающими, чем они. Уважать чужую точку зрения.

Базовые правила работы отдела должны редактироваться всеми. Вовлекать сотрудников в постоянные улучшения базовых правил работы. Это значительно мотивирует сотрудников на исполнение этих процессов и на создание изначально работоспособных процессов.

Вовлекать сотрудников в описание и формализацию процессов через создание необходимых процессных артефактов.

Формировать СМК и БЗ отдела силами всех сотрудников отдела. Реестры «Артефакты аналитического отдела», «Хорошие практики», «Открытые вопросы и проблемы» также должны вести все сотрудники. Не только давать прямые поручения на заполнение соответствующих разделов соответствующих реестров и регламентов сотрудникам, когда они сами «изобрели» какую-то полезную практику, но и использовать эффект перекрестного взаимодействия — когда идея пришла в голову одному, а фиксирует эту идею в регламенте другой сотрудник. Так обеспечиваются эффект синергии и сплоченность команды.

Проводить Peer Review и проектных артефактов, и процессных регламентов.

Периодически проводить ревью актуальности информации в реестрах отдела, в БЗ и СМК отдела. Использовать для принятия решений прозрачные методы и принципы (см. подраздел «Принципы и методы принятия управленческих решений»).

Учить сотрудников управлять своими чувствами и эмоциями по модели 4О, описанной в книге Дуга Де Карло «Управление экстремальными проектами».

## Профессиональное развитие

Мотивировать сотрудника иметь план персонального развития.

Минимум раз в месяц проводить ревью артефактов, создаваемых аналитиком, выявлять пробелы в знаниях и определять перспективы его профессионального развития.

Общаться «лицом к лицу», давая оценку профессионального уровня сотрудника и внося коррективы в план его развития при непосредственной встрече с сотрудником.

Проводить игровые тренинги, например такие.



Разыгрывается ситуация: я заказчик, готовый профинансировать проект по созданию ПО для какой-то организации. У меня есть отличная команда программистов-профессионалов. Мне сказали, что если я найму в команду аналитика, то «будет мне счастье». Но я не имею никакого представления, кто такой аналитик, чем он будет заниматься и за что я ему буду платить деньги. Игра состоит в том, чтобы сотрудник моего отдела смог «устроиться ко мне на работу».

Пример анализа результатов этой игры: Евгений (сотрудник) предложил мне свою кандидатуру на должность аналитика. Я пригласил его на переговоры, в процессе которых он должен был убедить меня в полезности работы аналитика, рассказать о том, какие конкретно деятельности и артефакты будет разрабатывать, зачем они нужны и чем полезны команде.

В течение переговоров Евгений не акцентировал мое внимание на том, что заказчик должен активно участвовать в согласовании и утверждении результатов работы аналитика. Я ожидал, что Евгений будет добиваться от заказчика подтверждения, что тот готов участвовать и будет выделять время на встречи с аналитиком, на утверждение каких-то документов и т. п. Евгений согласился с замечанием.

В целом оцениваю знания Евгения методологий анализа на «отлично». Дополнительно ему необходимо поработать над практиками невербального общения, особенно над моментом выхода из интервью.

### 6.2.7. Процессный менеджмент. Аналитические практики

В этом подразделе я хочу поделиться с вами идеей, как можно применить практики анализа и моделирования, в которых мы с вами профессионалы, для анализа производственных процессов и разработки системы управления предприятием.

Под системой управления (СУ) понимается совокупность процедур, рабочих инструкций, методологий, инструментов, АИС и навыков сотрудников

компании, работающих как единое целое и позволяющих за счет синергетического эффекта от их совместного использования эффективно выполнять производственную деятельность в какой-либо области знаний (дисциплине). Проще говоря, СУ — это совокупность организационной структуры, методик, процессов и ресурсов, необходимых для достижения целей управления.



---

Суть идеи состоит в использовании для понимания границ систем управления и формирования концептуального представления о возможных атрибутах каждой СУ и ее деятельности (процедурах) следующих артефактов:

- ◆ концептуальная модель;
  - ◆ модель вариантов использования;
  - ◆ диаграммы последовательности;
  - ◆ логическая модель системы.
- 

То есть как будто мы разрабатываем систему для автоматизации управления бизнесом компании (ALM-систему) по упрощенной методологии.

## Создать концептуальную модель СУПР

Границы систем управления и их взаимодействие в рамках СУПР определены в СММІ взаимодействиями процессных областей СММІ [19] (рис. 74–76).

Далее приводятся только основные диаграммы взаимодействия СММІ, достаточные для того, чтобы показать принцип формирования границ СУ в рамках СУПР (рис. 77).

### Области процесса разработки (Engineering Process Areas)

Области процесса разработки покрывают действия по разработке и обслуживанию, которые разделены на технические дисциплины и были формализованы с помощью терминологии общего машиностроения так, чтобы любая техническая дисциплина, вовлеченная в процесс разработки новой продукции (например, разработка программного обеспечения или машиностроение), могла использовать их для усовершенствования процесса.

Следует отметить, что практики TS и PI в основном будут реализованы в виде процедур и рабочих инструкций СУАР, но некоторые, в частности по моделированию и интеграции требований на уровне продуктов, будут использоваться напрямую в СУТ. Практики областей RD и REQМ будут реализованы прежде всего в рамках СУТ, но отдельные из них — и в СУАР.

Таким образом, границы СУ в рамках СУПР являются условными и могут быть изменены с целью оптимизации процедур и улучшения процессов с точки зрения здравого смысла и результативности.

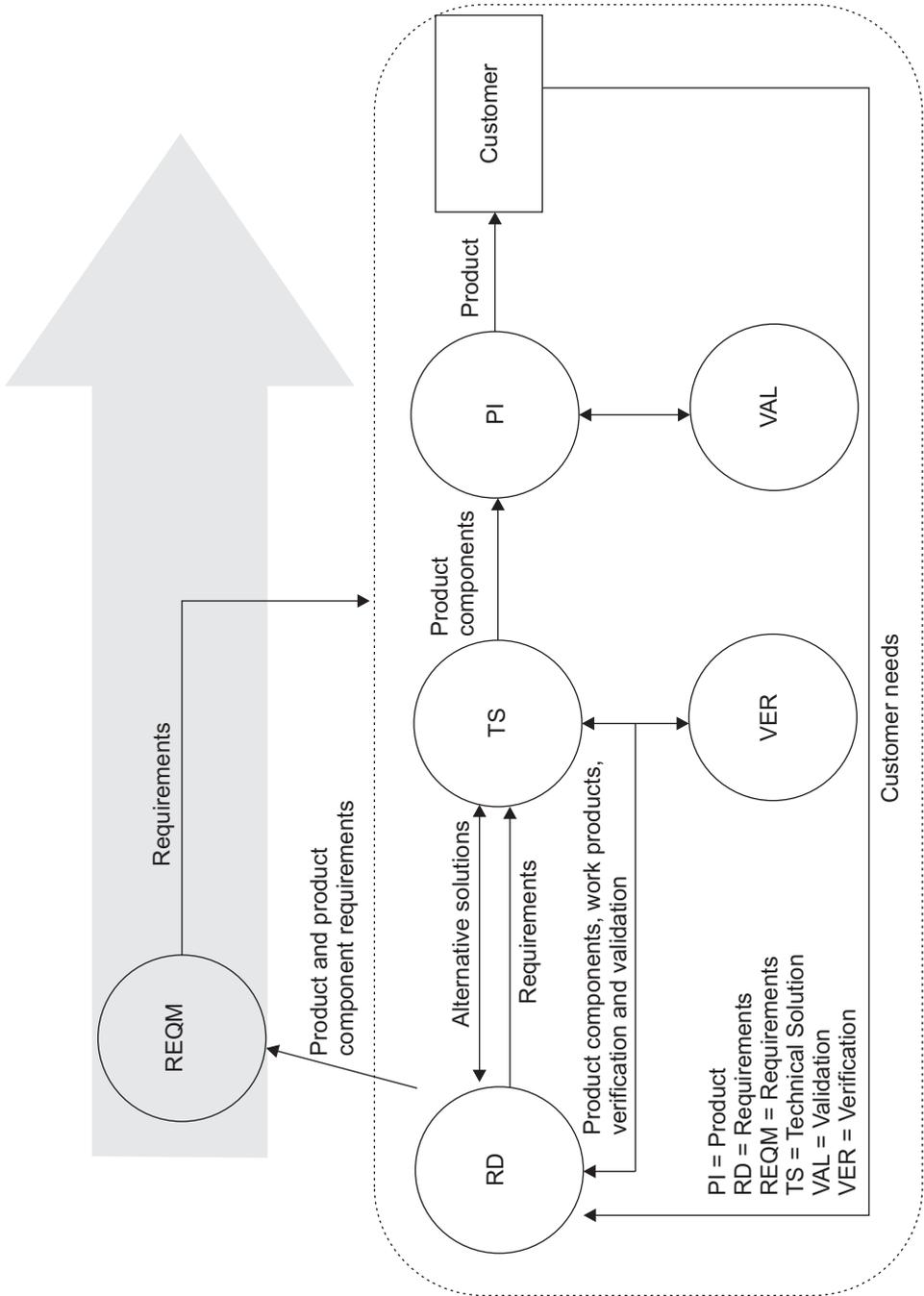
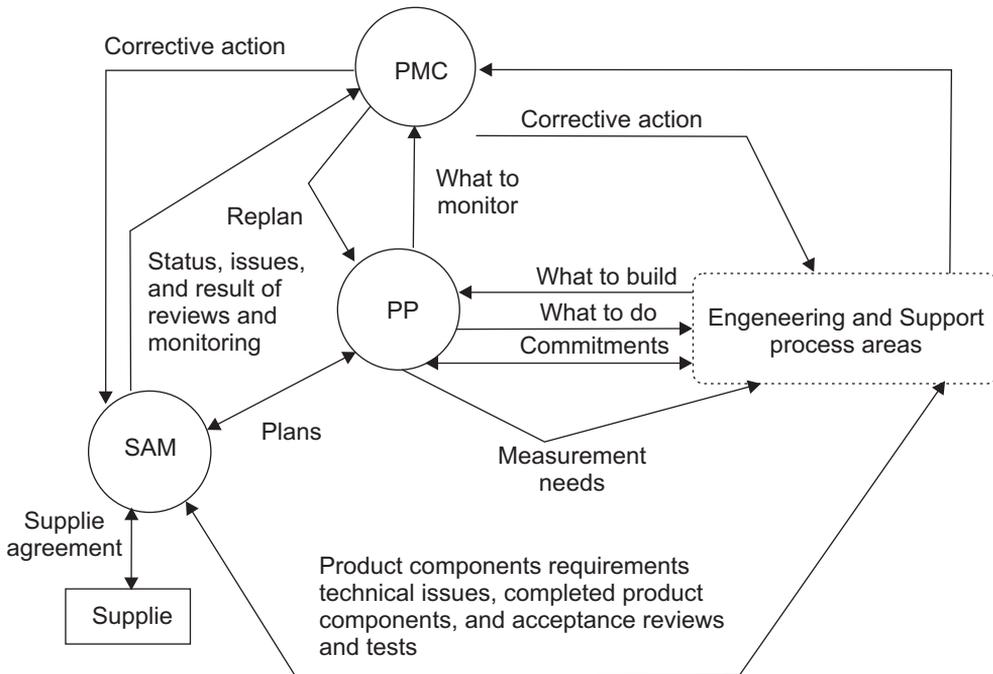


Рис. 74. Взаимодействие процессных областей Basic Engineering Process Area



PMC = Project Monitoring and Control  
 PP = Project Planning  
 SAM = Supplier Agreement Management

**Рис. 75.** Взаимодействие процессных областей  
 Basic Project Management Process Area

## Области процесса управления проектом (Project Management Process Areas)

Области процесса управления проектом покрывают действия руководства проектом, связанные с планированием, контролем и управлением проектом.

