Разработка требований к программному обеспечению

Содержание

Предисловие

Несмотря на более чем пятидесятилетнее существование компьютерной отрасли, многие компании разработчики программного обеспечения по-прежнему прикладывают значительные усилия для сбора и документирования требований к ПО, а также управления ими. Недостаточный объем информации, поступающей от пользователей, требования, сформулированные не полностью, их кардинальное изменение — вот основные причины, из-за которых зачастую командам, работающим в области информационных технологий, не удается вовремя и в рамках бюджета предоставить клиентам всю запланированную функциональность[1]. Многие разработчики не умеют спокойно и профессионально собирать требования пользователей к ПО. У клиентов зачастую. не хватает терпения участвовать в разработке требований к ПО, или они норовят передать свои пожелания через совершенно неподходящих для этого дела людей. Иногда участники проекта даже не могут прийти к единому мнению, что же такое «требование». Как заметил один писатель, «программисты скорее предпочтут расшифровать слова классической песни Кингсмена (Kingsmen) «Louie Louie», чем требования клиентов».[2]

Разработка ПО включает по крайней мере столько же общения, сколько и обычная работа с компьютером, но зачастую мы делаем акцент на работе с компьютером и не уделяем достаточно внимание общению. В этой книге описаны дюжины способов, позволяющих реализовать это общение и помогающих разработчикам ПО, менеджерам, маркетологам и клиентам использовать на практике эффективные методы формулирования требований к ПО. Настоящее издание дополнено новыми главами, посвященными роли аналитика требований, важности бизнес-правил, а также рассказом о том, как формулирование требований важно для проектов по обслуживанию, для комплексных решений, для извлечения данных из внешних источников и новых проектов. Во многочисленных врезках я привожу реальные примерь, иллюстрирующие типичные ситуации, возникающие при формулировании требований.

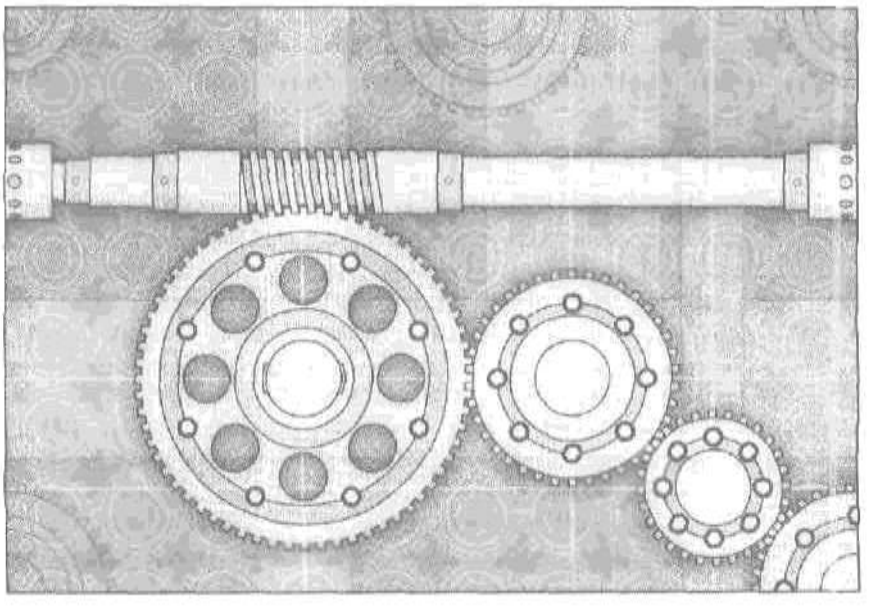
В книге описаны основные «отличные способы» формулирования требований, а не экзотичные или тщательно разработанные методологии, обещающие решить все проблемы, возникающие при формулировании требований.

---

[1] Исследование «CHAOS Report», проведенное компанией Standish Group International, 1995.

[2] Peterson, Gary. Статья «Risque Requirements» в апрельском номере журнала CrossTalk за 2002.

Часть I. Требования к продукту: что, почему и кто



Глава 1. Особенности разработки требований к ПО

«Привет, Фил, это Мария из отдела персонала. У нас проблема с системой, которую вы разрабатывали для нас. Одна из сотрудниц изменила фамилию на Спакл Старлайт, а система не принимает это изменение. Ты можешь помочь?»
Она вышла замуж за парня по имени Старлайт?»
"Нет, она просто взяла другую фамилию, — ответила Мария. — В этом-то и проблема. Как будто мы можем брать другую фамилию только при изменении семейного положения».
«Да, я не предусмотрел такой возможности. Я не помню, чтобы, и ты говорила мне о ней, когда мы обсуждали систему. Вот поэтому-то диалоговое окно «Изменить фамилию» доступно только из окна «Изменение семейного положения», - сказал Фил.
«Я полагаю, ты знаешь, что люди могут законным образом менять фамилию, когда захотят, — ответила Мария. — Мы должны исправить это к пятнице, а то Спакл не сможет обналичить чек. Ты сможешь исправить этот дефект к такому сроку?»
«Это не дефект, — резко парировал Фил. — Я и не знал, что вам нужна эта функциональность. Сейчас я занимаюсь оценкой производительности системы. Думал, что потребуется менять что-то другое». (Шелест бумаги.) «О-о, да здесь есть еще кое-что. Я, вероятно, смогу исправить это в конце месяца, но не на этой неделе. Извини меня. Но в следующий раз с такими просьбами звони пораньше и, пожалуйста, описывай их». «А что же я скажу Спакл? — возмутилась Мария. — Ей придется залезать в долги, если она не сможет обналичить чек». «Эй, Мария, это не моя вина, — запротестовал Фил. — Если ты не сказала мне при первом осуждении, что вам понадобится изменять фамилию в любое время, этого и нет в системе. Ты не можешь винить меня за то, что я не умею читать мысли в твоей голове».
Сердитая и смирившаяся, Мария отрывисто сказала: «Это как раз то, из-за чего я ненавижу компьютерные системы. Позвони мне, как только сможешь исправить недостаток».

Если вы когда-либо бывали в шкуре клиента на переговорах, подобных этому, то знаете, как расстраиваются пользователи продукта[3], который не позволяет выполнять основные задачи. Разработчики испытывают чувство разочарования, когда узнают о функциональности, которую пользователи ожидали получить, только после развертывания системы. Точно также раздражает, если приходится прерывать работу из-за необходимости модифицировать систему, так как она не выполняет те задачи, о которых вы говорили еще в самом начале разработки.

Масса проблем с ПО возникает из-за несовершенства способов, которые люди применяют для сбора, документирования, согласования и модификации требований к ПО. Как в случае с Филом и Марией, проблемы могут возникать из-за неформального сбора информации, предполагаемой функциональности, ошибочных или несогласованных предположений, недостаточно определенных требований и бессистемного изменения процесса.

Большинство людей при строительстве дома даже не интересуются подрядчиками, пока в полном объеме не обсудят свои потребности и желания и не уточнят все детали. Покупатели понимают, что внесение изменений влечет за собой изменение цены; им это не нравится, но они это понимают. Однако люди весело ищут оправдания, когда дело касается разработки ПО. Ошибки, допущенные на стадии сбора требований, составляют от 40 до 60% всех дефектов проекта (Davis, 1993; Leffingwell, 1997). Две наиболее распространенные проблемы, о которых сообщается в большом Европейском обзоре индустрии ПО, — определение и управление требованиями заказчика (ESPITI, 1995). Тим не мене многие организации еще применяют неэффективные методы на этих стадиях разработки ПО. Типичный исход — неожидаемые

[3] Я использую термины *продукт*, *система* и *приложение* как взаимозаменяемые в этой книге. Концепции и приемы, обсуждаемые здесь, применимы к ПО любого вида и к ПО, содержащему элементы, которые вы строите.

пробелы, а также различия между тем, что разработчики собираются строить, и тем, в чем клиенты реально нуждаются.

Нигде более, как на стадии сбора требований, так тесно не связаны интересы всех заинтересованных в проекте лиц с успехом проекта. К заинтересованным в проекте лицам относятся:

- **заказчики**, которые финансируют проект или приобретают продукт для решения каких-то бизнес-задач;
- **пользователи**, которые взаимодействуют напрямую или не напрямую с приложением (подкласс заказчиков);
- **аналитики требований**, которые пишут требования и передают их разработчикам;
- **разработчики**, которые создают, разворачивают и поддерживают продукт;
- **тестеры**, которые определяют соответствие поведения ПО желаемому;
- **технические писатели**, которые отвечают за создание руководства пользователей, тренировочные материалы и справочную систему;
- **менеджер по проекту**, который планирует процесс и руководит командой разработчиков вплоть до успешного выпуска продукта;
- **сотрудники правового отдела,** которые следят, чтобы продукт не противоречил всем действующим законам и постановлениям;
- **производственники**, которые должны построить продукт, содержащий данное ПО;
- **сотрудники отдела продаж и маркетинга**, выездной службы поддержки и другие, кому придется работать с продуктом и его пользователями.

Хорошо справившись с этой стадией процесса, вы получите отличный продукт, восхищенных заказчиков и удовлетворенных разработчиков. В противном случае вам грозит непонимание, разочарование и разногласия, которые подрывают веру в продукт и его ценность. Так как требования представляют собой основу для действий и разработчиков и менеджеров проекта, все заинтересованные в проекте лица должны быть вовлечены в создание этого документа.

Но разработка требований и управление ими — трудный процесс! Здесь нет быстрых или волшебных решений. Однако многие организации борются с одними и теми же трудностями, для преодоления которых мы можем найти общие приемы, годные в различных ситуациях. В этой книге описаны дюжины таких приемов. Их рекламируют для построения абсолютно новых систем, однако они отлично подходят для поддержки проектов и выбора коммерческих готовых решений (см. главу 16). Не используйте эти приемы только для модели «водопада». Тем командам, которые выбирают итеративный процесс, необходимо знать, что же делать на каждом этапе.

Из этой главы вы узнаете:

- несколько ключевых терминов, применяемых при конструировании ПО;
- особенности разработки требований, которые отличают этот процесс от управления требованиями;
- тревожные симптомы некоторых связанных с требованиями проблем, которые могут иногда возникать;
- некоторые характеристики великолепных требований.

Держите руку на пульсе

Чтобы оперативно проверить приемы построения требований в вашей организации, опростите себя, как много следующих положений верны для ваших текущих проектов. Если вы пометите хотя бы три или четыре положения, эта книга для вас.

- Образ и границы проекта никогда не определены ясно.
- Заказчики очень заняты, чтобы работать с аналитиками и программистами над требованиями.
- Заместители пользователей — менеджеры по продуктам, по разработке, менеджеры пользователей или маркетологи — вызываются говорить от имени клиентов, но не точно представляют их потребности.
- Требования существуют только в головах «экспертов», работающих в вашей организации, и никогда не фиксируются в письменном виде.
- Заказчики настаивают, чтобы все требования были критическими, без учета их приоритетов.
- Разработчики получают двусмысленную и неполную информацию, поэтому при кодировании им приходится делать предположения.
- Взаимодействие между разработчиками и заказчиками ограничивается внешним видом пользовательского интерфейса и не затрагивает того, что же действительно клиенты собираются делать с помощью приложения.
- Ваши заказчики подписывают требования и затем постоянно изменяют их.
- Проект разрастается, когда вы принимаете изменения требований, график при этом нарушается, потому что никаких дополнительных ресурсов не выделяется и никакие функции не удаляются.
- Запросы на изменение требований теряются по пути: ни вы, ни ваши клиенты не знают статус каждого запроса.
- Заказчики настаивают на определенной функциональности, которую разработчики и создают, однако эти возможности системы клиентам не нужны.
- Технические требования хороши, а вот заказчики нет.