## Области расширенного процесса управления проектом (Advanced Project Management Process Areas)

Области расширенного процесса управления проектом устанавливают определенный процесс, адаптируемый из набора стандартных процессов организации, проектную рабочую среду, которая базируется на стандартах рабочей среды организации, освещает вопросы координации и сотрудничества с соответствующими заинтересованными лицами, а также управления рисками, формируется и разрабатывается через процессы интеграции.

Практики процессных областей CMMI Basic Project Management и Advanced Project Management являются частью СУП. Следует отметить, что практики процессных областей Project Planning (PP) и Project Monitoring and Control (PMC) – это своего рода интерфейсные области и практики по взаимодействию с СУТ,

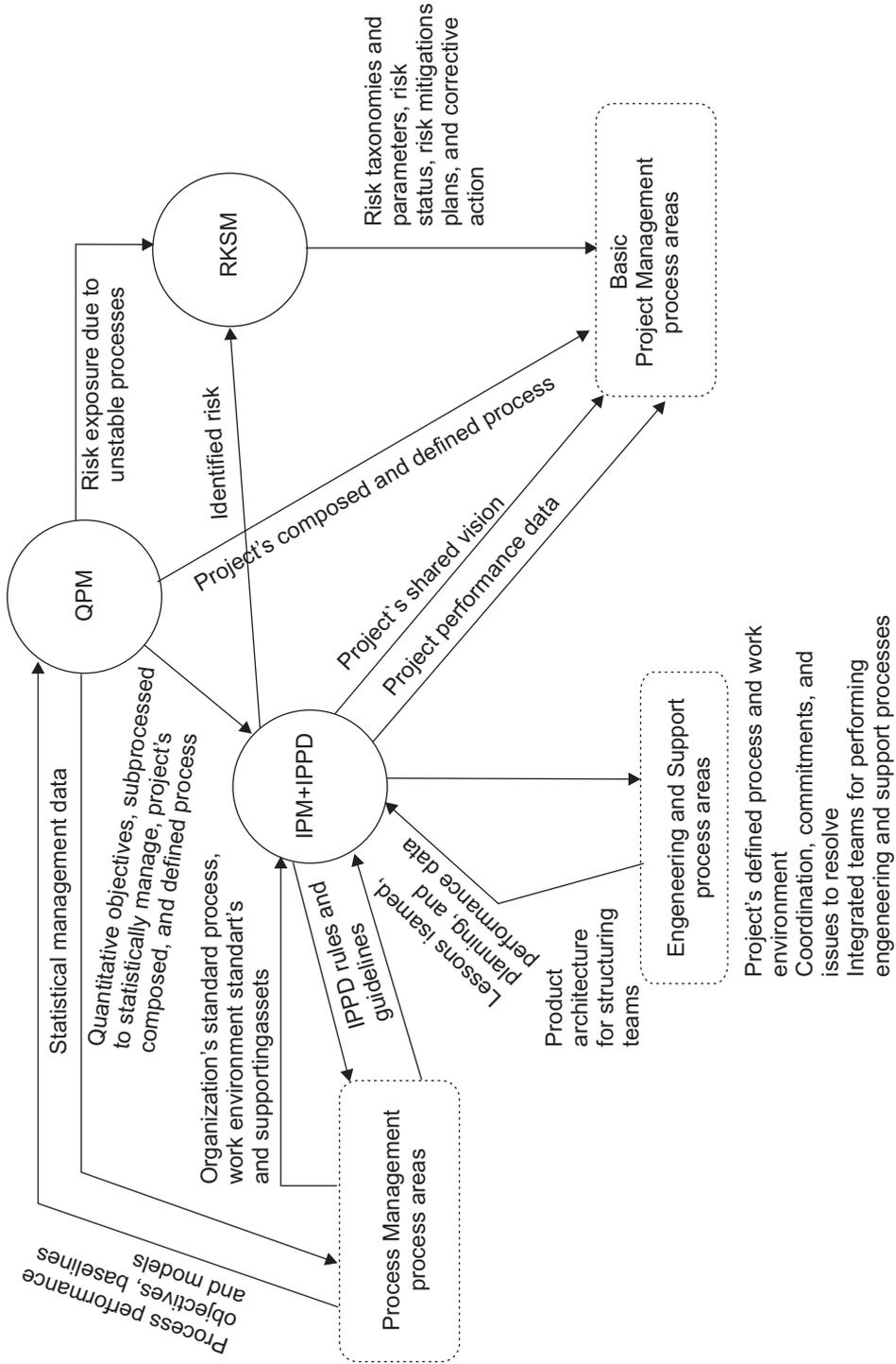
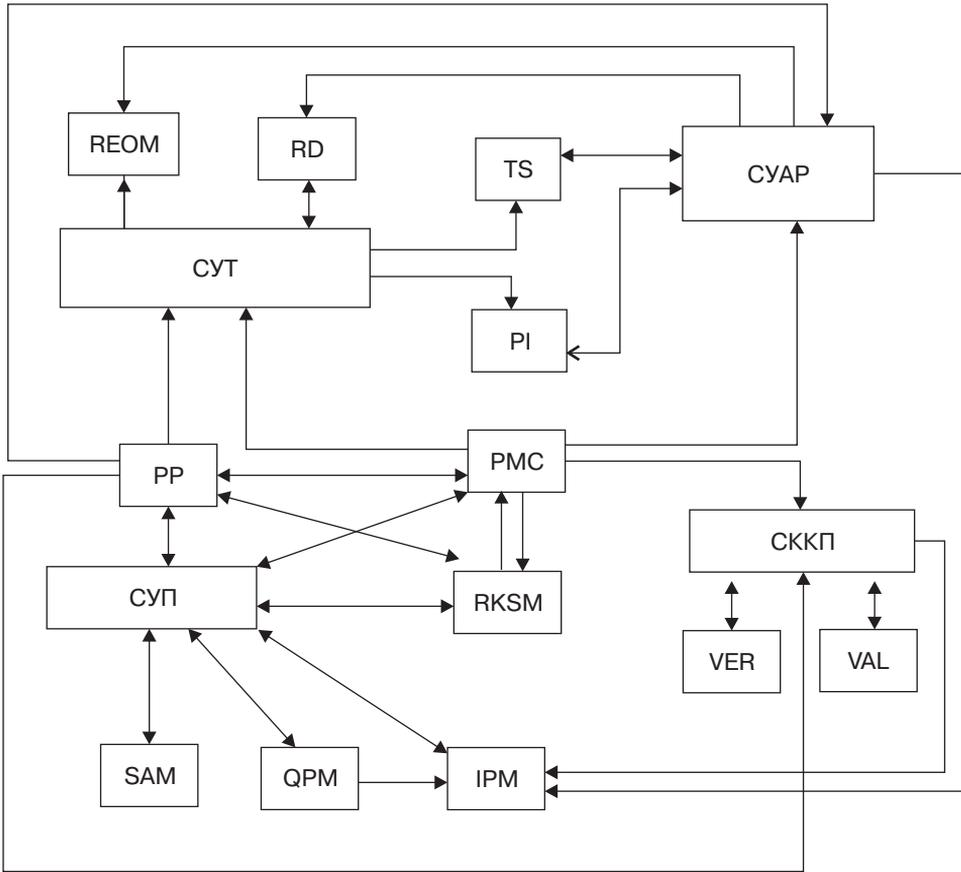


Рис. 76. Взаимодействие процессных областей Advanced Project Management Process Area



**Рис. 77.** Концептуальная модель СУПР

СУАР и СККП. Практики области Integrated Product Management (IPM) также напрямую используются в СУТ и СУАР, но основные принципы определены в рамках СУП. Таким образом, СУП является интегрирующей системой управления в рамках СУПР. Наглядный пример — процедура управления изменениями, определяемая в СУП, но на которую ссылаются и которая частично реализуется, в частности, и через процедуры других СУ.

Система менеджмента качества является хранилищем всей регламентной базы предприятия и не показана на диаграмме взаимодействия СУ. С ней взаимодействуют все СУ, так как процедуры, рабочие инструкции, методологии всех СУ располагаются в СМК и после модификации в рамках совершенствования процессов в любой СУ обновляются в СМК.

Система управления изменениями конфигурации (СУИК) также используется всеми СУ для управления их объектами конфигурации и информационными сущностями. СУИК реализует унифицированный подход к управлению версиями и конфигурациями.

Процедуры СУП строятся по стандарту РМВОК, который реализует все практики модели СММ1, учитывая, таким образом, и все практики перечисленных процессных областей.



Далее приведено концептуальное описание основных потоков деятельности СУ в рамках СУПР.

Система управления планированием выполняет стратегическое, тактическое и операционное планирование, формирует портфель проектов и инициирует выполнение проектов. В стратегическом и тактическом планировании активно участвуют СУАР и СУТ.

Система управления трудоемкостью анализирует высокоуровневые требования на всем этапе планирования, проводит оценку трудоемкости работ по созданию требований. СУАР и СККП обеспечивают оценку трудоемкости и участвуют в управлении изменениями в СУП.

Система управления планированием принимает управленческие решения и достигает соглашения с заказчиком по показателям рентабельности, срокам выпуска продукта и объему ожиданий заказчика, которые он хочет видеть реализованными к этому сроку. СУТ активно участвует в управлении ожиданиями заказчика, СУАР активно участвует в управлении архитектурно значимыми запросами заинтересованных лиц и в выборе технологий реализации продуктов.

Система управления планированием определяет тип проекта и жизненный цикл разработки ПО, инициирует проект и его выполнение. СУТ в соответствии с жизненным циклом создает требования с учетом запросов заинтересованных лиц, требований бизнеса и технических требований, определенных в СУП на этапе стратегического, тактического и операционного планирования. СУАР в соответствии с жизненным циклом разрабатывает архитектуру и активно участвует в выявлении и управлении архитектурными требованиями.

Разработанные требования и архитектура проходят Peer Review (одноранговые ревью) в проектной команде. После этого согласования требования и архитектура верифицируются в СККП, которая создает полный набор тест-планов и тест-кейсов, их ревью осуществляется в СУТ и СУП. СУАР создает продукт, который подается на вход СККП для тестирования. СККП в соответствии с разработанными тест-планами и кейсами тестирует продукт.

Система управления планированием обеспечивает интеграцию процессов всех технологических СУ, отвечает за планирование, авторизацию и управление работами и за выпуск конечного продукта или его очередной версии, которая поступает в отдел маркетинга и продаж.

Система менеджмента качества выполняет аудит качества процессов всех СУ, в ней хранятся записи о качестве и об аудитах.

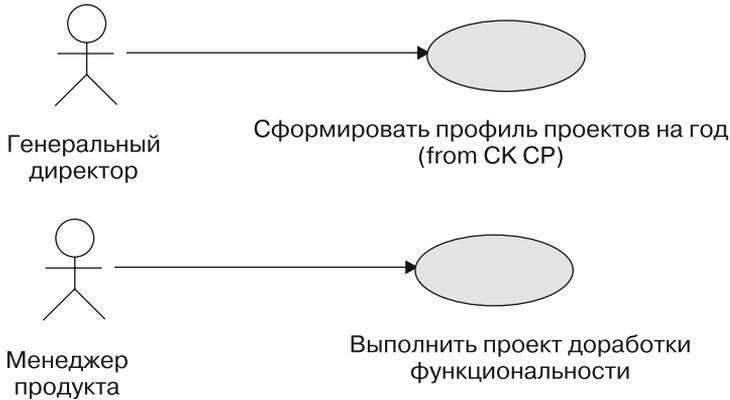
Система управления планированием осуществляет закрытие проекта и подводит результаты выученных уроков.