Оговоренные требования к ПО

Одна из проблем, существующих в индустрии программного обеспечения, — это отсутствие общепринятых определений терминов, которыми мы пользуемся для описания нашей работы. Разные эксперты, говоря об одном и том же документе, называют его и *требования пользователя*, и *требования к ПО*, и *функциональные требования*, и *системные требования*, и *технологические требования*, и *бизнес-требования*, и *требования к продукту*. Заказчики зачастую считают, что требования — это развитая концепция продукта, предназначенная для разработчиков. Те, в свою очередь, полагают, что в отношении клиентов это детальная разработка интерфейса пользователя. Такое многообразие ведет к сумятице и раздражающим проблемам во взаимодействии сторон.

Основной закон: **требования должны быть документированы**. Однажды я занимался проектом, который делала опытнейшая команда, однако ее состав постоянно менялся. Заказчик впал в неистовство, когда, к нему пришел еще один аналитик требований и сказал: «Мы должны поговорить о том, что же вам нужно». На что заказчик ответил: «Я уже излагал свои требования. А

теперь постройте мне систему!» Дело в том, что никто не документировал собранную информацию, поэтому каждому новому аналитику приходилось начинать с набросков. Полный бред объявлять о готовности требований, если на самом деле у вас есть куча электронных писем, голосовых сообщений, записок, протоколов совещаний и неясных воспоминаний о беседах в коридоре.

Особенности интерпретации требований

Консультант Brian Lawrence предположил, что требования - это «нечто такое, что приводит к выбору проектрования» (Lawrence, 1997). Многие категории информации попадают в эту категорию. IEEE Standard Glossary of Software Engineering Terminology (1990) определяет требования как:

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

Это определение охватывает требования как пользователей (внешнее поведение системы), так и разработчиков (некоторые скрытые параметры). Термин пользователи следует распространить на всех заинтересованных лиц, так как не все, кто заинтересован в проекте — пользователи. Я думаю о требованиях, как о свойствах, которыми должен обладать продукт, чтобы представлять какую-то ценность для пользователей. Следующее определение подтверждает различие типов требований (Sommerville и Sawyer, 1997):

> *Требования ... это спецификация того, что должно быть реализовано. В них описано поведение системы, свойства системы или ее атрибуты. Они могут быть ограничены процессом разработки системы.*

Ясно, что нет универсального определения требований. Для облегчения восприятия материала необходимо согласовать набор характеристик, чтобы модифицировать трудный термин требования (requirements), а также оценить важность их записи в общедоступной форме.

Ловушка

Не полагайте, что все лица, заинтересованные в вашем проекте, разделяют общее представление о том, что же такое требования. Прежде всего определитесь с пониманием термина «требование», чтобы вы все изначально говорили на одном языке.

Уровни требований

В этом разделе приводятся определения, которыми я буду пользоваться для терминов, наиболее часто применяемых у такой сфере, как разработка требований. Требования к ПО состоят из трех уровней — *бизнес-требования*, *требования пользователей* и *функциональные требования*. Вдобавок каждая система имеет свои *нефункциональные требования*. Модель на рис. 1-1 иллюстрирует способ представления этих типов требований. Как и все модели, она не полная, но схематично показывает организацию требований. Овалы обозначают типы информации для требований, а прямоугольники — способ хранения информации (документы, диаграммы, базы данных).

Дополнительная информация

В главе 7 приводится множество примеров различных типов требований.

Бизнес-требования (business requirements) содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга или «провидец» продукта. В этом документе объясняется, почему организации нужна такая система, то есть описаны цели, которые организация намерена достичь с ее помощью. Мне нравится записывать бизнес-требования в форме ***документа об образе и границах проекта***, который еще иногда называют ***уставом проекта*** (project charter) или ***документом рыночных требований*** (market requirements document). Создание такого документа описано в главе 5. Определение границ проекта представляет собой первый этап управление общими проблемами расползания границ.

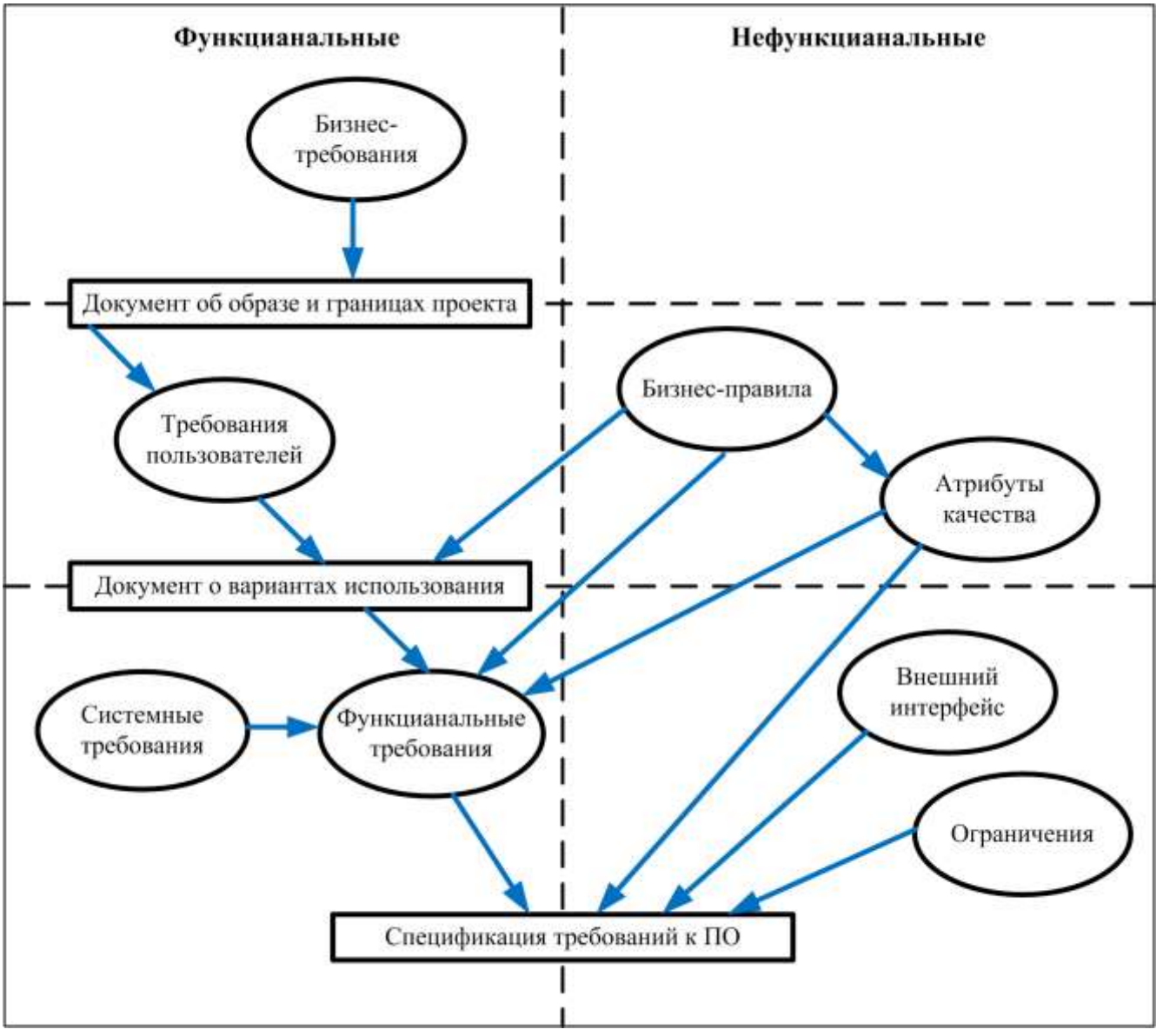


Рис. 1-1. Взаимосвязи нескольких типов информации для требований

Требования пользователей (user requirements) описывают цели и задачи, которые пользователям позволит решить система. К отличным способам представления этого вида

требований относятся ***варианты использования***, ***сценарии*** и ***таблицы «событие - отклик»***. Таким образом, в этом документе указано, что клиенты смогут делать с помощью системы. Пример варианта использования — «Сделать заказ» для заказа билетов на самолет, аренды автомобиля, заказа гостиницы через Интернет.

Дополнительная информация

Глава 8 посвящена требованиям пользователей.

Функциональные требования (functional requirements) определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Иногда именуемые ***требованиями поведения*** (behavioral requirements), они содержат положения с традиционным «должен» или «должна»: «Система должна по электронной почте отправлять пользователю подтверждение о заказе». Как иллюстрируется в главе 10, функциональные требования описывают, что разработчику необходимо реализовать.

Термином ***системные требования*** (system requirements) обозначают высокоуровневые требования к продукту, которые содержат многие подсистемы, то есть система (IEEE, 1998c). Говоря о системе, мы подразумеваем программное обеспечение или подсистемы ПО и оборудования. Люди — часть системы, поэтому определенные функции системы могут распространиться и на людей.

Однажды моя команда писала ПО для контроля некоторой лабораторной аппаратуры, которая автоматизировала скучное добавление точного количества вещества в ряд мензурок. Из требований для всей системы мы вывели производные функциональные требования к ПО, чтобы посылать сигналы оборудованию для перемещения распределяющих вещество насадок, считывания показаний датчиков положения и включения/выключения насоса.

Бизнес-правила (business rules) включают корпоративные политики, правительственные постановления, промышленные стандарты и вычислительные алгоритмы. Как вы увидите в главе 9, бизнес-правила не являются требованиями к ПО, потому что они находятся вне границ любой системы ПО. Однако они часто налагают ограничения, определяя, кто может выполнять конкретные варианты использования, или диктовать, какими функциями должна обладать система, (подчиняющаяся соответствующим правилам. Иногда бизнес-правила становятся источником атрибутов качества, которые реализуются в функциональности. Следовательно, вы можете отследить происхождение конкретных функциональных требований вплоть до соответствующих им бизнес-правил.

Функциональные требования документируются в ***спецификации требований к ПО*** (software requirements specification, SRS), где описывается так полно, как необходимо, ожидаемое поведение системы. Я буду относиться к спецификации, как к документу, хотя это может быть база данных или крупноформатная таблица с требованиями, хранилище данных в коммерческом инструменте управления требованиями (см. главу 21) или даже, может быть, куча карточек для небольшого проекта. Спецификация требований к ПО используется при разработке, тестировании, гарантии качества продукта, управлении проектом и связанных с проектом функциях..

В дополнение к функциональным требованиям спецификация содержит нефункциональные, где описаны цели и атрибуты качества. ***Атрибуты качества*** (quality attributes) представляют собой дополнительное описание функций продукта, выраженное через описание его характеристик, важных для пользователей или разработчиков. К таким характеристикам относятся легкость и простота использования, легкость перемещения, целостность, эффективность и устойчивость к сбоям. Другие нефункциональные требования описывают внешние взаимодействия между системой и внешним миром, а также ограничения проектирования и реализации. ***Ограничения*** (constraints) касаются выбора возможности разработки внешнего вида и структуры продукта.

Люди часто рассуждают о характеристиках продукта. ***Характеристика*** (feature) — это набор логически связанных функциональных требований, которые обеспечивают возможности пользователя и удовлетворяют бизнес-цели. В области коммерческого ПО характеристика представляет собой узнаваемую всеми заинтересованными лицами группу требований, которые важны при принятии решения о покупке — элемент маркированного списка в описании продукта. Характеристики продукта, которые перечисляет клиент, не эквивалентны тем, что входят в список необходимых для решения задач пользователей. В качестве примеров характеристик продуктов можно привести избранные страницы или закладки Web-браузера, контроль за орфографией, запись макрокоманды, сервопривод стекла в автомобиле, on-line- обновление или изменение налогового кодекса, ускоренный набор телефонного номера или автоматическое определение вируса. Характеристики могут охватывать множество вариантов использования, и для каждого варианта необходимо, чтобы множество функциональных требований было реализовано для выполнения пользователем его задач.

Чтобы вы лучше восприняли некоторые из различных видов требований, я рассмотрю программу подготовки текстов. Бизнес-требование может выглядеть так: «Продукт позволит пользователям исправлять орфографические ошибки в тексте эффективно». На коробке CD-ROM указано, что проверка грамматики включена как характеристика, удовлетворяющая бизнес-требование. Соответствующие требования пользователей могут содержать задачи (варианты использования) вроде такой: «Найдите орфографическую ошибку» или «Добавьте слово в общий словарь». Проверка грамматики имеет множество индивидуальных функциональных требований, которые связаны с такими операциями, как поиск и выделение слова с ошибкой, отображение диалогового окна с фрагментом текста, где это слово находится, и замена слова с ошибкой корректным вариантом по всему тексту. Атрибут качества легкость и ***простота использования*** (usability) как раз и определяет его значение посредством слова «эффективно» в бизнес-требованиях.

Менеджеры и сотрудники отдела маркетинга определяют бизнес-требования для ПО, которые помогут их компании работать эффективнее (для информационных систем) или успешно конкурировать на рынке (для коммерческих продуктов). Каждое требование пользователя должно быть сопоставлено бизнес-требованию. На основе требований пользователя аналитики определяют функции, которые позволят пользователям выполнять их задачи. Разработчикам необходимы функциональные и нефункциональные требования, чтобы создавать решения с желаемой функциональностью, определенным качеством и требуемыми рабочими характеристиками, не выходя за рамки налагаемых ограничений.

Хотя в модели на рис. 1-1 показан поток требований в направлении сверху вниз, следует ожидать и циклов, и итераций между бизнес-требованиями, требованиями пользователей и функциональными требованиями. Кто бы, когда бы ни предложил новые характеристики, варианты использования или функциональные требования, аналитик должен спросить; «Они попадают в указанные границы?» Если ответ «да», то они соответствуют спецификации. Если — «нет», то на «нет», как известно, и суда нет. А вот если ответ «нет, но они должны там быть», то заказчик или тот, кто финансирует проект, должен решить, готов ли он раздвинуть границы проекта, чтобы принять новое требование.

Каких требований не должно быть

Спецификация требований не содержит деталей проектирования или реализации (кроме известных ограничений), данных о планировании проекта или сведений о тестировании (Letfingwell и Widrig, 2000). Удалите указанные элементы из требований, чтобы из этого документа было абсолютно ясно, что надлежит построить команде разработчиков. Для проекта, как правило, создаются требования других типов: документ, где описана среда разработки, ограничения бюджета, руководство пользователя или требования для выпуска продукта и продвижения его в поддерживаемую среду. Все это относится к требованиям к проекту, но не к продукту, поэтому они не будут рассмотрены в этой книге.

Разработка и управление требованиями

Путаница, которая возникает, как только речь заходит о требованиях затрагивает даже то, что именно называть этим словом. Некоторые авторы называют целую область *разработкой технических условий*.(requirements engineering) (Sommerville и Kotonya, 1998); другие употребляют термин *управление требованиями* (requirements management) (Leffingwell и Widrig, 2000). Я считаю полезным разделить область разработки технических условий на ***разработку требований*** (requirements development) — подробнее об этом в части II этой книги — и ***управление требованиями*** (requirements management) — см. часть III, как показано на рис. 1-2.

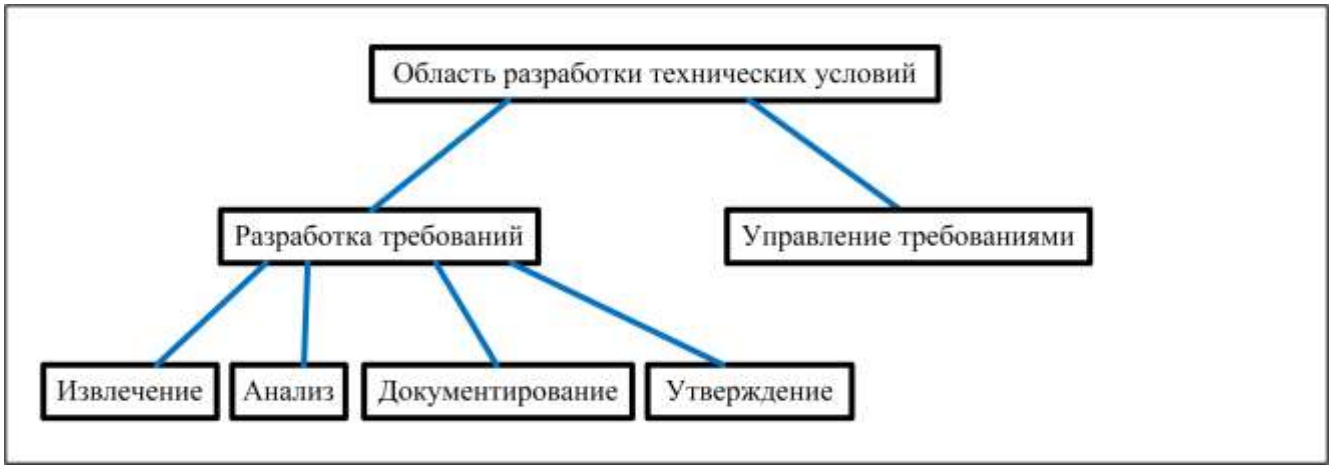


Рис. 1-2. Компоненты области разработки технических условий

Разработка требований

Далее мы можем подразделить этот этап на ***извлечение*** (elicitation), ***анализ*** (analysis), ***документирование*** (specification) и ***утверждение*** (validation) (Abran и Moore, 2001). В эти подэтапы входят все действия, включающие сбор, оценку и документирование требований для ПО или ПО-содержащих продуктов, в том числе:

- идентификация классов пользователей для данного продукта;
- выяснение потребностей тех, кто представляет каждый класс пользователей;
- определение задач и целей пользователей, а также бизнес-целей, с которыми эти задачи связаны;
- анализ информации, полученной от пользователей, чтобы отделить задачи от функциональных и нефункциональных требований, бизнес правил, предполагаемых решений и поступающих извне данных;
- распределение высокоуровневых требований по компонентам: к определенным в системной архитектуре;
- установление относительной важности атрибутов качества;
- установление приоритетов реализации;
- документирование собранной информации и построение моделей;
- просмотр спецификации требований, который позволяет удостовериться в том, что запросы пользователей всеми понимаются одинаково, и устранение возникших проблем до передачи документа разработчикам.

Постепенность процесса — ключ к успеху разработки требований. Планируйте цикличность исследования требований, детализируйте высокоуровневые требования и уточняйте их корректность у пользователей. Выполнение этих задач требует времени и может разочаровать, однако это важный аспект работы с неясно определенным новым ПО.

Управление требованиями

Этот этап определяется как «выработка и поддержание взаимного согласия с заказчиками по

поводу требований к разрабатываемому ПО» (Paulki др., 1995). Это соглашение воплощается в спецификации (в письменной форме) и моделях. Одобрение пользователей — только половина дела. Разработчики также должны принять задокументированные требования и высказаться за создание этого продукта. К действиям по управлению требованиями относятся:

- определение основной версии требований (моментальный срез требований для конкретной версии продукта);
- просмотр предлагаемых изменений требований и оценка вероятности воздействия каждого изменения до его принятия;
- включение одобренных изменений требований в проект установленным способом;
- согласование плана проекта с требованиями;
- обсуждение новых обязательств, основанных на оцененном влиянии изменения требований;
- отслеживание отдельных требований до их дизайна, исходного кода и вариантов тестирования;
- отслеживание статуса требований и действий по изменению на протяжении всего проекта.

На рис. 1-3 показан другой способ разделения областей разработки требований и управления ими. В этой книге описано несколько дюжин специальных приемов для выполнения извлечения требований, их анализа, документации, проверки и управления,

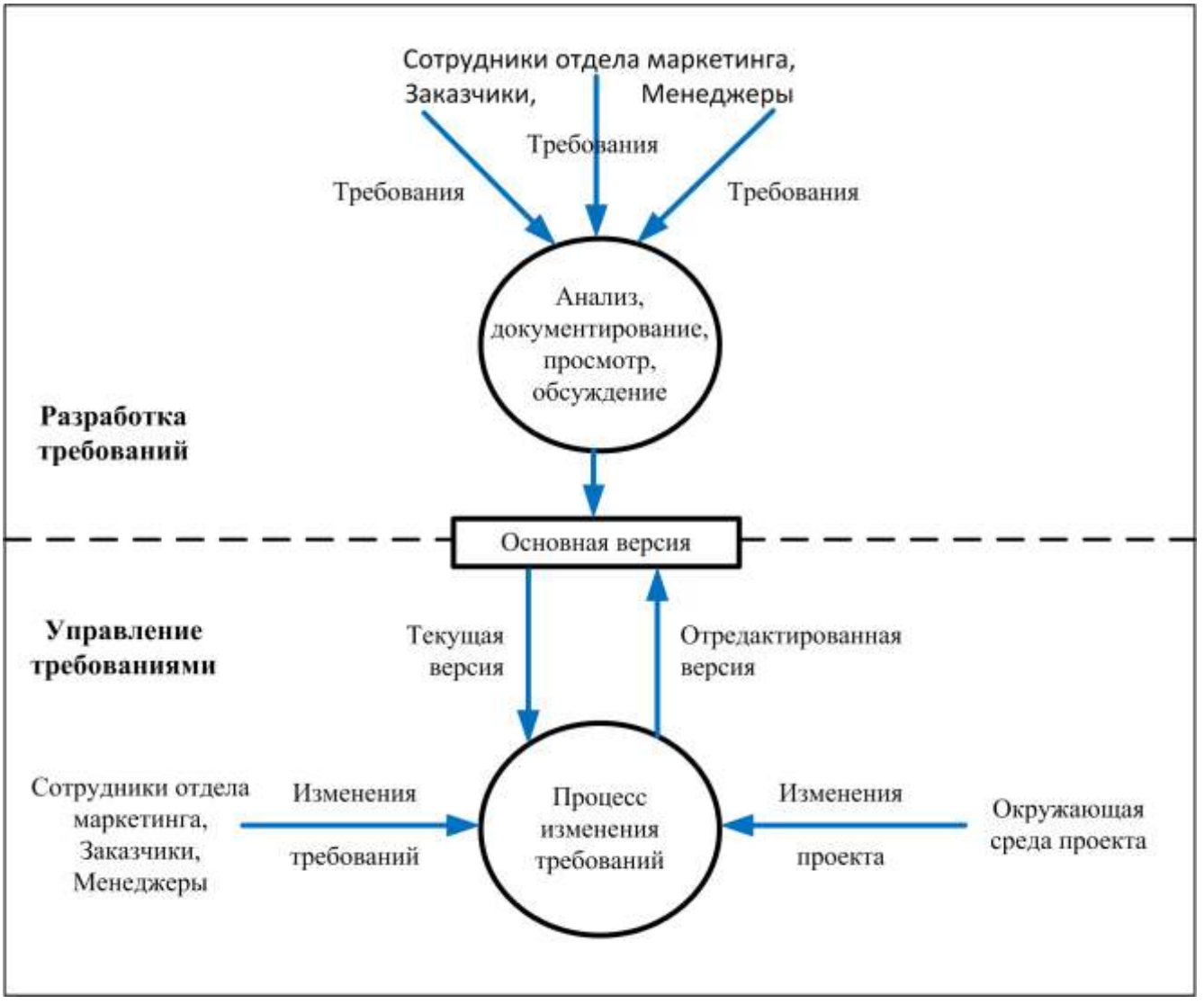


Рис. 1-3. Разделение областей разработки требований и управления ими

Каждый проект имеет требования

Frederick Brooks выразительно определил критическую роль требований при разработке ПО в классическом эссе (1987) «No Silver Bullet: Essence and Accidents of Software Engineering»:

> *Строжайшее и единственное правило построения систем ПО — решить точно, что же строить. Никакая другая часть концептуальной работы не является такой трудной, как выяснение деталей технических требований, в том числе и взаимодействие с людьми, с механизмами и с иными системами ПО. Никакая другая часть работы так не портит результат, если она выполнена плохо. Ошибки никакого другого этапа работы не исправляются так трудно.*

Каждое приложение или содержащая ПО система имеет пользователей, которые полагаются на нее, как на средство улучшения их жизни. Время, которое тратится на выяснение потребностей пользователей, представляет собой высокоэффективную инвестицию в успех проекта. Если разработчики не записывают требования, с которыми заказчики уже согласились, как могут они удовлетворить этих заказчиков? Даже требования к ПО, не предназначенного для коммерческого использования, следует хорошо понять. Например, библиотеки ПО, компоненты и инструменты, созданные для внутреннего применение разработчиками.