Система менеджмента качества выполняет совершенствование процедур и артефактов всех СУ, предоставляет сводный анализ использования СУ в Steering

Committee (управляющий комитет) и группу SEPG (группу совершенствования процессов инженерии программного обеспечения). СУП поставляет в эти же органы результаты выученных уроков и обновляет базу активов компании.

## Разработать модель поведения СУПР

Идея состоит в том, что мы рассматриваем СУПР как ИС и, как и в случае с ИС, выявляем акторы и варианты использования ими системы СУПР (рис. 78).



**Рис. 78.** Модель поведения СУПР

С целью дальнейшего определения поведения СУПР требуется построить диаграммы последовательности для выявленных вариантов использования. Сами варианты можно не описывать. Они необходимы лишь для того, чтобы выявить функции разрабатываемой СУПР.



Диаграммы последовательности строятся не только «аналитиком». Здесь техники и практики ее построения ближе всего к механизму достижения согласованных позиций, рассмотренному нами ранее, в подразделе «Политика. Баланс интересов. Переговоры». В результате получается картина взаимодействия «классов» (СУ и их компонентов). Каждая стрелка — какая-то полезная функция/деятельность СУ, которую эта СУ выполняет в СУПР, параметры в методах — какие-то артефакты, которыми оперируют СУ в процессе взаимодействия (рис. 79).



После построения диаграмм последовательности выделяются кандидаты в процедуры систем управления. Отличительной чертой при этом является последовательность деятельностей разных СУ с целью достижения значимого результата, для чего требуется выполнение каждой из этих деятельностей. На диаграммах последовательности очень легко выделять такие наборы деятельностей. На рис. 79 показаны два возможных кандидата в процедуры.

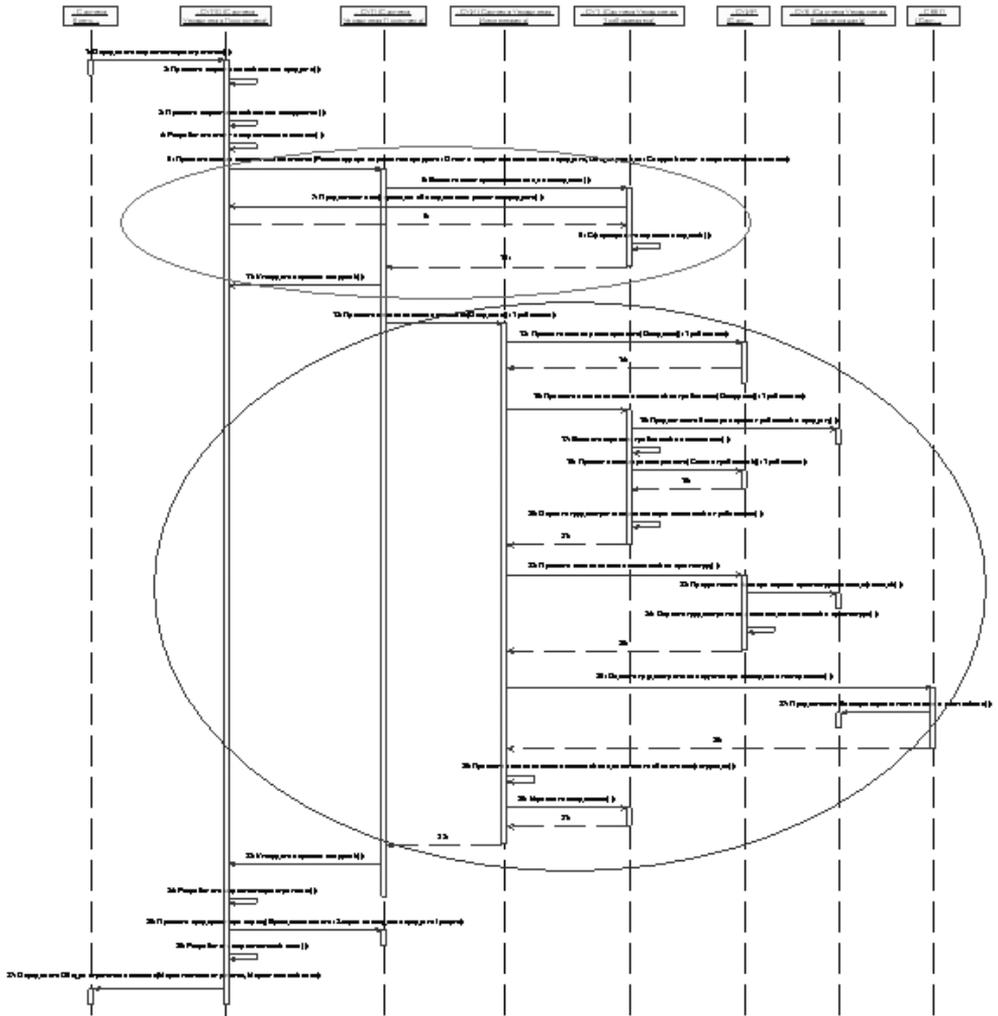


Рис. 79. Коммуникационная модель СУПР

## Определить основные артефакты и процедуры СУПР

На основании диаграмм последовательности разрабатывается логическая модель СУПР (рис. 80).

В этой модели атрибуты соответствуют артефактам СУ, а методы — деятельности этой СУ.



**Рис. 80.** «Логическая» модель СУПР



Применение такого подхода при всей его простоте позволяет быть уверенными в том, что деятельности всех СУ в совокупности покрывают все варианты использования системы СУПР со стороны основных управляющих позиций (акторов СУПР) компании. Кроме того, если команда, отвечающая за процессный менеджмент, владеет UML, то процесс построения такой модели — очень быстрый и гибкий.

Дополнительное преимущество — при желании внести какие-то изменения в СУПР можно провести полноценный импакт-анализ, что в конечном итоге приводит к управляемым, достижимым и действительно нужным изменениям СУПР. Кроме того, наличие логической модели и диаграмм последовательности позволяет не терять контекст и учитывать взаимодействия и интерфейсы с другими классами (СУ) при дальнейшей детализации методов классов, то есть при фактическом создании процедур и рабочих инструкций соответствующих систем управления. Все это дает возможность создать гармоничную и целостную СУПР.



## 6.3. Типичные проблемы и вопросы

В этом разделе приведены наиболее типичные проблемы и вопросы, с которыми сталкивается начальник отдела.

Вопрос/ проблема	Рекомендация/комментарий
Кому нужен мой отдел и зачем?	<p>Осмотритесь. Постарайтесь понять конфигурацию этой компании. Корректируйте все методы управления и взаимодействия в соответствии с конфигурацией организации.</p> <p>Разработайте положение об отделе, в котором зафиксируйте ответ на этот вопрос.</p> <p>Разработайте четкие и понятные должностные инструкции аналитика и начальника аналитического отдела.</p> <p>Утвердите все эти регламенты у руководства</p>
Кто все эти люди?	<p>Проведите очные персональные встречи с каждым из ваших подчиненных, узнайте из первых уст о том, что они делают, какова основная цель их деятельности, какие проблемы у них есть, каковы пожелания для улучшения работы и процессов.</p> <p>Развивайте корпоративную культуру в своем подразделении.</p> <p>Установите измеримые KPI для своих сотрудников, используйте S.M.A.R.T.-цели, сделайте прозрачными и понятными ожидания от ваших подчиненных.</p> <p>Проработайте механизм делегирования конфликтов в производственных проектах. Включите его в базовые правила работы отдела</p>
Институцировать или не институцировать?	<p>Институция (institution) – процесс формального описания процессов в виде процедур, рабочих инструкций и регламентов.</p> <p>Я рекомендую обязательно иметь следующие процессные регламенты и артефакты:</p> <ul style="list-style-type: none"> <li>• базовые правила работы отдела;</li> <li>• должностная инструкция аналитика;</li> <li>• должностная инструкция начальника отдела;</li> <li>• шаблон ПУТ;</li> <li>• шаблон ПУД;</li> <li>• WBS работ аналитика в производственном проекте;</li> <li>• РИ «Согласование и утверждение проектных артефактов»;</li> <li>• реестр артефактов аналитического отдела;</li> </ul>

Вопрос/ проблема	Рекомендация/комментарий
	<ul style="list-style-type: none"> <li>• реестр проблем и открытых вопросов;</li> <li>• реестр выученных уроков и хороших практик.</li> </ul> <p>Остальные процессные регламенты я рекомендую разрабатывать, только если в компании существует СМК и практикуется ее развитие и постоянное совершенствование</p>
Мода и «поджигатели»	<p>Время от времени в разработке ПО возникает мода на использование каких-либо новых методологий. Почти всегда находится человек с «горячим взором», который утверждает, что только применение таких новшеств «спасет всех» и «сделает счастливыми».</p> <p>Я рекомендую всегда поддерживать подобные инициативы, но применять методику DAR и метод многокритериальных оценок для принятия управленческих решений. Естественно, что процесс принятия решения должен быть прозрачен для инициатора таких изменений.</p> <p>Как правило, изменения касаются многих заинтересованных лиц, поэтому необходимо выявлять их и вовлекать в процесс принятия решения и пилотирования новых практик. Для управления изменениями я настойчиво рекомендую пользоваться механизмом достижения согласованных позиций, который рассмотрен в данной главе</p>
Кто виноват и что делать?	<p>На эти извечные вопросы можно дать ответ, только выстраивая систему управления с обратной связью. Такая система повторяет цикл PDCA (plan – do – check – act):</p> <p>P – WBS аналитических работ и методология анализа;</p> <p>D – выполнение работ в соответствии с WBS в проекте с использованием существующей методологии анализа;</p> <p>C – выученные уроки по итогам выполнения проекта;</p> <p>A – изменения в WBS и методологии анализа с целью учесть выявленные хорошие практики.</p> <p>Ведение WAS в отделе позволит вам проводить анализ статистики по проектам и по сотрудникам для принятия решений, основанных на объективных цифрах, а не на интуиции. Например, вы сможете получать ответы на такие вопросы:</p> <ul style="list-style-type: none"> <li>• каков средний процент конкретных типов аналитических работ в проектах типа А? Скажем, «на создание Use Case-модели уходит в среднем 30 % времени работы аналитика в проекте».</li> </ul>

(Продолжение)

Вопрос/ проблема	Рекомендация/комментарий
	<p>Такая информация будет крайне полезна для оценок трудоемкости в будущем и для принятия решений о возможном сокращении трудозатрат и времени выполнения проекта при снижении уровня требований к качеству на основе статистической информации;</p> <ul style="list-style-type: none"> <li>• каков процент типов аналитических работ в конкретном проекте у каждого сотрудника? Например, «Иванов потратил на создание Use Case-модели 40 % времени работы в проекте, при этом общее время аналитических работ проекта на 15 % больше среднего». Согласитесь, есть о чем подумать. Попытаться выяснить причины и, возможно, что-то изменить в процессах или профессиональной подготовке сотрудника Иванова;</li> <li>• каков процент работ отдела по проектам/продуктам? Эти трудозатраты должны соответствовать приоритетам проектов в портфеле проектов и т. д.</li> </ul> <p>Естественно, для того, чтобы информация была достоверной, очень важно объяснить вашим сотрудникам, что от них требуется реальная информация об их трудозатратах в WAS и зачем это надо</p>
Что дальше?	<p>Дальше вы можете развиваться в сторону старшего и высшего менеджмента, например получить степень MBA/MBI.</p> <p>Сама по себе эта степень не сделает вас высшим менеджером по мановению волшебной палочки, но даст вам все необходимые знания, чтобы вы могли выполнять более сложную работу и занимать более ответственные позиции в компании</p>

## 6.4. Заключение

В этой главе мы рассмотрели вопросы организации командной работы, оригинальную методику использования аналитических методов в процессном менеджменте и, самое главное, разобрались, что такое базовые правила работы отдела и какую информацию лучше включить в них; поговорили о стиле управления, основанном на лидерстве, о политике и о том, как можно эффективно сбалансировать интересы разных заинтересованных лиц; коснулись методов анализа таблицы ролей и ответственностей (RACI), принципов и методов принятия управленческих решений; проанализировали конкретные практики и механизмы лидерства и самомотивации.

Для более глубокого усвоения информации рекомендую вам прочитать все источники (книги и статьи), на которые я ссылался в этой главе, а также ознакомиться с другими материалами, указанными в квалификационной таблице навыков (табл. 11).

Таблица 11. Профессиональные и специальные навыки начальника отдела

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
<p><b>Начальник отдела</b></p>	<p>Книги:</p> <p>Capability Maturity Model for Software. SEI;</p> <p><i>Mulcahy Rita</i>. PMP® Exam Prep. PMP;</p> <p><i>De Carlo Doug</i>. eXtreme Project Management;</p> <p><i>Минцберг Г.</i> Структура в кулаке: создание эффективной организации;</p> <p><i>Адизес Ицхак</i>. Идеальный руководитель;</p> <p><i>Фиорина Карли</i>. Трудный выбор. Уроки бескомпромиссного лидерства в сложных ситуациях от экс-главы Hewlett-Packard.</p> <p>Рекомендую регулярно посещать сайты: <a href="http://www.iteam.ru/">http://www.iteam.ru/</a>; <a href="http://msdn.microsoft.com/en-us/architecture/default.aspx">http://msdn.microsoft.com/en-us/architecture/default.aspx</a>; <a href="http://www.cecsi.ru">http://www.cecsi.ru</a>.</p> <p>Рекомендую зарегистрироваться в сети профессиональных контактов (например, <a href="http://moikrug.ru">moikrug.ru</a>, <a href="http://linkedin.com">linkedin.com</a>), заниматься просветительской деятельностью, участвовать в конференциях, форумах</p>	<p>Все навыки аналитика, а также:</p> <ul style="list-style-type: none"> <li>• знать модель зрелости процессов компании CMMI в областях: RD, REQM, DAR, TS, PI, VER, RSKM, PP, PMC, IPM, VAL, QPM, SAM;</li> <li>• уметь анализировать эти области на предмет требуемых улучшений и несоответствий с моделью;</li> <li>• разрабатывать методологию системного анализа;</li> <li>• проводить тренинги и семинары;</li> <li>• иметь четкое представление об управлении проектом/программой проектов;</li> <li>• уметь строить и развивать команду аналитиков в проектах;</li> <li>• уметь предотвращать и разрешать конфликты в проектах;</li> <li>• уметь выявлять аналитические риски и управлять ими;</li> <li>• проводить выученные уроки по результатам выполненных работ/проектов;</li> <li>• участвовать в совершенствовании процессов;</li> <li>• разрабатывать процедуры, регламенты, рабочие инструкции;</li> <li>• создавать функциональную стратегию своего направления;</li> <li>• планировать развитие отдела, вовлекать топ-менеджеров в решение стратегических и тактических вопросов;</li> <li>• уметь строить и развивать команду;</li> <li>• строить эффективное взаимодействие с другими подразделениями;</li> </ul>

Продолжение ⇨

*(Продолжение)*

Квалификация	Необходимая теория и профессиональные навыки	Необходимые специальные, лидерские и управленческие навыки
		<ul style="list-style-type: none"><li>• разрешать конфликты на всех уровнях;</li><li>• уметь управлять проектом;</li><li>• профессионально развивать подчиненных;</li><li>• уметь проводить аттестацию сотрудника;</li><li>• курировать создание базы знаний отдела;</li><li>• управлять совершенствованием процессов в области анализа, разработки и управления требованиями;</li><li>• управлять формализацией процессов и созданием СМК отдела;</li><li>• разрабатывать процедуры, регламенты, должностные и рабочие инструкции</li></ul>

# 7. Итак...

---

---

Мы с вами прошли длинный путь и рассмотрели хорошие практики, приемы и конкретные примеры для всех уровней профессии аналитика — от начинающего аналитика до начальника отдела.

Надеемся, что рекомендации окажутся полезными и книга поможет вам не повторять наших ошибок. Мы будем очень рады, если вы сможете выстроить свою профессиональную карьеру максимально эффективно.

Возможно, эта книга откроет целую серию «Путь аналитика», по крайней мере, у нас есть идеи, которые заслуживают отдельного и достаточно детального изложения. Так что, возможно, мы с вами еще встретимся на страницах следующей книги. Ну а пока...

Вы закрываете последнюю страницу книги «Путь аналитика. Практическое руководство IT-специалиста».

Спасибо за вашу работу.

Удачи.

И легкого вам Пути.

Вы можете написать ваше мнение, вопросы и пожелания непосредственно авторам книги по e-mail: [a\\_way@mail.ru](mailto:a_way@mail.ru) либо на сайте <http://saway4ru.codeplex.com>.

# Приложение

---

---

## Запросы заинтересованного лица

Введение

Цель документа

В этом разделе описывается цель создания документа.

*Пример*

Целью данного документа является описание проблем заинтересованного лица для дальнейшего анализа и выявления требований к системе «Название системы».

Рамки документа

В данном разделе документа устанавливаются рамки документа: указываются причина появления данного документа, его роль в будущем, другие документы, которые должны появиться на базе этого документа, кто и что должен сделать в связи с появлением текущего документа.

*Пример*

Настоящий документ описывает запросы конкретного заинтересованного лица, которые являются результатом проведения с ним одного или нескольких интервью. Документ будет использоваться при разработке требований к системе «Название системы». Документ не утверждается и не согласовывается.

Целевая аудитория документа

Здесь описывается целевая аудитория документа.

*Пример*

Документ предназначен для бизнес- и системных аналитиков, которые участвуют в сборе и анализе требований.

## Материалы

Необязательный раздел.

В данном разделе приводятся ссылки на различные документы, с которыми полезно ознакомиться перед изучением данного документа.

## Описание заинтересованного лица или пользователя

В этом разделе необходимо дать краткую характеристику заинтересованного лица или пользователя.

Предлагается ответить на следующие вопросы.

1. Имя... Компания...
2. Должность...
3. Каковы ваши ключевые обязанности?
4. Что является результатом вашей работы с материальной точки зрения (изделие, программный код, документ и т. п.), кто им пользуется?
5. Каковы критерии успеха вашей работы?
6. Какие проблемы являются помехой для успешной работы?
7. Какие, если есть, отклонения делают вашу работу сложнее или легче?

## Исследование проблем заинтересованного лица

В этом разделе дается описание выявленных задач и проблем в работе заинтересованного лица.

Предлагается ответить на следующие вопросы.

1. Для решения каких задач вам требуются инструментальные средства?
2. Что это за задачи? (Подсказка – продолжайте спрашивать: «Что-нибудь еще?»)

Для каждой задачи уточните следующее.

1. Какая проблема возникла при решении задачи и почему?
2. Как вы сейчас ее решаете?
3. Как вы хотели бы ее решать?

## Понимание рабочего окружения пользователя

Предлагается ответить на следующие вопросы.

1. Кто является пользователем?
2. Каков их образовательный уровень?

3. Каков их уровень владения компьютером?
4. Обладают ли пользователи опытом работы с подобными приложениями?
5. Какие платформы используются? Какие планируется использовать?
6. Интерфейс с какими дополнительными приложениями необходим?
7. Каковы ваши ожидания относительно удобства пользования продуктом?
8. Каковы ваши ожидания относительно времени обучения, необходимого для пользования продуктом?
9. Какая бумажная и онлайн-документация требуется?

#### Подтверждение понимания проблем

Предлагается ответить на следующие вопросы.

1. Вы сказали мне (список описанных заинтересованным лицом проблем своими словами): ...
2. Правильно ли это отражает проблемы, возникающие при использовании существующего решения?
3. Какие, если есть, другие проблемы были обнаружены?

#### Вклад аналитика в определение проблем заинтересованных лиц (подтвердить или опровергнуть предположения)

Предлагается ответить на следующие вопросы.

1. (В этом разделе опишите проблемы, которые не связаны с заинтересованным лицом, но которые, по вашему мнению, могут его касаться).

Список любых потребностей и проблем, которые могут касаться заинтересованного лица: ...

2. Для каждой предложенной проблемы выяснить следующее.
  1. Является ли эта проблема реальной?
  2. Каковы ее причины?
  3. Как вы решаете данную проблему?
  4. Как хотели бы решать?
  5. Какой бы вы выставили приоритет решения данной проблемы по сравнению с ранее указанными?

#### Исследование (оценка) предлагаемого решения (если применимо)

Предлагается ответить на следующие вопросы.

1. Что было бы, если бы вы (резюмируйте ключевые возможности предлагаемого решения) ... ?
2. Какой бы вы выставили приоритет важности этого?

Исследование (оценка) перспектив

Предлагается ответить на следующие вопросы.

1. Кто является потребителем обсуждаемого приложения в вашей организации?
2. Как много пользователей разных типов будут использовать приложение?
3. Какова для вас ценность успешного решения?

Исследование (оценка) надежности, производительности и поддержки

Предлагается ответить на следующие вопросы.

1. Каковы ваши ожидания относительно надежности (отказоустойчивости)?
2. Каковы ваши ожидания относительно производительности?
3. Кто будет заниматься поддержкой продукта?
4. Есть ли специфичные потребности в поддержке? Есть ли требования по техническому обслуживанию?
5. Каковы требования к безопасности?
6. Каковы требования к процессу инсталляции и конфигурирования?
7. Есть ли специфичные лицензионные требования?
8. Какими способами будет распространяться ПО?
9. Каковы требования к оформлению продукта (логотипы и т. п.)?

Другие требования

Предлагается ответить на следующие вопросы.

1. Каковы, если есть, законодательные либо дополнительные требования или необходимые для поддержки стандарты?
2. Можете ли вы вспомнить другие требования, о которых следует сказать?

Резюме

Предлагается ответить на следующие вопросы.

1. Есть ли еще вопросы, которые необходимо обсудить?
2. Если мне потребуются разъяснения, могу ли я позвонить?
3. Могли бы вы принять участие в рецензировании требований?

Сводка аналитика

Приведите резюме 3–4 самых высокоприоритетных проблем для данного пользователя/заинтересованного лица.

# Дополнительные материалы

---

---

На сайте книги вы найдете дополнительные материалы, перечисленные в порядке упоминания в книге:

- ◆ шаблон «Протокол совещания»;
- ◆ шаблон «Запросы заинтересованного лица»;
- ◆ шаблон «Концепция системы»;
- ◆ специальный скрипт для Visio, который позволяет специфицировать GUI-элементы непосредственно в Visio, сохраняя комментарий/требование в базе данных MS Access;
- ◆ конспект «UML и Rational Unified Process»;
- ◆ презентация «Возможности Rational SoDA и полезные шаблоны»;
- ◆ шаблоны SoDA для генерации спецификаций требований;
- ◆ шаблон плана управления требованиями для водопадного жизненного цикла разработки;
- ◆ рекомендации по проведению презентаций;
- ◆ выбор ЖЦ проектов в зависимости от характеристик системы и требований к качеству продукта;
- ◆ шаблон плана управления документами.