Зачастую невозможно полностью определить требования до предварительного конструирования. В этом случае действуйте итеративно и постепенно разрабатывайте одну порцию требований за раз, обязательно дождитесь ответной реакции заказчика, прежде чем приступить к следующему циклу. У вас не будет оправдания, если вы начнете писать код до того, как просмотрите требования перед следующим шагом. Итерации при кодировании стоят гораздо дороже, чем при разработке концепций.

Люди иногда впустую тратят время, отведенное на написание требований, но этот этап не самый сложный. Самое трудное — выявить эти требования. Первоначальное написание требований представляет собой процесс выяснения, разработки и расшифровки данных. Четкое понимание требований к продукту дает вам уверенность, что ваша команда работает над теми проблемами, над которыми нужно, и придумает лучшее их решение. Требования позволят вам расставить приоритеты в работе и оценить ресурсы, которые понадобятся. Не зная, что собой представляют требования, вы не сможете определить момент окончания проекта, установить, достигнуты ли цели, или выбрать компромиссное решение, когда придется сужать границы проекта.

Никаких предположений в требованиях

Я недавно имел дело с группой разработчиков, которая применяла доморощенные инструменты разработки ПО, в том числе и редактор исходного кода. К сожалению, ни один из них не записал, что инструмент должен позволять пользователям печатать код, хотя пользователи без сомнения предполагали, что это будет именно так. Разработчикам приходилось вручную переписывать исходник, чтоб просмотреть код.

Если вы не записываете даже подразумеваемые и предполагаемые требования, не удивляйтесь, если продукт не будет отвечать ожиданиям пользователей. Периодически спрашивайте: «Что мы предполагаем?», чтобы постараться извлечь на поверхность эти спрятанные мысли. Если вы заметите какое-то предположение при обсуждении требований, запишите его и подтвердите его точность. Если вы занимаетесь заменой системы, просмотрите системные характеристики, чтобы выяснить, действительно ли они необходимы при замещении, а не выносите поспешного решения — да или нет.

Когда плохие требования появляются у хороших людей

Основное следствие проблем с требованиями — переделка того, что, как вы думаете, уже готово. На это расходуется от 30 до 50% общего бюджета разработки (Boehm и Papaccio, 1988), а ошибки в требованиях стоят от 70 до 85% стоимости этой переделки (Leffingwell, 1997). Как показано на рис. 1-4, гораздо дороже исправить дефекты, которые найдены позднее в проекте, чем сразу после создания (Grady, 1999). Следовательно, предотвращение ошибок в требованиях и обнаружение их на ранних стадиях сильно уменьшает объем переделки. Вообразите, насколько изменится ваша жизнь, если вы сократите этот объем наполовину! Вы сможете построить ПО быстрее или сконструировать больший и лучший продукт за тоже время и даже иногда будете приходить домой.

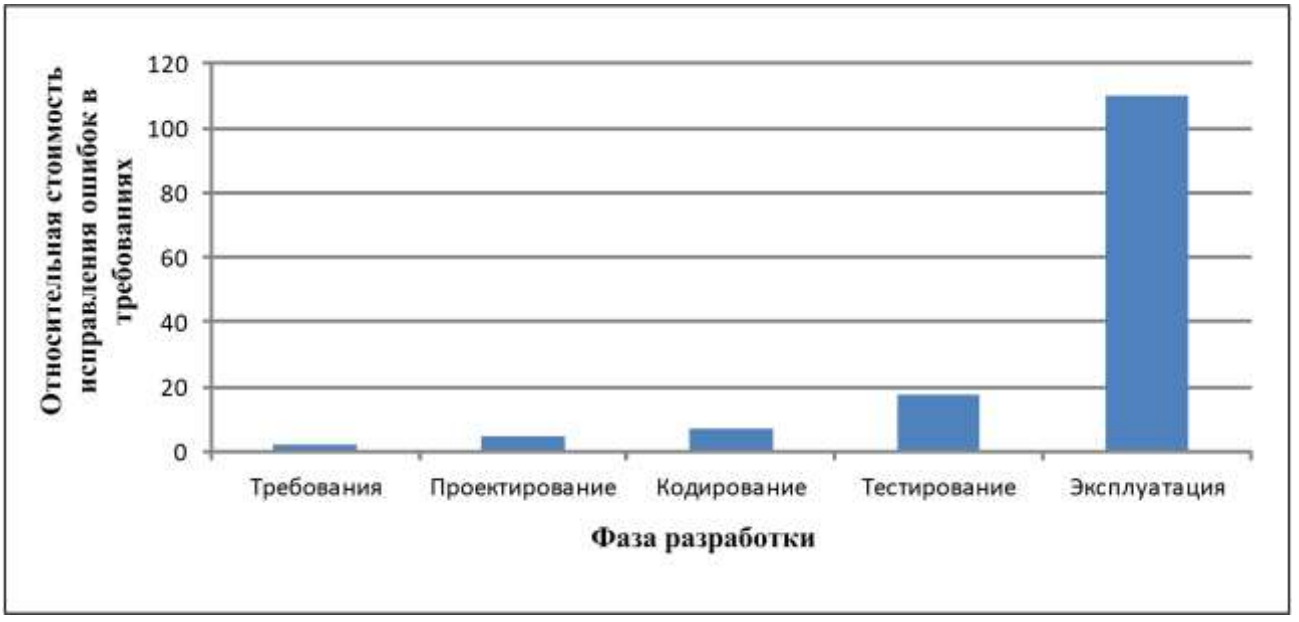


Рис. 1-4. Относительная стоимость исправления ошибок в требованиях в зависимости от того, когда они обнаружены

Дефекты в требованиях представляют собой угрозу успеху проекта, где успех означает выпуск продукта, который удовлетворяет ожиданиям пользователей в качестве и функциональности при соблюдении бюджета и графика проекта. В главе 23 рассказано, как управлять такими рисками, чтобы предотвратить крушение проекта. Некоторые из наиболее общих рисков описаны далее в этой главе.

Недостаточное вовлечение пользователей

Заказчики зачастую не понимают, почему так важно тщательно собрать требования и обеспечить их качество. Разработчики не всегда придают значение вовлечению пользователей в процесс, из-за того, что среди них больше фанатов написания кода, а не любителей возиться с клиентами или же потому, что они считают, что все уже знают о потребностях пользователей. В любом случае трудно добраться до людей, которые непосредственно будут иметь дело с продуктом, а выразители мнения пользователей не всегда понимают, что тем нужно в реальности. Недостаточное вовлечение пользователей ведет к обнаружению ошибок в требованиях на поздних стадиях проекта, а значит, к задержке завершения проекта. Нет альтернативы сотрудничеству разработчиков непосредственно с реальными пользователями на протяжении всего проекта (см. главу 6).

«Разрастание» требований пользователей

Так как требования тщательно прорабатываются и их объем со временем увеличивается,

проект часто выходит за установленные рамки, как по срокам, так и по бюджету. Первоначально принятые планы не всегда основаны на реалистичном понимании размера и сложности требований, ограничение же на модификацию только усиливает проблемы. Эти проблемы частично кроются в частых запросах пользователей на изменения, а частично — в том, какими способами разработчики отвечают на них.

Чтобы управлять границами требований, для начала уточните бизнес-цели проекта, стратегическое представление, рамки, ограничения, критерий успеха и ожидаемую пользу. Оцените, как предполагаемые характеристики или изменения требований отразятся на связанной с ними структуре. Эффективный процесс модификации заставляет интенсивнее работать аналитиков, которые помогут всем заинтересованным лицам принять обдуманные бизнес-решения относительно того, какие изменения следует принять, и увязать их стоимость со временем, ресурсами или возможными компромиссами. Изменения зачастую критически важны для успеха, однако они всегда имеют цену.

По мере того как продукт модифицируется в процессе разработки, его архитектура может медленно разрушаться. «Заплатки» кода затрудняют понимание и поддержку продукта. Добавление кода может стать причиной нарушения твердых и взаимосвязанных принципов проектирования и разрыва связей (McConnell, 1993). Чтобы минимизировать возможность потери качества продукта в результате проблем такого рода, вначале протестируйте, как возможные изменения отразятся на архитектуре и дизайне, а уж затем реализуйте их непосредственно в коде.

Двусмысленность требований

Двусмысленность — страшилка любой спецификации требований (Lawrence 1996). Один из ее симптомов — пользователь имеет возможность интерпретировать одно и то же положение по-разному. Другой — что у нескольких читателей требований возникает разное представление о продукте. В главе 10 указано множество слов и фраз, которые порождают двусмысленность, возлагая ответственность интерпретации на пользователя. Кроме того, двусмысленность зачастую проистекает из неточности и плохой детализации требований, в результате чего разработчикам приходится заполнять возникающие пробелы собственными силами.

Двусмысленность ведет и к формированию различных ожиданий у заинтересованных лиц. Впоследствии некоторые из них удивляются «Что же это получилось?» Разработчики же впустую тратят время, занимаясь не теми проблемами. А тестеры готовятся к проверке не тех особенностей поведения системы.

Один из способов обнаружить двусмысленности — пригласить различных представителей пользователей для официальной экспертизы требований. (Подробнее об этом рассказано в главе 15.) Пусть они просмотрят документ и выскажут свои замечания. Если они интерпретируют требования различными способами, но это имеет смысл для каждого из них, то неясность не проявится сейчас, а гораздо позже, когда исправлять ее будет весьма дорого. Другой способ вскрыть двусмысленность — написать вариант тестирования для требования и построить прототип. Gause и Weinberg (1989) описывают и другие методы.

«Золочение» продукта

Под «золочением» понимают такие ситуации, когда разработчики добавляют функции, которых нет в спецификации, но им кажется, что это понравится пользователям. Зачастую же клиентам не нужны такие избыточные возможности, получается, что время, отведенное на реализацию, тратится впустую. Прежде чем просто вставлять новые функции, разработчики и аналитики должны представить свои творческие идеи на суд заказчиков. Задача команды — четко соблюдать требование спецификации, а не действовать за спиной клиентов без одобрения.

Пользователи иногда требуют функции или элементы интерфейса которые выглядят отлично, но не представляют особой ценности для продукта. Все, что вы захотите добавить, стоит времени и денег, поэтому постарайтесь осознать ценность своевременного выпуска продукта. Чтобы уменьшить «золочение», отслеживайте каждый бит функциональности до его первоисточника, чтобы четко понимать, почему именно он включен в продукт. Применение вариантов использования для извлечения требований поможет сосредоточиться на выборе тех элементов,

которые помогут пользователям выполнять их бизнес-задачи.

Минимальная спецификация

Иногда сотрудников отдела маркетинга или менеджеров охватывает искушение создать урезанный вариант спецификации, как, например, набросок концепций продукта на салфетке. Они ожидают, что разработчики «нарастят мясо» на основе этих набросков, пока проект развивается. Это годится для тесно сплоченной команды, которая занимается небольшой системой, или когда выполняется проект-исследование, или когда требования действительно гибкие (McConnell, 1996), Хотя в большинстве случаев это все-таки раздражает разработчике! (которые могут действовать при некорректных предположениях и с ограниченными инструкциями) и дезориентирует клиентов (которые не получат тот продукт, который они воображали).