# Использованная литература

---

---

1. Тернер М. Основы Microsoft Solution Framework. — М.: Русская редакция; СПб.: Питер, 2009.
2. Comparing the Rational Unified Process (RUP) and Microsoft Solutions Framework (MSF): <http://www.ibm.com/developerworks/rational/library/apr07/santos/index.html>.
3. IASA IT Architecture Glossary: <http://www.iasahome.org/web/home/ITarchitecture/Glossary>.
4. *Eeles P.* Capturing Architectural Requirements: <http://www.ibm.com/developerworks/rational/library/4706.html>.
5. Rational Unified Process (RUP) 2003.
6. *Minoli D.* Enterprise architecture A to Z: frameworks, business process modeling, SOA, and infrastructure technology. — Auerbach Publications Taylor & Francis Group, 2008.
7. [http://www.ittoday.info/Articles/What\\_Is\\_Enterprise\\_Architecture.htm](http://www.ittoday.info/Articles/What_Is_Enterprise_Architecture.htm).
8. Вигерс К. И. Разработка требований к программному обеспечению. — М.: Русская редакция, 2004.
9. *Rosenberg D., Kendall S.* Applying Use Case Driven Object Modeling with UML. — Addison-Wesley, 2001.
10. CHAOS Chronicles v. 3.0, The Standish Group Int'l, 2003: <http://www.standishgroup.com/chaos/toc.php>.
11. *McGrath M.* Next Generation Product Development: How to Increase Productivity, Cut Costs, and Reduce Cycle Times, McGraw-Hill, 2004.
12. *Giesen J., Voelker A.* Requirements Interdependencies and Stakeholder Preferences, Proc. Int'l Conf. Requirements Eng. (RE 02). IEEE CS Press, 2002.

13. *Ebert C., Dumke R., Bundschuh M., Schmietendorf A.* Best Practices in Software Measurement. — Springer, 2004.
14. *Boehm Barry W.* A Spiral Model of Software Development and Enhancement. IEEE Computer 21, 15 (May 1988).
15. *Doug DeCarlo.* eXtreme project management: using leadership, principles, and tools to deliver value in the face of volatility. Published by Jossey-Bass. — A Wiley Imprint, 2004.
16. *Мициберг Г.* Структура в кулаке: создание эффективной организации. — СПб., 2001.
17. *Фиорина Карли.* Трудный выбор. Уроки бескомпромиссного лидерства в сложных ситуациях от экс-главы Hewlett-Packard / Пер. с англ. Э. В. Кондуковой. — М.: Эксмо, 2009.
18. SEI. Capability Maturity Model for Software. Version 1.1, Document No. CMU/SEI-93-TR-25, ESC-TR-93-178. — Pittsburgh, PA: Carnegie-Mellon University Software Engineering Institute, 1993.
19. *Chrissis M., Konrad M., Shrum C.* CMMI: Guidelines for Process Integration and Product Improvement. — Addison-Wesley, 2003.
20. *Ebert C.* Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques. IEEE Software, 2006, Volume 23, Issue 3.
21. *Leffingwell D., Widrig D.* Managing Software Requirements: A Use Case Approach (2nd Edition). — Addison-Wisley Professional, 2003. — 544 p.

# Использованные интернет-ресурсы

---

---

1. Центр предпринимательского творчества и системных инноваций Вадима Котельникова: <http://www.cecsi.ru/>
2. [http://www.ittoday.info/Articles/What\\_Is\\_Enterprise\\_Architecture.htm](http://www.ittoday.info/Articles/What_Is_Enterprise_Architecture.htm)
3. <http://www-01.ibm.com/software/awdtools/rup/>
4. <http://www.iasahome.org/web/home/ITarchitecture/Glossary>
5. <http://www.standishgroup.com/chaos/toc.php>
6. <http://www.ibm.com/developerworks/rational/library/apr07/santos/index.html>
7. <http://www.apkit.ru/files/analitik.doc>
8. <http://www-01.ibm.com/software/awdtools/rmc/library/>
9. <http://www.agilerussia.ru/>
10. <http://www-01.ibm.com/software/awdtools/rmc/library/>
11. <http://www.ibm.com/developerworks/rational/practices/>
12. <http://ru.wikipedia.org/wiki/UML>
13. <http://www.interface.ru>
14. Iconix Process Overview: <http://iconixprocess.com/iconix-process/>; <http://www.iconixsw.com/>; <http://iconixprocess.com/?s=architect>

# Глоссарий

---

---

ИТ	Информационные технологии	<p>Информационные технологии (ИТ, от англ. information technology, ИТ) — широкий класс дисциплин и областей деятельности, относящихся к технологиям управления и обработки данных, а также создания данных, в том числе с применением вычислительной техники.</p> <p><a href="http://ru.wikipedia.org/wiki/Информационные_технологии">http://ru.wikipedia.org/wiki/Информационные_технологии</a></p>
ИС	Информационная система	<p>Информационная система (ИС) — система хранения, обработки и передачи информации, представленной в определенной форме.</p> <p>В широком смысле информационная система есть совокупность технического, программного и организационного обеспечения, а также персонала, предназначенная для того, чтобы своевременно обеспечивать надлежащих людей надлежащей информацией.</p> <p><a href="http://ru.wikipedia.org/wiki/Информационная_система">http://ru.wikipedia.org/wiki/Информационная_система</a></p>
ООП	Объектно-ориентированное проектирование	<p>Подход к проектированию информационных систем на основании определения объектов информационных систем, которые являются в определенном смысле проекциями объектов реального мира/предметной области, для автоматизации которой разрабатывается информационная система.</p> <p>Объектно-ориентированное проектирование (ООП) — это часть объектно-ориентированной методологии, которая предоставляет возможность программистам оперировать понятием «объект», а не понятием «процедура» при разработке своего кода. Объекты содержат инкапсулированные данные и процедуры, сгруппированные вместе, отображая таким образом сущность объекта. «Интерфейс объекта» описывает</p>

Продолжение ⇨

(Продолжение)

		<p>взаимодействие с объектом, то, как он определен. Программа, полученная при реализации объектно-ориентированного исходного кода, описывает взаимодействие этих объектов.</p> <p><a href="http://ru.wikipedia.org/wiki/Объектно-ориентированное_проектирование">http://ru.wikipedia.org/wiki/Объектно-ориентированное_проектирование</a></p>
ТЗ	Техническое задание	<p>Техническое задание (ТЗ, техзадание) — исходный документ для проектирования сооружения или промышленного комплекса, конструирования технического устройства (прибора, машины, системы управления и т. д.), разработки информационных систем, стандартов либо проведения научно-исследовательских работ (НИР).</p> <p>ТЗ содержит основные технические требования, предъявляемые к сооружению, изделию или услуге, и исходные данные для разработки; в ТЗ указываются назначение объекта, область его применения, стадии разработки конструкторской (проектной, технологической, программной и т. п.) документации, ее состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов предварительных исследований, расчетов и моделирования.</p> <p><a href="http://ru.wikipedia.org/wiki/Техническое_задание">http://ru.wikipedia.org/wiki/Техническое_задание</a></p>
ПУТ	План управления требованиями	<p>План управления требованиями (ПУТ) — описание подходов к разработке и управлению требованиями, которые будут применяться в конкретном проекте: какие требования будут создаваться, кем и когда, как они будут связаны между собой и с другими проектными артефактами; как будет производиться управление состояниями требований, какие полномочия в управлении у каких ролей и т. д. и т. п.</p> <p>ПУТ входит в состав плана управления проектом (ПУП) и должен находиться в соответствии с планом управления документами (ПУД) проекта</p>
ПУП	План управления проектом	<p>План управления проектом (ПУП) — основной документ для управления проектом, описывает подходы, которые будут применяться для управления конкретным проектом. Включает в себя несколько планов управления для разных областей: управление требованиями, коммуникациями, рисками, качеством и т. д. и т. п.</p>

ПУД	План управления документами	План управления документами (ПУД) – описывает, какие проектные артефакты на каком этапе проекта в каком состоянии должны находиться, какие роли являются согласующими лицами для каждого артефакта, какие роли входят в команду утверждения, какая роль является ответственной за артефакт, даты готовности артефактов и т. д. и т. п.
БД	База данных	База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных. <a href="http://ru.wikipedia.org/wiki/База_данных">http://ru.wikipedia.org/wiki/База_данных</a>
ПО	Программное обеспечение	Совокупность программ, процедур и правил, а также документации, относящихся к функционированию системы обработки данных (СТ ИСО 2382/1-84). <a href="http://ru.wikipedia.org/wiki/Программное_обеспечение">http://ru.wikipedia.org/wiki/Программное_обеспечение</a>
ОС	Операционная система	Операционная система, сокр. ОС (англ. operating system) – комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой – предназначены для управления устройствами, вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надежных вычислений. Это определение применимо к большинству современных ОС общего назначения. <a href="http://ru.wikipedia.org/wiki/Операционная_система">http://ru.wikipedia.org/wiki/Операционная_система</a>
	Спонсор проекта	Спонсор (куратор) проекта – сотрудник (как правило, руководитель высшего звена) организации, реализующей проект, который курирует проект со стороны организации (владельца проекта), обеспечивает общий контроль и поддержку проекта (финансовые, материальные, человеческие и другие ресурсы). Спонсор (куратор) проекта отвечает за достижение проектом конечных целей и реализацию выгод для организации. Спонсор проекта несет ответственность перед генеральным директором/президентом или перед управляющим советом

(Продолжение)

		<p>Спонсор проекта назначает менеджера проекта и обеспечивает ему необходимую поддержку.</p> <p><a href="http://www.pmpractice.ru/knowledgebase/managment/keypoints/participants/">http://www.pmpractice.ru/knowledgebase/managment/keypoints/participants/</a></p>
ЗЛ	Заинтересованное лицо	<p>Заинтересованное лицо — это человек или организация, которые могут повлиять или на которых может повлиять результат определенных действий (в контексте книги — это разработка ИС, процесс выполнения проекта, результат выполнения проекта) в целом</p>
ЖЦ	Жизненный цикл	<p>В книге под ЖЦ понимается ЖЦ ИС. Жизненный цикл информационной системы — период времени, который начинается с момента принятия решения о необходимости создания информационной системы и заканчивается в момент ее полного вывода из эксплуатации.</p> <p><a href="http://ru.wikipedia.org/wiki/Жизненный_цикл">http://ru.wikipedia.org/wiki/Жизненный_цикл</a></p>
МП	Менеджер проекта	<p>Менеджер проектов — это специалист в области управления проектами, который несет ответственность за планирование, подготовку и исполнение конкретного проекта. Обычно такая должность и деятельность имеют место в отраслях, склонных к проектным работам, например: IT-области, строительство и др.</p> <p><a href="http://ru.wikipedia.org/wiki/Менеджер_проектов">http://ru.wikipedia.org/wiki/Менеджер_проектов</a></p>
ООАП	Объектно-ориентированный анализ и проектирование	См. ООП
МПП	Менеджер продукта	<p>Менеджер продукта — это специалист в области управления продуктами, который несет ответственность за планирование, подготовку и развитие конкретного продукта. В ИТ менеджер продукта достаточно часто не отвечает за распространение и продажи продукта, а скорее отвечает за стратегию его развития, находясь на стыке коммуникаций с менеджерами по продажам/ по обслуживанию клиентов, с одной стороны, и менеджерами проектов по разработке продукта, с другой стороны</p>