Пропуск классов пользователей

Большинство продуктов предназначены для нескольких групп пользователей, которые могут применять различные наборы функций с разной частотой и имеют опыт работы с ПО самого широкого диапазона. Если вы не определили важные классы пользователей для вашего продукта заранее, некоторые потребности клиентов не будут учтены. После идентификации всех классов удостоверьтесь, что каждый из них имеет голос (см. главу 6).

Небрежное планирование

«Я кое-что придумал для нового продукта. Когда вы сможете это сделать?» Не отвечайте на подобный вопрос, пока больше не узнаете о проблеме. Неопределенные, не детализированные требования порождают слишком оптимистические оценки: они «выходят боком», когда возникает перерасход ресурсов и средств. Неподготовленная оценка звучит как обязательство для слушателя. При плохо просчитанной смете проект больше всего страдает из-за затрат на частые изменения требований, пропущенных требований, недостаточного взаимодействия с пользователями, не детализированной спецификации требований и плохого анализа (Davis, 1995). Когда вы высказываете оценку, то указывайте либо качественные показатели («…в лучшем случае…», «…наиболее вероятно...», «… в худшем случае…»), либо количественные, но с учетом вашего личного мнения («Я на 90% уверен, что мы сделаем это за три месяца»).

Выгоды от высококачественного процесса разработки требований

Те организации, которые применяют высококачественные процессы разработки требований получают массу преимуществ. Одно из них — уменьшение объема переделок в течение последней стадии разработки и на протяжении всего длительного периода обслуживания продукта. Эффективность качественных требований не очевидна, и многие люди ошибочно считают, что время, которое тратится на обсуждение требований, просто отсрочивает выпуск продукта. При завершении проекта становится особо ясной ценность приемов повышения качества на ранних стадиях (Wiegers, 1996a).

В удачных процессах создания требований к совместной работе над проектом привлекалось множество заинтересованных лиц, причем на протяжении всего проекта. Сбор требований позволяет команде разработчиков лучше понять пользователей или реалии рынка, что критически важно для любого проекта. Гораздо дешевле понять все особенности до построения ПО, чем после передачи его в руки пользователей.

Привлечение пользователей к созданию требований порождает восхищение продуктом и верность клиентов. Делая упор на задачи пользователей, а не на внешне привлекательные функции, команда избежит необходимости переписывать код, который даже не понадобится. Вовлечение клиентов в процесс снижает вероятность появления ложных ожиданий, возникающих из-за различия того, в чем пользователи нуждаются, и того, что разработчики выпускают. Подумайте о том, как со временем получать отзывы заказчиков. Причем лучше раньше, чем позже. Может быть, в этом вам помогут прототипы, которые стимулируют активность пользователей в этом вопросе. Для разработки требований необходимо время, но его все-таки нужно меньше, чем

для исправления массы проблем в бета-версии или после выпуска.

Кроме того, есть и дополнительные выгоды. Ясное разделение требований на те, что относятся к ПО, оборудованию или подсистемам, взаимодействующим с людьми, позволяет применять системный подход к разработке продукта. Эффективные процессы контроля за изменениями минимизируют неблагоприятные последствия от изменения требований. Не двусмысленно составленные документы облегчают тестирование продукта, что в свою очередь повышает шансы создать высококачественный продукт, который удовлетворит всех пользователей

Никто не станет обещать конкретный возврат инвестиций в процесс улучшения требований. Вы можете мысленно проанализировать, что вы от этого получите. Сначала прикиньте общую сумму вложений. Она складывается из стоимости оценки применяемых сейчас приемов, разработки новых процедур и документирования шаблонов, тренинга команды, покупки книг и инструментов и, возможно, привлечения консультантов со стороны. Наибольшая инвестиция — это время, которое ваша команда тратит на сбор, документирование, просмотр и управление требованиями. Но теперь подумайте о возможных выгодах, которыми вы будете наслаждаться, и о том, сколько времени и денег они вам сэкономят. Невозможно количественно оценить все преимущества, однако они реальны:

- меньше дефектов в требованиях
- меньше переделок;
- меньше ненужных функций
- ниже стоимость модификации
- быстрее разработка
- меньше разобщенности
- меньше расползания границ;
- меньше беспорядка в проекте;
- точнее оценки тестирования;
- выше удовлетворение заказчиков и разработчиков.

Характеристики превосходных требований

Как можно отличить отличную спецификацию требований к ПО от проблематичной? В этом разделе обсуждается несколько характеристик, которым должны отвечать отдельные положения требований, а также соответствующие им черты спецификации в целом (Davis, 1993; IEEE, 1998b). Лучший способ определить, действительно ли ваши требования имеют желаемые атрибуты, — попросить нескольких заинтересованных в проекте лиц внимательно просмотреть спецификацию. Они обнаружат различные виды проблем. Так, например, аналитики и разработчики не смогут точно судить о *полноте* или *корректности* документа, тогда как пользователям не удается оценить технические характеристики.

Характеристики отдельных положений спецификации требований

В идеальном мире каждый отдельный пользователь, бизнес или функциональные требования отвечают различным параметрам качества, которые и описаны в следующих разделах,

Полнота

Каждое требование должно полно описывать функциональность, которую следует реализовать в продукте. То есть оно должно содержать всю информацию, необходимую для разработчиков, чтобы тем удалось создать этот фрагмент функциональности. Если вы понимаете, что данных определенного рода не хватает, используйте пометку «TBD» (to be determined — необходимо определить) на полях как стандартный флаг для выделения такого места. Восполните все пробелы в каждом фрагменте требований, прежде чем приступать к конструированию этой функции.

Корректность

Каждое требование должно точно описывать желаемую функциональность. Для соблюдения корректности необходима связь с источниками требований, например с пожеланиями пользователей или высокоуровневыми системными. Требования к ПО, которые конфликтуют с

родительскими требованиями, нельзя считать корректными. Однако основная оценка здесь— за представителями пользователей, вот почему им или их непосредственным заместителям необходимо предоставлять требования для просмотра.

Осуществимость

Необходима возможность реализовать каждое требование при известных условиях и ограничениях системы и операционной среды. Чтобы не придумывать недостижимые положения, обеспечьте взаимодействие разработчиков с маркетологами и аналитиками требовании на период всего извлечения требований. Разработчики реально оценят, что можно выполнить технически, а что нет, или что сделать можно, но при дополнительном финансировании. Инкрементальная разработка и подтверждающие концепцию прототипы позволяют проверить осуществимость требования.

Необходимость

Каждое требование должно отражать возможность, которая действительно необходима пользователям или которая нужна для соответствия внешним системным требованиям или стандартам. Кроме того, оно должно исходить от лица, которое имеет полномочия на запись положения. Отследите каждое требование вплоть до стадии сбора мнений пользователей, когда выявлялись варианты использования, бизнес-правила или другие источники.

Назначение приоритетов

Назначьте приоритеты каждому функциональному требованию, характеристике или варианту использования, чтобы определить, что необходимо для каждой версии продукта. Если все положения одинаково важны, менеджеру проекта будет трудно справиться с уменьшением бюджета, нарушением сроков, потерей персонала или добавлением новых требований в процессе разработки.

> **Дополнительная информация**
>
> В главе 14 назначение приоритетов обсуждается в деталях.

Недвусмысленность

Все читатели требований должны интерпретировать их одинаково, но естественный язык зачастую грешит многозначностью. Пишите документацию просто, кратко и точно, применяя лексику, понятную пользователям. «Ясность» — цель качества требований, связанная с точностью: читатели должны четко понимать каждое положение. Занесите все специальные и запутанные термины в словарь.

Проверяемость

Попробуйте разработать несколько тестов или примените другие приемы для проверки, например экспертизу или демонстрации, чтобы установить, действительно ли в продукте реализовано каждое требование. Если требование не проверяется, вопрос его корректной реализации становится предметом заключения, а не целью анализа. Неполные, несогласованные, невыполнимые или двусмысленные требования также не проверяются (Drabick 1999).

Характеристики спецификации требований

Недостаточно получить прекрасные отдельные положения. Набор требований, составляющий спецификацию должен отвечать характеристикам, описанным в следующих разделах.

Полнота

Никакие требования или необходимые данные не должны быть пропущены.

Согласованность

Согласованные требования не конфликтуют с другими требованиями такого же типа или с высокоуровневыми пользовательскими, системными или бизнес-требованиями. Несогласованность документов следует устранить до начала процесса разработки. Вы не всегда знаете, какое именно положение некорректно (если какое-то некорректно), пока не выполните исследование.

Рекомендуется записывать автора каждого требования, чтобы узнать, кто его высказал, если конфликт все-таки будет обнаружен.

Способность к модификации

Необходимо обеспечить возможность переработки требований, если понадобится, и поддерживать историю изменений для каждого положения. Для этого все они должны быть уникально помечены и обозначены, чтобы вы могли ссылаться на них однозначно. Каждое требование должно быть записано в спецификации только единожды. Иначе вы легко получите несогласованность, изменив только одно положение из двух одинаковых. Лучше используйте ссылки на первоначальные утверждения, а не дублируйте положения. Модификация спецификации станет гораздо легче, если вы составите содержание документа и указатель. Сохранение спецификации в базе данных коммерческого инструмента управления требованиями сделает их пригодными для повторного использования.

Трассируемость

Трассируемость, или возможность для анализа, можно реализовать как в направлении назад, к первоисточникам, так и вперед, к элементам проектирования и исходному коду, который его реализует, а также к вариантам использования, которые позволяют проверить корректность. реализации. Трассируемые требования помечены соответствующими идентификаторами. Они записаны в структурированной, детализированной форме, в противоположность параграфам в длинной повествовательной форме. Избегайте слипания множества требований в один ком, отдельные требования можно трассировать к различным элементам проектирования и кода.

Дополнительная информация

В главе 10 обсуждается написание высококачественных функциональных требований, а в главе 20 - трассировка.

Вам никогда не удастся создать спецификацию, в которой бы проявлялись все эти атрибуты. Однако если вы будете помнить о них при написании и просмотре требований, вы создадите более качественный документ, а значит, и продукт.

Что теперь?

- Опишите связанные с требованиями проблемы, с которыми вы сталкивались в текущем или предыдущих проектах. Идентифицируйте каждую проблему разработки или управления, а также то влияние, которое она оказывает на проект и ее основные причины.
- Обсудите с членами вашей команды и всеми заинтересованными лицами, влияния этих проблем и их основные причины. Объясните участникам, что они должны противостоять этим трудностям, если когда-нибудь надеются справиться с ними.
- Организуйте однодневный тренинг, посвященный требованиям, для команды, которая работает над вашим проектом. Пригласите основных заказчиков, маркетологов и менеджеров, используя что угодно, чтоб заманить их всех в одну комнату. Тренинг представляет собой эффективное средство сплочения команды. Он позволяет выработать единую лексику и общее понимание эффективных приемов и поведения, чтобы команда смогла решить поставленную перед ней задачу.

Глава 2. Требования с точки зрения клиента

Герхард, старший менеджер компании Contoso Pharmaceuticals, встретился с Синтией, новым сотрудником отдела по разработке информационных систем. «Нам нужно создать новую систему контроля химических препаратов, — начал Герхард. — Она должна обеспечить контроль за всеми химическими контейнерами на складе и. лабораториях. Возможно, благодаря этому химики смогут отказаться от заказа новых контейнеров. Система сэкономит компании уйму денег. Кроме того, отделу контроля безопасности и здравоохранения необходимо отчитываться перед правительством». "Понимаю, почему этот проект важен, Герхард, — сказала Синтия. — Но прежде чем я набросаю график разработки проекта, нам потребуется собрать некоторые требования к системе".

Герхрад удивился: «Что вы имеете в виду? Я только что перечислил вам требования».

«На самом деле вы описали концепцию и некоторые бизнес-цели проекта, — объяснила Синтия. — Бизнес-требования такого высокого уровня не дают достаточно информации, чтобы точно определить, какую систему создавать и сколько времени на это может потребоваться. Я за то, чтобы аналитик поработал с несколькими пользователями и понял, что они ожидают от системы. Затем мы определим, какая функциональность удовлетворит одновременно ваши бизнес-цели и потребности пользователей. Возможно, вам даже не потребуется новая система, чтобы сэкономить средства».

Герахрд никогда раньше не сталкивался с подобной реакцией специалиста отдела информационных систем. «Химики – занятые люди, — запротестовал он. — Вряд ли у них найдется время объяснить все подробности до того, как вы начнете писать программу. Не могут ли ваши люди сами определить конечную цель?»

Синтия попыталась объяснить, почему необходимо выслушать именно пользователей новой системы. «Если мы сами станем угадывать ожидания пользователей, ничего хорошего не выйдет. Мы — разработчики ПО, а не химики. Нам на самом деле неизвестно, чего именно хотят специалисты от системы контроля препаратов. Я по собственному опыту знаю, что, если не потратить время на изучение проблемы до начала программирования, результаты окажутся неудовлетворительными». «У нас нет времени на это, — настаивал Герхард. — Я описан вам мои требования. Теперь, пожалуйста, просто создайте систему. Сообщайте мне о ходе работы».

Такие диалоги регулярно возникают при разработке ПО. Клиенты, которым требуется новая информационная система, зачастую не понимают, насколько важно непосредственно опросить будущих реальных пользователей. Специалисты по маркетингу разработавшие концепцию нового замечательного продукта, считают, что адекватно представляют интересы предполагаемых покупателей. Тем не менее, мнение непосредственных покупателей ПО неоценимо, и заменить его чем-либо иным нельзя. Согласно некоторым современным концепциям разработки ПО, например концепции экстремального программирования (Extreme Programming), клиент, даже если он постоянно занят собственным бизнесом, должен тесно взаимодействовать с командой разработчиков. Как говорится в одной книге по экстремальному программированию, «успех проекта зависит от согласованных действий клиента и программистов» (Jeffries, Anderson и Hendrickson, 2001)

Одна из проблем при формировании требований к ПО состоит в том, что люди путают разные уровни требований: *бизнес-уровень*, *уровень пользователей* и *функциональный*. Герхард перечислил несколько преимуществ, которые, по его мнению, получит компания Contoso, внедрив новую систему контроля химикатов. Однако он не знает требований пользователей, поскольку не работает с этой системой. Пользователи, в свою очередь, могут описать необходимые им возможности системы, но не способны грамотно перечислить функции, которые должны реализовать разработчики для предоставления им таких возможностей.

В этой главе речь пойдет о взаимодействии клиента и разработчика, как о жизненно важной составляющей для успеха проекта по разработке ПО. Я предлагаю вашему вниманию «Билль о правах клиента ПО» и соответствующий «Билль об обязанностях клиента ПО» при формировании требований. Таким образом, я надеюсь прояснить роль клиента, а конкретнее, пользователя, в процессе создания требований,

Страх отказа

Недавно я побывал в отделе информационных систем одной фирмы и услышал печальную историю. Разработчики только что создали новую внутрикорпоративную систему. Пользователи с самого начала не хотели общаться с разработчиками, и когда те с гордостью представили новую систему, пользователи отвергли ее как совершенно неприемлемую. Разработчики, приложившие немало усилий, чтобы удовлетворить потребности пользователей, как они их понимали, испытали настоящий шок. И что же они предприняли? Да просто все исправили. При несоответствии требований систему всегда можно подправить, однако это значительно дороже, чем если бы пользователи описали свои потребности с самого начала.

Безусловно, разработчикам пришлось потратить на доводку проекта больше времени и, значит, отложить следующий проект. Это абсолютно проигрышная ситуация. Разработчики растеряны и расстроены, клиенты разочарованы, так как система не оправдала их ожиданий, а компания потеряла кучу денег. Если бы клиенты с самого начала плотно и постоянно были бы вовлечены в разработку системы, такого неудачного, но, к сожалению, нередкого итога удалось бы избежать.

Кто же клиент?

В широком смысле, клиент — это человек или организация, получающая от продукта прямую или косвенную выгоду. Клиентами ПО можно считать заинтересованных в проекте лиц, требующих, оплачивающих, выбирающих, определяющих, использующих и получающих программный продукт. Как говорилось в главе 1, прочие заинтересованные лица — аналитики требований, разработчики, специалисты по тестированию, технические писатели, менеджеры проектов, а также персонал службы поддержки, юридической службы и маркетинговой службы.

Менеджер Герхард — это клиент, оплачивающий, или спонсирующий, проект по разработке ПО. Клиенты уровня Герхарда определяют бизнес-требования к системе. Они формулируют высокоуровневую концепцию продукта и бизнес-обоснование для его развертывания.

Как вы узнаете из главы 5, бизнес-требования описывают бизнес-цели, которых хочет достичь клиент, компания или другие посредники. Бизнес-требования формируют каркас всего проекта. Любые другие функции и требования к продукту должны удовлетворять бизнес-требования. Тем не менее бизнес-требования не предоставляют разработчикам достаточно информации для создания продукта.

Следующий уровень требований — требования пользователей: их определяют те, кто прямо или косвенно взаимодействуют с продуктом, то есть конечные пользователи. Они способны описать требуемую функциональность, а также ожидаемые качественные характеристики продукта.

Клиенты, предоставляющие бизнес-требования, иногда пытаются говорить от имени пользователей, но обычно они слишком далеки от реальной работы, чтобы точно описать их

нужды. В случае с информационными системами или разработкой нестандартного приложения бизнес-требования должен определять тот, кто платит, а требования пользователей — те, кто будет стучать по клавишам и непосредственно работать с продуктом. Всем участникам проекта рекомендуется проверить согласованность бизнес-требований и требований пользователей.

К сожалению, клиенты иногда не хотят найти время, чтобы поработать с аналитиком требований, собирающим, анализирующим и документирующим требования. Иногда клиенты полагают, что аналитики или разработчики собственными силами определят, что именно необходимо пользователям, и не желают вступать в долгие дискуссии. Если бы все было так просто...

В области разработки коммерческого ПО, где клиент и пользователь зачастую представлены одним лицом, ситуация несколько иная. Представители клиента, например, из отдела маркетинга или менеджеры по продукту, обычно пытаются на свой вкус определить, что клиент счел бы привлекательным. Тем не менее без конечных пользователей сформулировать требования пользователей не удастся (подробнее — в главе 7). В противном случае будьте готовы читать обзоры в журналах, описывающие недостатки вашего продукта, которых удалось бы избежать при активном участии пользователей.

Неудивительно, что бизнес-требования и требования пользователей иногда противоречат друг другу. Бизнес-требования отражают организационную стратегию или бюджетные ограничения, скрытые от пользователей. Пользователи, разочарованные тем, что менеджмент насильно внедряет новую информационную систему, не всегда хотят общаться с разработчиками ПО, считая последних предвестниками неблагоприятного будущего. Избежать этого помогает ясное и полное обсуждение целей и ограничений проекта. Возможно, действительность не удовлетворит никого, но у людей появится возможность понять и подстроиться под нее. Для сглаживания всех конфликтов аналитику необходимо взаимодействовать с полномочными представителями пользователей и менеджеров.

Сотрудничество клиентов и разработчиков

Отличные программные продукты — результат правильно реализованной архитектуры, основанной на требованиях, полученных в результате эффективного, то есть тесного взаимодействия разработчиков и клиентов. Слишком часто сотрудничество разработчиков и клиентов оборачивается враждой. Напряженность также создают менеджеры, изменяющие требования пользователей по собственному усмотрению. В этом случае не выигрывает никто.

Совместная работа возможна только тогда, когда все участники процесса разработки знают, что именно необходимо им для успеха, и когда они понимают и уважают стремление их соратников к успеху. Когда при работе над проектом нагнетается напряжение, очень легко забыть, что у всех участников единая цель — создать успешный программный продукт, ценный для бизнеса и отвечающий чаяниям всех участников проекта,

«Билль о правах клиента ПО» (табл.2-1) содержит 10 положений на выполнении которых клиенты могут на вполне законных основаниях настаивать при общении с аналитиками и разработчиками на этапе формулирования требований к проекту. Каждый пункт этого права описывает соответствующую ответственность аналитиков и разработчиков ПО. «Билль об обязанностях клиента ПО» (табл.2-2), напротив содержит 10 положений, определяющих ответственность клиента перед аналитиком и разработчиком на этапе формулирования требований. Возможно, его стоит назвать «Билль о правах разработчика».

Перечисленные ниже права и обязанности распространяются непосредственно на клиентов в случае, когда программный продукт разрабатывается для внутрикорпоративного использования, на заказ или для определенной группы крупных клиентов. При разработке продуктов для массового рынка интересы клиентов представляют сотрудники, например, отдела маркетинга.

В процессе планирования проекта клиенту и разработчикам следует изучить оба этих списка и постараться достигнуть взаимопонимания. Сильно занятые клиенты, возможно, предпочтут не принимать участия в формулировании требований (то есть попытаются избежать обязанности № 2). Однако мы знаем, что в этом случае значительно повышается риск создания продукта

ненадлежащего качества. Убедитесь, что основные участники процесса формулирования требований понимают и принимают свои обязанности. В случае затруднений организуйте семинар и обсудите спорные пункты. Это позволит избежать трений позже, когда одна сторона будет ожидать чего-то, что другая не может или не желает предоставить.

Таблица 2-1. Билль о правах клиента ПО при формировании требований

Клиент имеет право	
1.	Иметь дело с аналитиком, который разговаривает на вашем языке
2.	Иметь дело с аналитиком, хорошо изучившим ваш бизнес и цели, для которых создается система
3.	Потребовать, чтобы аналитик преобразовал требования, предоставленные вами устно в письменную спецификацию требований к программному продукту
4.	Получить от аналитика подробный отчет обо всех рабочих продуктах, созданных в процессе формулирования требований
5.	На уважительное и профессиональное отношение к вам со стороны аналитиков и разработчиков
6.	Знать о вариантах и альтернативах требований и их реализации
7.	Описать характеристики, упрощающие работу с продуктом
8.	Изменить требования или разрешить использование имеющихся программных компонентов
9.	Получить исчерпывающие сведения о сумме затрат, ожидаемом эффекте и необходимых компромиссах, которые возникают в связи с изменениями в ПО
10.	Потребовать, чтобы система функциональностью и качеством удовлетворяла требования заказчика

Таблица 2-2. Билль об обязанностях клиента ПО при формировании требований

Клиент обязан	
1.	Ознакомить аналитиков и разработчиков с особенностями вашего бизнеса
2.	Потратить столько времени, сколько необходимо на объяснение требований
3.	Точно и конкретно описать требования к системе требований
4.	Принимать своевременные решения
5.	Уважать определенную разработчиком оценку стоимости и возможность реализации ваших требований
6.	Определять приоритеты требований
7.	Просматривать документы с требованиями и оценивать прототипы
8.	Своевременно сообщать об изменениях требований
9.	Поддерживать принятый в организации-разработчике порядок контроля изменений
10.	Уважительно относиться к методам, с помощью которых аналитики создают требования

Ловушка

Не предполагайте, что участники проекта интуитивно знают, как сотрудничать при формулировании требований. Потратьте время и обсудите, каким образом организовать взаимодействие наиболее эффективно.