	Проект требований	Часть общей технологической инфраструктуры проекта, предназначенная для хранения и управления требованиями в соответствии с утвержденным планом управления требованиями (см. ПУТ)
	Спецификация требований	<p>Спецификация требований программного обеспечения (Software Requirements Specification) – законченное описание поведения системы, которую требуется разработать.</p> <p>Включает ряд пользовательских сценариев (Use cases), которые описывают все варианты взаимодействия между пользователями и программным обеспечением.</p> <p>Пользовательские сценарии являются средством представления функциональных требований. В дополнение к пользовательским сценариям спецификация также содержит нефункциональные требования, которые налагают ограничения на дизайн или реализацию (такие как требования производительности, стандарты качества или проектные ограничения).</p> <p><a href="http://ru.wikipedia.org/wiki/Спецификация_программного_обеспечения">http://ru.wikipedia.org/wiki/Спецификация_программного_обеспечения</a></p> <p>В более широком смысле спецификация требований – документ с требованиями или моделями, который используется в процессе разработки и управления требованиями в проекте</p>
	Драфт документа	Версия документа, нуждающаяся в ревью (рецензировании), согласовании или утверждении
	Статус-митинг	Совещание, на котором проектная команда определяет состояние работ проекта, анализирует проблемы и пути их решения, выявляет новые риски и пути их минимизации и предотвращения
АИС	Автоматизированная информационная система	<p>Автоматизированная информационная система (АИС) – совокупность программно-аппаратных средств, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации.</p> <p>АИС являются, с одной стороны, разновидностью информационных систем (ИС), с другой – автоматизированных систем (АС), вследствие чего их часто называют ИС или АС</p>

(Продолжение)

СУ	Система управления	<p>Под системой управления (СУ) понимается совокупность процедур, рабочих инструкций, методологий, инструментов, АИС и навыков сотрудников компании, работающих как единое целое и позволяющих за счет синергетического эффекта от их совместного использования эффективно выполнять производственную деятельность в какой-либо области знаний (дисциплине).</p> <p>Или, проще говоря, СУ – это совокупность организационной структуры, методик, процессов и ресурсов, необходимых для достижения целей управления</p>
СМК	Система менеджмента качества	<p>Система менеджмента качества (СМК) – система управления качеством производимой продукции в какой-либо организации.</p> <p>Это задокументированный «образ» предприятия как организма, то есть саморегулирующегося механизма, приспособленного к жизни в конкретной экономической среде.</p> <p><a href="http://ru.wikipedia.org/wiki/Система_менеджмента_качества">http://ru.wikipedia.org/wiki/Система_менеджмента_качества</a></p>
СУПР	Система управления производством	СУ производством компании объединяет в себе все остальные СУ по областям деятельности компании и является составной частью системы менеджмента качества компании
СУАР	Система управления архитектурой	СУ, позволяющая эффективно выполнять производственную деятельность в области проектирования и разработки архитектуры
СУТ	Система управления требованиями	СУ, позволяющая эффективно выполнять производственную деятельность в области разработки и управления требованиями
СУП	Система управления проектами	СУ, позволяющая эффективно выполнять производственную деятельность в области управления проектами
СККП	Система контроля качества продуктов	СУ, позволяющая эффективно выполнять производственную деятельность в области контроля качества продуктов
СУИК	Система управления изменениями	СУ, позволяющая эффективно выполнять производственную деятельность в области управления изменениями

РИ	Рабочая инструкция	Рабочая инструкция — пошаговая инструкция для одной или нескольких ролей компании с целью выполнения какой-либо деятельности. Чаще всего рабочие инструкции стараются писать для одной роли/должности, объединяя несколько РИ в одну процедуру. Целостный процесс, в свою очередь, описывается через совокупность процедур. Все эти описания являются частью СМК компании
ERP	Enterprise Resource Planning System — система планирования ресурсов предприятия	ERP-система (англ. <b>Enterprise Resource Planning System</b> — система планирования ресурсов предприятия) — это интегрированная система на базе ИТ для управления внутренними и внешними ресурсами предприятия (значимые физические активы, финансовые, материально-технические и человеческие ресурсы). Цель системы — содействие потокам информации между всеми хозяйственными подразделениями (бизнес-функциями) внутри предприятия и информационная поддержка связей с другими предприятиями. Построенная, как правило, на централизованной базе данных, ERP-система формирует стандартизованное единое информационное пространство предприятия. <a href="http://ru.wikipedia.org/wiki/ERP">http://ru.wikipedia.org/wiki/ERP</a>
CRM	Customer Relationship Management System — система управления взаимодействием с клиентами	Система управления взаимодействием с клиентами (или CRM, сокр. от англ. Customer Relationship Management System — система управления взаимодействием с клиентами) — корпоративная информационная система, предназначенная для автоматизации CRM-стратегии компании, в частности для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путем сохранения информации о клиентах (контрагентах) и истории взаимоотношений с ними, установления и улучшения бизнес-процедур и последующего анализа результатов. Под термином «CRM-система» понимается программный продукт (ПО), направленный на реализацию концепции CRM. <a href="http://ru.wikipedia.org/wiki/CRM">http://ru.wikipedia.org/wiki/CRM</a>
Enterprise Architecture	Архитектура предприятия/ корпоративная архитектура	Это высокоуровневая архитектура всего предприятия, покрывающая бизнес-потребности ИТ-способностями. Корпоративная архитектура фокусируется на определении потоков и бизнес-процессов, действий, функций, информации, данных и технологий предприятия и на вызовах, стоящих перед ИТ, необходимых для того, чтобы эффективно применить технологию в ответ на изменение бизнес-потребностей

(Продолжение)

Business Architecture	Бизнес-архитектура	Описывает все бизнес-процессы, бизнес-акторы, бизнес-сущности и бизнес-правила с точки зрения бизнеса. Бизнес-архитектура не зависит от применяемых в разработке технологий
Information Architecture	Информационная архитектура	Определяет структуры данных и описывает все потоки данных, которые используются для поддержки бизнес-архитектуры. Такие операции, как идентификация, систематизация, категоризация, хранение данных, относятся к информационной архитектуре (Information Architecture). Может представляться в виде модели данных (Data Model)
Solution (System/ Application) Architecture	Архитектура решения	Архитектура программного обеспечения, которое реализует функции бизнес-архитектуры (Business Architecture)
Technology Architecture	Технологическая архитектура	Описывает архитектуру ИТ-окружения, которое используется для поддержки информационной архитектуры (Information Architecture) и архитектуры решения (Solution (System/ Application) Architecture)
System Architecture	Системная архитектура	Это представление системы, которое показывает реализацию функциональных возможностей системы аппаратными средствами и компонентами программного обеспечения, устанавливает связь архитектуры программного обеспечения и архитектуры аппаратных средств, а также регламентирует взаимодействие пользователя с этими компонентами. Существуют и другие определения системной архитектуры (System Architecture), например: ряд взаимосвязанных шаблонов (паттернов), которые структурируют модули и данные и обеспечивают требуемое поведение системы (см. определение Data Architecture). Системная архитектура (System Architecture) является составной частью архитектуры решения (Solution Architecture)
Software Architecture	Архитектура программного обеспечения	Является составной частью системной архитектуры (System Architecture). Описывает организацию системы с точки зрения программных компонентов, из которых она состоит, и связи между компонентами
Data Architecture	Архитектура данных	Является составной частью системной архитектуры (System Architecture). Описывает структуры данных и логические связи между данными

Hierarchical model	Иерархическая модель	Объекты этой модели организованы через отношения «родитель – ребенок» и составляют иерархию объектов
Network model	Сетевая модель	Объекты этой модели чаще всего связаны в виде графа. Все связи обычно имеют одинаковые характеристики с разными значениями. Хороший пример – сетевая диаграмма проекта в проектном управлении
Relational model	Реляционная (взаимосвязанная) модель	Наиболее распространенная в настоящее время для проектирования реляционных баз данных модель. Объектами модели являются структуры данных, которые связаны между собой разными типами связей
Object model	Объектная модель	См. также ООП
CMMI	Capability Maturity Model Integration (CMMI) – набор моделей (методологий) совершенствования процессов в организациях	Capability Maturity Model Integration (CMMI) – набор моделей (методологий) совершенствования процессов в организациях разных размеров и видов деятельности. CMMI содержит набор рекомендаций в виде практик, реализация которых, по мнению разработчиков модели, позволяет реализовать цели, необходимые для полной реализации определенных областей деятельности. <a href="http://www.sei.cmu.edu/cmmi/">http://www.sei.cmu.edu/cmmi/</a>
CMMI Process Improvement	Улучшение процессов в соответствии с CMMI	См. CMMI. Про метод оценки зрелости процессов по CMMI можно прочитать здесь: <a href="http://en.wikipedia.org/wiki/Standard_CMMI_Appraisal_Method_for_Process_Improvement">http://en.wikipedia.org/wiki/Standard_CMMI_Appraisal_Method_for_Process_Improvement</a>
SWEBoK	Software Engineering Body of Knowledge	SWEBOOK (Software Engineering Body of Knowledge) – документ, подготавливаемый комитетом Software Engineering Coordinating Committee, в который вовлечено сообщество IEEE Computer Society. Назначение SWEBOOK – в объединении знаний по инженерии программного обеспечения (разработке программного обеспечения). <a href="http://ru.wikipedia.org/wiki/SWEBOOK">http://ru.wikipedia.org/wiki/SWEBOOK</a>
Rational Unified Process (RUP)	Rational Unified Process – унифицированный процесс разработки ПО компании Rational	Rational Unified Process – унифицированный процесс разработки ПО компании Rational с однозначно выраженными рекомендациями по разработке ПО, включающими в себя перечень всех необходимых деятельностей, выполняемых проектными ролями на каждой итерации, и шаблонов артефактов. <a href="http://ru.wikipedia.org/wiki/RUP">http://ru.wikipedia.org/wiki/RUP</a>

(Продолжение)

Microsoft Solution Framework (MSF)	Microsoft Solution Framework – руководство по созданию решений от Microsoft	Microsoft Solution Framework (MSF) не является методологией разработки ПО. MSF предлагает подходы, основанные на определенной совокупности принципов, моделей, дисциплин, руководств и методик для проектов различной степени сложности, ориентированных на поставку решений. <a href="http://ru.wikipedia.org/wiki/Microsoft_Solutions_Framework">http://ru.wikipedia.org/wiki/Microsoft_Solutions_Framework</a>
Iconix	Название процесса не является аббревиатурой по разъяснению ICONIX Software	Процесс разработки программного обеспечения с использованием ограниченного количества UML моделей и диаграмм, состоящий из небольших шагов, ведущих к цели. <a href="http://iconixprocess.com/iconix-process/">http://iconixprocess.com/iconix-process/</a>
Спиральная разработка Barry W. Boehm	Спиральная разработка	Спиральная модель (англ. spiralmodel) была разработана в середине 1980-х годов Барри Бозмом. Она основана на классическом цикле Деминга PDCA ( <b>plan-do-check-act</b> ). При использовании этой модели ПО создается в несколько итераций (витков спирали) методом прототипирования. <a href="http://en.wikipedia.org/wiki/Barry_Boehm">http://en.wikipedia.org/wiki/Barry_Boehm</a>
Agile		Гибкая методология разработки программного обеспечения. <a href="http://en.wikipedia.org/wiki/Agile_software_development">http://en.wikipedia.org/wiki/Agile_software_development</a>
Extreme Programming (XP)	Экстремальное программирование	Экстремальное программирование (англ. Extreme Programming, XP) – одна из гибких методологий разработки программного обеспечения. Авторы методологии – Кент Бек, Уорд Каннингем, Мартин Фаулер и др. <a href="http://ru.wikipedia.org/wiki/Extreme_programming">http://ru.wikipedia.org/wiki/Extreme_programming</a>
Robustness diagram	Диаграмма устойчивости	Диаграмма устойчивости, или диаграмма пригодности. Используется в Iconix. <a href="http://iconixprocess.com/iconix-process/analysis-and-preliminary-design/robustness-analysis/">http://iconixprocess.com/iconix-process/analysis-and-preliminary-design/robustness-analysis/</a>
Sequence diagram	Диаграмма последовательности	Диаграмма последовательности. Показывает взаимодействие объектов во времени. Используется в языке UML. <a href="http://en.wikipedia.org/wiki/Sequence_diagram">http://en.wikipedia.org/wiki/Sequence_diagram</a>