Билль о правах клиента ПО

Право № 1. Иметь дело с аналитиком, который разговаривает на вашем языке

При обсуждении требований следует выяснить потребности и задачи вашего бизнеса, используя при этом принятую в вашем бизнесе терминологию. Заставьте аналитиков говорить с вами на вашем языке (возможно, для этого им следует предоставить небольшой словарь), не продирайтесь в разговорах с ними через компьютерный жаргон.

Право № 2. Иметь дело с аналитиком, который изучил ваш бизнес и цели

Выявляя требования, аналитики смогут лучше понять задачи вашего бизнеса и осознать, какое место уготовано создаваемому ПО в вашем бизнесе. Это поможет им удовлетворить ваши ожидания. Пригласите разработчиков и аналитиков к себе в офис. Если заказанная система заменит существующее приложение, предложите разработчикам поработать с ним. Таким образом, им будет легче понять его сильные и слабые стороны и то, как его можно усовершенствовать.

Право № 3. Потребовать, чтобы аналитик преобразовал требования, предоставленные вами устно, в письменную спецификацию требований к ПО

Аналитик отсортирует всю информацию, предоставленную вами и другими клиентами, и выявит на основе бизнес-требований, бизнес-правил, функциональных требований, целей качества, возможных решений и прочих элементов область применения. Конечный итог этого анализа — спецификация требований к программному обеспечению (software requirements specification, SRS), представляющая собой соглашение между разработчиками и клиентами о функциях, качестве и ограничениях создаваемого продукта. Она должна быть организована и написана так, чтобы вы ее легко поняли. Если вас что-то не устраивает, выскажите свое мнение: документ должен точно и полно отражать ваши нужды и чаяния.

Право № 4. Получить подробный отчет обо всех рабочих продуктах, созданных в процессе формулирования требований

Аналитик может представить требования с помощью различных диаграмм, дополняющих текст спецификации требований к программному обеспечению. В главе I рассматриваются несколько моделей анализа. Альтернативные представления требований очень важны, поскольку иногда графики позволяют яснее выразить некоторые особенности по ведения системы, например последовательность выполняемых действий. И хотя эти диаграммы могут показаться непривычными, понять их несложно. Попросите аналитика объяснить назначение каждой из них (а также других продуктов, созданных в процессе формулирования требований), смысл обозначений и процедуру проверки диаграмм на предмет ошибок.

Право № 5. На уважительное и профессиональное отношение к вам со стороны аналитиков и разработчиков

Если клиенты и разработчики не понимают друг друга, обсуждение требований может обернуться большим разочарованием. Совместная работа позволяет открыть друг другу глаза на проблемы, с которым сталкивается каждая из этих групп. Клиенты, участвующие в процессе выработки требований, имеют полное право требовать от аналитиков и разработчиков уважительного отношения к себе и бережного отношения к затраченному времени. В свою очередь, и клиентам следует оказывать уважение разработчикам, ведь они все вместе

сотрудничают для достижения общей цели — успешного проекта.

Право № 6. Знать о вариантах и альтернативах требований и их реализации

Чтобы гарантировать, что новая система не будет автоматизировать неэффективные или устаревшие процессы, аналитик должен знать, почему существующие системы не годятся для ваших бизнес-процессов. Аналитики, основательно разбирающиеся в предметной области бизнеса, иногда предлагают те или иные усовершенствования ваших бизнес-процессов. Полезен и аналитик, творчески подходящий к делу: он предлагает новые возможности программы, о которых пользователи даже и не мечтали.

Право № 7. Описать характеристики, упрощающие работу с продуктом

Вполне вероятно, что аналитики спросят вас о характеристиках ПО, выходящих за рамки функциональных потребностей пользователей. Благодаря им программный продукт становится более удобным в работе, что позволяет клиентам эффективнее выполнять их задачи. Иногда пользователи просят, чтобы продукт был дружественным, надежным или эффективным, однако эти термины слишком субъективны, чтобы помочь разработчикам. Поэтому аналитик должен выяснить конкретные характеристики, означающие для вас дружественность, надежность или эффективность. Подробнее — в главе 12.

Право № 8. Изменить требования или разрешить использование имеющихся программных компонентов

Отношение к требованиям должны быть гибкими. Аналитику могут быть известны готовые программные компоненты, которые практически полностью удовлетворят некоторые названные вами потребности. В таком случае вам следует скорректировать отдельные требования, чтобы разработчики могли использовать имеющееся ПО. Разумные возможности применения существующих модулей сэкономит ваши время и деньги. Если вы посчитаете разумным включить в свой продукт готовые коммерческие компоненты, будьте готовы проявить гибкость, поскольку характеристики таких компонентов врядли будут точно соответствовать вашим потребностям.

Право № 9. Получить исчерпывающие сведения о сумме затрат, ожидаемом эффекте и необходимых компромиссах, которые возникают в связи с изменениями в ПО

Зная, что один вариант дороже другого, разные люди делают разный выбор. Для принятия правильных бизнес-решений необходимы данные об эффективности и стоимости предполагаемого изменения требований. У вас есть право ожидать от аналитиков добросовестной оценки эффекта, затрат и компромиссов. Разработчики не должны завышать предполагаемую стоимость изменения только потому, что не хотят его реализовывать.

Право № 10. Потребовать, чтобы система функциональностью и качеством удовлетворяла требования заказчика

Такого завершения проекта желают все, однако он возможен, только если вы сумеете донести до разработчиков всю информацию, которая поможет им создать подходящий продукт, и если разработчики четко изложат вам все варианты и ограничения. Убедитесь, что вы высказали все свои ожидания и предположения; в противном случае программисты, скорее всего, не смогут реализовать их.

Билль об обязанностях клиента ПО

Обязанность № I Ознакомить аналитиков и разработчиков с особенностями вашего бизнеса

Только вы можете полноценно познакомить разработчиков с концепциями и терминологией своего бизнеса. Вам не надо делать аналитиков экспертами в предметной области, основная задача — помочь им понять ваши проблемы и цели. Не ожидайте, что аналитики постигнут нюансы и

неявные особенности бизнеса. Скорее всего у них нет знаний, воспринимаемых вами и вашими коллегами, как должное. Невысказанные предположения о таких знаниях могут вызвать проблемы в дальнейшем.

Обязанность № 2. Потратить столько времени, сколько необходимо, на объяснение требований

Клиенты — занятые люди, и те, кто участвует в формулировании требований — обычно самые занятые из них. Тем не менее вы обязаны потратить время на участие в совещаниях, мозговых штурмах, интервью и прочих процедурах, необходимых для выявления требований. Иногда аналитик считает, что понял вашу идею, а позже сообразит, что ему необходимы дополнительные разъяснения. Пожалуйста, терпеливо относитесь к такому поэтапному подходу к формулированию и прояснению требований; это — природа сложного человеческого общения и ключ к успеху. Терпимо относитесь к глупым, на ваш взгляд, вопросам; хороший аналитик задает вопросы, которые заставляют вас говорить.

Обязанность № 3. Точно и конкретно описать требования к системе

Весьма заманчиво оставить требования неопределенными и нечеткими, ведь прояснять подробности так утомительно и долго. Тем не менее на каком-то этапе разработки участникам проекта необходимо устранить все неоднозначности и неточности. Вам, как клиенту, и карты в руки. В противном случае угадывать, что же именно вам нужно, будут разработчики.

В спецификации требований к программному обеспечению рекомендуется использовать пометки «TBD» (to be determined — необходимо определить), указывающие на необходимость дополнительных исследований, анализа и информации. Тем не менее иногда такие маркеры используют в случаях, когда конкретное требование трудно определить точно и никто не хочет с ним возиться. Попытайтесь прояснить цель каждого требования, чтобы аналитик мог точно выразить его в спецификации требований к ПО. Если точное определение невозможно, выработайте совместно процедуру, которая позволит достичь необходимой ясности. Зачастую в таких случаях применяют метод прототипов — когда вы совместно с разработчиками формулируете требования поэтапно и постепенно.

Обязанность № 4. Принимать своевременные решения

Точно так же, как и подрядчик при строительстве дома, аналитик задаст вам множество вопросов и попросит выбрать различные варианты и принять решения. Это может быть согласование противоречивых запросов от разных клиентов, выбор между конфликтующими атрибутами качества и оценка точности информации. Клиенты, обладающие соответствующими полномочиями, должны принять эти решения, когда их об этом попросят. Зачастую работа стопорится, так как клиент не может принять окончательного решения, и поэтому, медля с ответом, он затягивает работу над проектом.

Обязанность № 5. Уважать определенную разработчиком оценку стоимости и возможность реализации ваших требований

Все программные функции чего-то стоят. Разработчики лучше всего способны оценить эту стоимость, хотя многие из них и не имеют опыта в этом деле. Может оказаться, что некоторые необходимые вам функции технически неосуществимы или их реализация слишком дорога, например недостижимая в рабочей среде производительность или доступ к данным, которые системе просто недоступны. Даже если сведения об осуществимости и стоимости некоторых функций, сообщенные разработчиком, вам не понравятся, следует уважать эту оценку.

Иногда требования можно изменить так, что они станут дешевле и достижимее. Например, «мгновенное» выполнение операции реализовать нельзя, однако вполне по силам задать минимальный временной интервал (скажем, «в пределах 50 миллисекунд»).

Обязанность № 6. Определять приоритеты требований

Лишь при работе над ограниченным кругом задач можно рассчитать время и ресурсы для

реализации всей желаемой функциональности. Определить, какие возможности необходимы, какие полезны и без каких пользователи обойдутся, — важная составляющая анализа требований. Именно вы определяете эти приоритеты, поскольку разработчики не в состоянии расставить приоритеты. Чтобы вам облегчить задачу, разработчики предоставляют информацию о стоимости и рисках всех требований. Определив приоритеты, вы поможете разработчикам вовремя и с минимальными затратами создать максимально эффективный продукт.

Обязанность № 7. Просматривать документы с требованиями и оценивать прототипы

Как говорится в главе 15, просмотр требований — одна из наиболее значимых операций, обеспечивающих качество ПО. Участие клиентов в таких просмотрах — единственный способ узнать, отражены ли потребности клиента наиболее полно и точно. Если клиенты не уверены в точности задокументированных требований, им следует как можно раньше сообщить об этом лицам, ответственным за реализацию требований, и предложить варианты усовершенствования.

Читая спецификацию требований, не всегда удается четко представить работу программы. Чтобы лучше понять потребности клиента выявить оптимальные способы их удовлетворения, разработчики иногда создают прототипы предполагаемого продукта. Ваша ответная реакция на эти предварительные, частичные или пробные версии дает разработчикам необходимую информацию. Поймите, что прототип не является рабочим продуктом, и не давите на разработчиков.

Обязанность № 8. Своевременно сообщать об изменениях требований

Постоянное изменение требований — серьезная угроза для возможности своевременной сдачи проекта командой разработчиков. Изменение неизбежно, но чем позже в ходе разработки о нем сообщаете, тем более сильное влияние оно окажет. Изменения могут стоить дорого, и сроки будут нарушены, если клиент пожелает реализовать в практически завершенном проекте новую функциональность. Как только вы решите изменить требования, сразу же сообщите об этом аналитику, с которым работаете.

Обязанность № 9. Поддерживать принятый в организации-разработчике порядок контроля изменений

Для снижения негативного эффекта от изменений выполняйте необходимые процедуры контроля изменений, определенные в проекте. Это гарантирует, что никакое требование не потеряется, будет проанализирован эффект каждого изменения и все предполагаемые коррективы будут согласованы друг с другом.

Обязанность № 10. Уважительно относиться к методам, с помощью которых аналитики создают требования

Сбор и проверка требований — одни из самых трудных задач в разработке ПО. У всех подходов, применяемых аналитиками, есть рациональная основа. И хотя операции по созданию требований могут вас разочаровать, время, затраченное, чтобы разобраться в требованиях, — не пропадет даром. Процесс окажется менее болезненным, если вы разберетесь в методах, используемых аналитиками для создания требований. Не стесняйтесь и просите аналитиков объяснить, зачем необходимы те или иные сведения и почему они просят вас поучаствовать в некоторых операциях по созданию требований.