Analysis model	Аналитическая модель	Объектная модель, описывающая реализацию вариантов использования. Создается на фазе уточнения (Elaboration) в RUP
Rational Requisite PRO		Инструмент для разработки и управления требованиями. <a href="http://www-01.ibm.com/software/awdtools/reqpro/">http://www-01.ibm.com/software/awdtools/reqpro/</a>
Requirement	Требование	(А) Условие или способность, необходимые пользователю, чтобы решить проблему или достичь цели. (В) Условие или способность, которыми должна обладать система или компонент системы для удовлетворения контракта, стандарта, спецификации или другого формально установленного документа. (С) Документированное представление условия или способности, показанных в определении (А) или (В). © (IEEE Std 610.12-1990)
Unified Modeling Language (UML)	Унифицированный язык моделирования	UML (англ. <b>Unified Modeling Language</b> – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования в основном программных систем. <b>UML не является языком программирования</b> , но в средствах выполнения UML-моделей как интерпретируемого кода возможна кодогенерация. <a href="http://ru.wikipedia.org/wiki/UML">http://ru.wikipedia.org/wiki/UML</a>
Use Case Model	Модель вариантов использования системы/ модель прецедентов системы	Описывает поведение разрабатываемой системы (как пользователи взаимодействуют с системой, и что система делает в ответ на эти взаимодействия). Эта модель описывает функциональные требования в терминах вариантов использования (Use Cases). Используется в RUP

(Продолжение)

Domain model	Модель предметной области	Модель, которая описывает основные сущности предметной области, их взаимосвязи друг с другом и атрибуты сущностей. Используется для выявления, классификации и формализации сведений обо всех аспектах предметной области, определяющих свойства разрабатываемой системы. <a href="http://en.wikipedia.org/wiki/Domain_model">http://en.wikipedia.org/wiki/Domain_model</a>
Business Vision	Бизнес-видение	Концепция создания и развития продукта. См. BVISION
Technical Vision	Техническое видение	Концепция системы. См. TVISION
Scrum team	Команда «регби»	Подход впервые описали Хиротака Такеути и Икудзиро Нонака [1] в статье The New Product Development Game (Гарвардский деловой обзор [2], январь-февраль 1986). Они отметили, что проекты, над которыми работают небольшие кросс-функциональные команды, обычно систематически производят лучшие результаты, и объяснили это как «подход регби». В 1991 году Де Грейс и Шталь в книге «Злые проблемы, справедливые решения» [3] ссылались на этот подход, как на Scrum (толкотня; схватка вокруг мяча (в регби)), спортивный термин, приведенный в статье Такеути и Нонакой
Workflow	Поток работ	Состоит из последовательности взаимосвязанных действий. Используется для моделирования процессов. <a href="http://en.wikipedia.org/wiki/Workflow">http://en.wikipedia.org/wiki/Workflow</a>
Business modeling	Бизнес-моделирование	Совокупность методов, практик и шаблонов для моделирования бизнеса
GUI Storyboard	Прототип пользовательского интерфейса	Раскадровка графических форм пользовательского интерфейса
Dynamic model	Динамическая модель	Описывает поведение разрабатываемой системы. Включает в себя модель вариантов использования (см. Use Case Model), диаграммы устойчивости (см. Robustness Diagram), диаграммы последовательности (см. Sequence Diagram)
Static model	Статическая модель	Описывает сущности системы. Включает в себя модель предметной области (см. Domain Model), диаграмму классов

Test Plan	Тест-план	План тестирования разрабатываемого ПО
Updated Domain Model	Обновленная модель предметной области	
Class Model	Модель классов	Статическая модель, элементами которой являются классы, типы, структуры данных и взаимосвязи между ними. Используется в RUP
Unit Testing	Модульное тестирование	Процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. <a href="http://en.wikipedia.org/wiki/Unit_testing">http://en.wikipedia.org/wiki/Unit_testing</a>
Proposed	Предложено	Значение статуса требования «Предложено». Требование готово для разработки
Incorporated	Реализовано	Значение статуса требования «Реализовано». Требование реализовано в коде
Postponed	Отложено	Значение статуса требования «Отложено». Реализация требования приостановлена до принятия решения
Rejected	Отклонено	Значение статуса требования «Отклонено». Требование не будет реализовываться
WBS	Work Breakdown Structure	Иерархическая структура работ проекта
BREQ	Business Requirements	Тип требования. Требования бизнеса к системе. См. BR
Use Case	Вариант использования системы/прецедент	Составная часть моделирования ИС – функция системы, которая приносит ощутимый и значимый результат актору. См. Вариант использования (UC)
Rational Rose	Имя собственное инструмента	Инструмент для моделирования ИС на языке моделирования UML. <a href="http://www-01.ibm.com/software/awdtools/developer/rose/">http://www-01.ibm.com/software/awdtools/developer/rose/</a>
DOORS Telelogic	Имя собственное инструмента	Инструмент для разработки и управления требованиями. <a href="http://www-01.ibm.com/software/awdtools/doors/">http://www-01.ibm.com/software/awdtools/doors/</a>

(Продолжение)

Sparx Enterprise Architect	Имя собственное инструмента	Инструмент для моделирования ИС. <a href="http://www.sparxsystems.com/products/ea/index.html">http://www.sparxsystems.com/products/ea/index.html</a>
Drug & drop	Бери и тащи	Общеиспользуемое обозначение действия мышью по перемещению элементов, например, на Рабочем столе ОС Windows при удерживаемой левой кнопке мыши
SSL	Secure Sockets Layer – уровень защищенных сокетов	SSL (англ. Secure Sockets Layer – уровень защищенных сокетов) – криптографический протокол, который обеспечивает установление безопасного соединения между клиентом и сервером. <b>SSL изначально разработан компанией Netscape Communications.</b> Впоследствии на основании протокола <b>SSL 3.0 был разработан и принят стандарт RFC</b> , получивший имя TLS.  Протокол обеспечивает конфиденциальность обмена данными между клиентом и сервером, использующими TCP/IP, <b>причем для шифрования</b> применяется асимметричный алгоритм с открытым ключом. При шифровании с открытым ключом используется два ключа, причем любой из них может применяться для шифрования сообщения. Тем самым, если используется один ключ для шифрования, то соответственно для расшифровки нужно задействовать другой ключ. В такой ситуации можно получать защищенные сообщения, публикуя открытый ключ и храня в тайне секретный ключ.  <a href="http://ru.wikipedia.org/wiki/SSL">http://ru.wikipedia.org/wiki/SSL</a>
Supplementary requirements	Дополнительные требования	
Microsoft Visio	Имя собственное инструмента	Продукт Microsoft для построения визуальных схем и диаграмм
Rational SoDA for Word	Имя собственное инструмента	Инструмент для создания отчетов из инструмента управления требованиями Rational Requisite Pro
Use Case Realization	Реализация варианта использования системы/прецедента	Описывает реализацию варианта использования в терминах объектов. Применяется в RUP

API	Application Programming Interface – <b>И</b> нтерфейс прикладного программирования	Интерфейс прикладного программирования (иногда интерфейс программирования приложений) (англ. application programming ginterface, API [эй-пи-ай]) [1] – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. Используется программистами для написания всевозможных приложений
IDEF0	DEF – методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем (Icam <b>DEF</b> inition – IDEF, или другой вариант – Integrated <b>DEF</b> inition)	IDEF0 – <b>Function Modeling</b> – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненности объектов
Диаграммы eEPC	Extended Event Process Chain	Составляющая моделирования ИС
STKR	Stakeholders' Requests	Тип требования. Запросы заинтересованных лиц
BR	Business Requirements	Тип требования. Бизнес-требования
ANSW	Answers	Тип требования. Ответы и собранная информация
BRULE	Бизнес-правила – <b>Busi</b> ness Rules	Непротиворечивый и полный перечень бизнес-правил (положения, определяющие или ограничивающие какие-либо стороны бизнеса с целью защитить структуру бизнеса, контролировать или влиять на его операции)
TERM Therminology	Глоссарий	Термины и понятия системы
ASSUM	Ограничения и допущения – Assumptions	Тип требования. Выявленные системным анализом внешние ограничения на систему со стороны заинтересованных лиц, бизнес-требований и бизнес-правил

(Продолжение)

TECH (Technology Characteristics)	Технологические особенности	Тип требования. Внутренние ограничения системы – результат анализа технологий, платформ разработки и инструментальных средств для разработки продукта
SERT	Требования сертифицирующих организаций – Certification Requirements	Тип требования. Выявленные требования сертифицирующих организаций, которым должна удовлетворять система
STSTD	Стандарты и ГОСТы – State Standarts	Тип требования. Выявленные в ходе первичного анализа и бизнес-анализа требования ГОСТов и других индустриальных стандартов, которым должна удовлетворять система
CHAR	Характеристики аналогичных/наследуемых систем – System Characteristics	Тип требования. Выявленные полезные свойства/характеристики систем, которые должна унаследовать система
BVISION	Концепция создания и развития продукта – Business Vision	Тип требования. Выявленное видение развития/создания продукта с точки зрения бизнеса, ожиданий заинтересованных лиц, требований к стандартизации и полезных/ценных для пользователей характеристик сторонних систем
TVISION	Концепция системы – Technical Vision	Тип требования. Техническая концепция реализации выявленного бизнес-видения, с учетом выявленных ограничений с точки зрения бизнеса и ожиданий заинтересованных лиц и с учетом современных технологий, платформ разработки и инструментальных средств разработки
UREQ	Пользовательские требования – User Requirements	Тип требования. Требования к системе, сформулированные непосредственно ее будущими пользователями
Use Cases (UC)	Варианты использования	Тип требования. Вариант использования системы каким-либо актором – набор последовательных действий, приносящих актору – инициатору выполнения варианта использования – значимый и ощутимый результат
FR	Функциональные требования – Functional Requirements	Тип требования. Тестируемые утверждения, отвечающие на вопрос «Что может делать система?» и описывающие функциональность системы в стиле «Система должна делать то-то и то-то...»

NFR	Нефункциональные требования – Non-functional Requirements	Тип требования. Тестируемые утверждения, часто содержащие количественные показатели, которым должна удовлетворять система, после реализации ее требуемого поведения
GUI	Graphical User Interface	Тип требования. Требования к пользовательскому интерфейсу
ICE	Interface Control Element	Тип требования. Требования к взаимодействию с внешними системами
DOC	Documentation	Тип требования. Требования к документации
DATA	Data	Тип требования. Требования к данным
CERT	Certification	Тип требования. Требования к сертификации
NFR, environment	Специфический тип нефункционального требования	Тип требования. Требования к окружению
NFR, hardware	Специфический тип нефункционального требования	Тип требования. Требования к аппаратно-техническим средствам
NFR, software	Специфический тип нефункционального требования	Тип требования. Требования к программным средствам
NFR, deployment	Специфический тип нефункционального требования	Тип требования. Требования к развертыванию системы
NFR, performance	Специфический тип нефункционального требования	Тип требования. Требования к производительности
NFR, reliability	Специфический тип нефункционального требования	Тип требования. Требования к надежности
NFR, scalability	Специфический тип нефункционального требования	Тип требования. Требования к масштабируемости

(Продолжение)

NFR, security	Специфический тип нефункционального требования	Тип требования. Требования к безопасности
NFR, ergonomics	Специфический тип нефункционального требования	Тип требования. Требования к эргономике
NFR, licensing	Специфический тип нефункционального требования	Тип требования. Требования к лицензированию
NFR, compatibility	Специфический тип нефункционального требования	Тип требования. Требования к совместимости
NFR, support	Специфический тип нефункционального требования	Тип требования. Требования к поддержке
Priority	Приоритет	Атрибут требования. Приоритет в уточнении, реализации и тестировании требования
Complexity	Сложность	Атрибут требования. Сложность реализации требования
Cost	Стоимость	Атрибут требования. Суммарная оценка трудоемкости полной реализации требования в человеко-часах
Assigned To	Назначено	Атрибут требования, участник (участники) проекта, которому назначена работа над требованием
Status	Статус	Атрибут требования. Статус требования
Target Release	Целевой релиз	Атрибут требования. Номер версии продукта, где требование будет реализовано
Phase	Фаза	Атрибут требования. Фаза проекта, в которой реализуется требование
Cause	Причина	Атрибут требования. Текстовое поле для указания запросов на изменение (Enhancement Requests), номеров запросов на разработку функциональности (Feature Request), номеров протоколов совещаний или просто для указания комментариев по разъяснению перевода требования из одного состояния в другое

Enhancement Requests	Запрос на изменение	Запрос на изменение
Feature Request	Запрос на реализацию	Номер запроса на разработку функциональности
Hot	Срочно	Значение атрибута требования
High	Высокий	Значение атрибута требования
Medium	Средний	Значение атрибута требования
Low	Низкий	Значение атрибута требования
Challenge	Вызов	Значение атрибута требования
Solution	Решение	Значение атрибута требования
Normal	Нормальный	Значение атрибута требования
Easy	Легкий	Значение атрибута требования
OMT	Object Modeling Technique	<p>В 1994 году Гради Буч и Джеймс Рамбо, работавшие в компании Rational Software, объединили свои усилия для создания нового языка объектно-ориентированного моделирования. За основу языка были взяты методы моделирования, разработанные ими (Object-Modeling Technique, OMT). OMT был ориентирован на анализ, а Booch — на проектирование программных систем. В октябре 1995 года была выпущена предварительная версия 0.8 унифицированного метода (англ. Unified Method). Осенью 1995 года к компании Rational присоединился Айвар Якобсон, автор метода Object-Oriented Software Engineering — OOSE. OOSE обеспечивал превосходные возможности для спецификации бизнес-процессов и анализа требований при помощи сценариев использования. OOSE был также интегрирован в унифицированный метод.</p> <p><a href="http://ru.wikipedia.org/wiki/Unified_Modeling_Language">http://ru.wikipedia.org/wiki/Unified_Modeling_Language</a></p>
OOSE	Object-Oriented Software Engineering	
THRM		Тип требования. Термин
STRQ		Тип требования. Запросы заинтересованных лиц
CCB	Change Control Board	Комиссия по управлению изменениями

# Словарь терминов

---

---

Computation-independent model	Вычислительно-независимая модель
Platform-independent model	Платформонезависимая модель
Platform-specific model	Платформоориентированная модель
More abstract	Более абстрактная
More specific	Более специфичная
MDA (Model-Driven Architecture)	Моделирование, управляемое архитектурой
VOPC (View Of Participated Classes)	Представление классов, участвующих в реализации поведения
WCF	Windows Communication Foundation
Functionality	Функциональность
Reliability	Надежность
Usability	Юзабилити
Efficiency	Эффективность
Maintainability	Поддерживаемость
Portability	Портируемость
Feasibility	Реализуемость
SDLC (Software Development Life Circle)	Жизненный цикл разработки программного обеспечения
Requirement Management Planning	Планирование управления требованиями (см. ПУТ)
Preliminary Analysis	Предварительный анализ

Business Analysis	Анализ текущего состояния и описание бизнеса объекта автоматизации
Conception Design	Концепция автоматизации
Enhancement Requests	Запрос на расширение функциональности
Business Use Case Model	Модель бизнес-кейсов
Use Case Modeling (UCM)	Описание желаемого поведение системы
Use Case Analysis (UCM)	Анализ ожиданий и поведения системы
System Analysis	Системный анализ
Functional Design	Иерархия функциональных требований
Legacy SRS Analysis and Requirements Reengineering	Восстановление требований из унаследованных спецификаций требований
Legacy Requirements Reverse Engineering	Восстановление наследованных требований
Additional Business Analysis	Дополнительный бизнес-анализ
Current Application Reverse Engineering	Восстановление требований из текущей реализации системы
PA – Preliminary Analysis	Предварительный анализ
BA – Business Analysis	Анализ текущего состояния и описание бизнеса объекта автоматизации
FD – Functional Design	Функциональный дизайн
SD – System Design	Системный дизайн
Unit_T – Unit Testing	Юнит-тестирование
FT – Functional Testing	Функциональное тестирование
Integr_T – Integrational Testing	Интеграционное тестирование
SOW – Statement of Work	Заявление о нужных работах

*(Продолжение)*

STKR – Stakeholder	Запросы заинтересованных лиц
PMP – Project Management Plan	План управления проектом (см. ПУП)
CD – Conceptual Design	Дизайн концепции
AD – Architectural Design	Архитектурный дизайн
DEV – Development	Разработка
Stress_T – Stress Testing	Нагрузочное (стрессовое) тестирование
CAT – Customer Acceptance Test	Приемочные испытания
TM – Test Management	Управление тестированием
Risk Mng – Risk Management	Управление рисками
QAtr – Quality Attributes	Атрибуты качества
Conception Creation	Создание концепции
Architecture Design	Проектирование архитектуры
Functionality Realization	Реализация функциональности
Production	Производство
Features	Функциональность
Spiral Architecture Driven Development	Спиральная архитектурно-ориентированная разработка ПО
Initiation	Инициация
Happy Path Definition	Определение удачных сценариев выполнения прецедентов
Draft	Версия
Release	Релиз
TD (Test Design)	Проектирование и планирование тестирования
SA	Системный анализ
Conceptual Model	Концептуальная модель

Product Group Manager	Менеджер группы продуктов (см. Менеджер продуктов)
Lead Architect	Главный архитектор
BSC (Balanced Score Card)	Система сбалансированных показателей
KGI (Key Goals Indicators)	Ключевые показатели достижения целей
Client Service	Клиент-сервисная технология
EBITDA	Сумма операционной прибыли
CSF (Critical Success Factors)	Критические факторы успеха выполнения процессов
PMO (Project Management Office)	Офис управления проектами
Lessons Learned	Выученные уроки
Next Release SOW	Задание на следующий релиз
Deployment Requirements	Требования к развертыванию
SOW feasibility analysis and estimations effort	Анализ реализуемости задания на следующий релиз и оценка трудозатрат
Roles and Responsibilities Analysis	Анализ ролей и ответственностей
Business Processes	Бизнес-процессы
Functional Roles	Функциональные роли
Decisions	Решения
Functions	Функции
Activities	Деятельности
Responsible	Ответственный
Accountable	Орган/лицо отчетности
Inform	Орган/лицо для информирования
Consulting	Консультант – владеет информацией/ожиданиями/возможностями, необходимыми для выполнения и завершения работы

*(Продолжение)*

KPI (Key Process Indicators)	Ключевые показатели хода выполнения процессов
SG (Specific Goal)	Специфическая цель
GG (General Goal)	Общая цель
GP (General Practice)	Общая практика
SP (Specific Practice)	Специфическая практика
Steering Committee	Исполнительный комитет
WAS ( <b>Work Authorization System</b> )	Система авторизации работ
PMS ( <b>Project Management System</b> )	Система управления проектов
ALM (Application Life Time System)	Система управления жизненным циклом производства продукта
Engineering Process Areas	Области процесса разработки
TS (Technical Solutions)	Область технических решений
PI (Project Interation)	Интеграционные процессы проекта
RD (Requirement Development)	Разработка требований
REQM (Requirement Management)	Управление требованиями
DAR (Decision Analysis and Resolutions)	Методика анализа и принятия решений
VER (Verification)	Верификация
RSKM (Risk Management)	Управление рисками
PP (Project Planning)	Планирование проекта
PMC (Project Monitoring and Control)	Мониторинг и контроль проекта
VAL (Validation)	Приемка продукта
Project Management Process Areas	Области процесса управления проектом

Basic Project Management Process Area	Взаимодействие процессных областей
Advanced Project Management Process Areas	Области расширенного процесса управления проектом
Integrated Product Management (IPM)	Интегрированное управление продуктом
PMBOK (Project Management Body of Knowledge)	Свод знаний по управлению проектом
Peer Review	Одноранговое ревью
Steering Committee	Управляющий комитет
SEPG	Группа совершенствования процессов инженерии ПО
Institution	Институция – формальное описание процессов в виде процедур, рабочих инструкций и регламентов
Workflow	Поток работ
Customer Relationship Management System	Система управления взаимоотношениями с клиентами
Enterprise Management System	Система управления предприятием
Capability	Способность, возможность
Backbone	Каркас (кость скелета)
Story	История
System	Система
Object Constraint Management	Управление ограничениями на объекты и связи объектов
Employee	Работник
Work For	Работает на
Company	Компания
Technology	Технология

# Послесловие

---

---



В последнее время стало очевидным, что важнейшей составляющей проектов по автоматизации деятельности предприятий является набор консалтинговых услуг — именно они обладают особой ценностью для клиента. Потребителю данных услуг важно обозначить свои «неразрешимые» проблемы и получить обоснованные ответы на такие вопросы: что и как ему надлежит сделать и каких эффектов он может добиться благодаря консалтинговой поддержке? От практической реализации полученной модели он, как правило, ожидает, «чтобы все работало так, как мы договорились».

На самом деле лучшим специалистом по консалтингу является... сам клиент, особенно тот, который неоднократно проходил от и до весь путь оптимизации работы предприятия. Задача консультанта, разумеется, — в выявлении бизнес-требований клиента и в развитии бизнес-процессов, но не только. Хороший специалист должен помочь клиенту взглянуть на себя «со стороны» и самому найти новые неожиданные решения.

Выстроив структуру конкретного проекта, консультанту необходимо уметь языком функциональных требований и спецификаций сформулировать задачу команде разработчиков программного обеспечения. Для этого требуются специальные знания и навыки, которые в программной инженерии обозначаются понятиями бизнес-анализ и системный анализ.

Компания «Кейсистемс» рекомендует книгу «Путь аналитика» как практическое руководство для профессионального развития бизнес-аналитиков в ИТ-сфере. Данное издание незаменимо именно для тех специалистов, которые работают на грани между бизнес-задачами клиента и программной инженерией компании-разработчика, или системного интегратора.

Книга «Путь аналитика» предназначена как для молодых специалистов, так и для опытных консультантов, желающих развивать свои навыки в области управления требованиями и программной инженерии.

*Генеральный директор ООО «Кейсистемс»  
Алексей Матросов*