Что насчет подписи?

Результатом сотрудничества клиента и разработчика считается соглашение по требованиям к продукту. Многие организации просят клиента поставить свою подпись на документах с требованиями: это означает, что клиент их подтверждает. Все участники процесса утверждения требований должны понимать свою меру ответственности. Подписав документ, клиент не может впоследствии заявить: «Мне дали лист бумаги с моим именем под чертой, и я на этой черте расписался, иначе разработчики не начали бы программировать». Если такой заказчик захочет через некоторое время изменить требования или его не устроит результат работы, детским лепетом прозвучат слова: «Ну да, я подписал эти требования, но у меня не было времени их читать. Я

доверял вам, ребята - и вы меня подвели!».

Иногда проблему создает менеджер по разработке, он не должен рассматривать подпись как способ сделать требования неизменными. Когда клиент просит о каких-то изменениях, глупо указывать ему на спецификацию требований к ПО и протестовать: «Но вы же подписали эти требования, и именно такую систему мы и создаем. Если вам нужно было что-то другое, следовало сказать об этом раньше».

Реальность такова, что на ранних этапах работы над проектом известна лишь часть требований, со временем они меняются. Утверждение требований — вот та самая операция, которая закрывает прение возникающие в процессе создания требований. Тем не менее участникам необходимо подтвердить свои слова подписями.

Ловушка

Не используйте подпись в качестве оружия. Применяйте ее как завершение этапа проекта и выработайте четкое коллективное понимание того, как подпись повлияет на возможные в будущем изменения.

Гораздо важнее ритуала подписи концепция создания базовой, или основной версии (baseline) требований — «моментального снимка документа на какой-то момент времени. Текст под подписью на странице базовой версии должен звучать примерно так: «Подтверждаю что настоящий документ наилучшим образом представляет наше понимание требований к этому проекту на данный момент и что описанная система удовлетворит наши потребности. Я согласен вносить в будущем изменения в эту базовую версию в соответствии с процессов изменения требований, определенным в проекте. Я понимаю, что утвержденные изменения могут потребовать переоценки стоимости, ресурсов и сроков сдачи проекта». Благодаря этому документу, команда получает возможность контролируемо изменять границы проекта, анализируя влияние каждого предполагаемого изменения на сроки сдач и прочие факторы успеха проекта.

Согласованное понимание значения подписи позволяет избежать прений, возникающих при изменении взглядов на требования, а также при изменении рыночных и бизнес-требований к проекту. Если клиенты боятся, что им не удастся внести коррективы после того, как утвердят спецификацию требований к ПО, они станут затягивать утверждение, в результате чего возникает страшная штука — паралич аналитического процесса. Осмысленный процесс создания без соглашения о требованиях дает уверенность всем участникам проекта:

- клиент уверен, что разработчики будут взаимодействовать с ним для создания нужной системы, даже если перед началом работы над проектом они не успели продумать все требования:
- клиент уверен, что границы проекта не выйдут из-под контроля, поскольку решения относительно границ принимает он;
- менеджер проекта уверен, что команда разработчиков получила достойного делового партнера, который совместно с разработчиками готов отвечать за качество продукта;
- аналитики требований уверены в том, что могут управлять изменениями проекта, минимизируя хаос.

Скрепив начальные операции по формулированию требований внешним соглашением, вы сплотите клиентов и разработчиков на пути к успеху проекта.

Что теперь?

- Определите, кто из клиентов предоставит бизнес-требования и пользовательские требования к проекту. Какие пункты Билля о правах» и «Билля об обязанностях» эти клиенты понимают, принимают и используют на практике, а какие — нет?
- Обсудите «Билль о правах» со своими основными клиентами и узнайте, нет ли у них ощущения бесправности. Обсудите «Билль об обязанностях» и выясните, какие из обязанностей клиенты принимают. Измените «Билль о правах» и «Билль об обязанностях» так, чтобы все стороны приняли концепцию дальнейшего сотрудничества.
- Если вы участвуете в проекте по разработке ПО в качестве клиента и чувствуете, что ваши права попираются, обсудите это с менеджером проекта или аналитиком требований. Согласитесь выполнять «Билль об обязанностях», это позволит наладить более тесные и дружеские рабочие отношения с программистами.
- Продумайте, какое значение на самом деле имеет подпись для подтверждения ваших документов с требованиями.

Глава 3. Хорошие приемы создания требований

Лет десять-пятнадцать назад мне нравились методики разработки программного обеспечения — комплексные наборы моделей и приемов, рассчитанные на полное решение проектов. Тем не менее сейчас я предпочитаю выявлять и использовать приемы, проверенные практикой отрасли. Лучший прием на сегодня таков: не разрабатывать или не приобретать полнофункциональное решение, а дополнить имеющийся пакет средств разработчика различными возможностями, позволяющими решать самые разнообразные проблемы. Даже если вы возьмете на вооружение коммерческую методику, переделайте ее так, чтоб она оптимально соответствовала вашим потребностям, и дополните ее компоненты эффективными приемами из собственного пакета разработчика.

Понятие «лучших приемов» довольно спорно: кто решает, что лучше, и на каком основании? Можно создать группу экспертов или исследователей и проанализировать проекты различных организаций (Browr, 1996; Brown, 1999; Dutta, Lee и Van Wassenhove, 1999).Таким образом, вам удастся выявить приемы, которые обычно эффективно применялись в успешных проектах, а в неудачных — показали себя плохо или не применялись вообще. В процессе работы эксперты согласованно определяют, какие действия неизменно дают превосходный результат обычно их называют лучшими приемами. Подразумевается, что это — высокоэффективные способы, которые при работе над определенными проектами и в определенных ситуациях существенно повышаю шансы профессиональных разработчиков ПО на успех.

В табл. 3-1 перечислено приблизительно 50 приемов; они разделены на 7 категорий. Надеюсь, они помогут разработчикам более эффективно формулировать требования. Некоторые приемы относятся к нескольким категориям, но в таблице они указаны только один раз. Приемы годятся не для всех ситуаций, и поэтому не стоит слепо следовать сценарию, а руководствоваться трезвым расчетом, здравым смыслом и опытом. Заметьте: не все из указанных приемов относятся к лучшим, проверенным практикой отрасли, и именно поэтому глава называется «Хорошие приемы создания требований», а не «Лучшие приемы». Сомневаюсь, что когда-либо будет проведена систематическая оценка всех их на предмет выбора лучшего. Тем не менее, многие практики, и я в том числе, считают данные способы эффективными (Sommerville и Sawyer, 1997; Hofmann и Lehner, 2001). Я кратко описываю каждый способ, а также указываю ссылки на другие главы и источники. В заключительном разделе главы приведен примерный план формулирования требований — последовательность действий, подходящая для большинства проектов по разработке ПО.

Таблица 3-1. Приемы формулирования требований

Обучение	Управление требованиями	Управление проектом
Обучите аналитиков требований	Определите процесс управления изменениями	Выберите соответствующий цикл разработки проекта
Ознакомьте представителей пользователей и менеджеров с требованиями	Установите границы для контроля изменений	Планируйте на основании требований
Обучите разработчиков основам предметной области	Проанализируйте, какое влияние оказывают изменения	Своевременно пересматривайте обязательства
Создайте словарь бизнес-терминов	Определите базовую версию и управляйте версиями требований	Управляйте рисками, касающимися требований
	Отслеживайте хронологию изменений	Отслеживайте объем работ по реализации требований
	Отслеживайте состояние требований	Делайте выводы из полученного опыта

	Определите изменяемость требований	
	Используйте утилиту управления требованиями	
	Создайте матрицу связей требований	

Разработка требований			
Выявление требований	**Анализ**	**Спецификации**	**Проверка**
Определите процесс формулирования требований	Нарисуйте контекстную диаграмму	Используйте шаблон спецификации требований к ПО	Изучите документы с требованиями
Определите образ и границы проекта	Создайте прототипы	Определите источники требований	Протестируйте требования
Определите классы пользователей	Проанализируйте осуществимость	Задайте каждому требованию уникальный идентификатор	Определите критерии приемлемости
Выделите из пользователей ярого сторонника продукта	Расставьте приоритеты для требований	Задокументируйте бизнес-правила	
Создайте фокус-группы	Смоделируйте требования	Укажите атрибуты качества	
Определите назначение продукта	Создайте словарь терминов		
Определите системные события и реакцию на них	Распределите требования по подсистемам		
Проводите совместные семинары, упрощающие сбор требований	Воспользуйтесь технологией развертывания функций качества (Quality Function Deployment)		
Наблюдайте за пользователями на рабочих местах			
Изучите отчеты о проблемах			
Используйте требования многократно			

Обучение

Не все разработчики ПО знают теорию сбора требований. Тем не менее на определенном этапе профессиональной деятельности IV из них осваивают обязанности аналитика требований и работают с клиентами, собирая, анализируя и документируя требования. Однако неразумно ожидать, что все разработчики умеют формулировать требования к ПО и общаться с клиентом. Обучение позволяет повысить профессиональные навыки сотрудников, выполняющих роли, но не может компенсировать нехватку навыков межличностного общения и отсутствие интереса к делу.

Процесс формулирования требований весьма важен, и поэтому все участники проекта должны понимать концепцию и приемы формулирования требований. Эффективный способ создать команду — собрать участников проекта на однодневный семинар, где и обсудить все требования. Стороны смогут глубже понять проблемы, стоящие перед остальными, а также то, что необходимо участникам друг от друга для успеха проекта. Аналогичным образом разработчиков следует просветить о концепциях и терминологии предметной области. Подробнее об этом — в следующих главах:

- в главе 4 — об обучении аналитиков требований;
- в главе 10 — о создании бизнес-словаря проекта.

Обучение аналитиков требований. Всем членам команды, которые будут исполнять функции аналитиков, необходимо научиться приемам формулирования требований — это может занять несколько дней. Квалифицированный аналитик требований терпелив и методичен, обладает навыками межличностного общения и коммуникативными навыками, сведущ в предметной области и знает множество способов, формулирования требований к ПО.

Ознакомление пользователей и менеджеров с требованиями. Пользователи, которые будут принимать участие в разработке ПО, должны пройти непродолжительный тренинг (один-два дня), чтоб научиться формулировать требования. Он полезен и для менеджеров по разработке и по работе с клиентами. Обучение поможет понять особое значение выделения требований, суть процесса их разработки, а также опасность пренебрежения ими. Посетив мои семинары по требованиям, некоторые пользователи замечали, что стали теплее относиться к разработчикам ПО.

Ознакомление разработчиков с концепциями предметной области. Чтобы помочь разработчикам в общих чертах понять предметную область, проведите семинар, на котором познакомьте их с бизнесом клиента, терминологией и назначением создаваемого продукта. Это уменьшит вероятность путаницы, непонимания и доработок. Можно также на время проекта назначить каждому разработчику «личного пользователя», который будет разъяснять профессиональные термины и бизнес-концепции. Лучше, если это будет настоящий фанат продукта.

Создание бизнес-словаря. Словарь со специализированными терминами из предметной области снизит вероятность непонимании. Включите в него синонимы, термины, имеющие несколько значений, и термины, имеющие в предметной области и повседневной жизни разные значения.

Выявление требований

В главе 1 обсуждались три уровня требований: бизнес-уровень, пользовательский уровень и функциональный. Они собираются из разных источников на различных этапах работы над проектом, имеют различные цели и аудиторию и должны документироваться по-разному. Бизнес-требования не должны отменять каких-либо значительных пользовательских требований; кроме того, необходимо проследить, как из конкретных требований пользователей проистекают все функциональные требования. Также следует выявить нефункциональные характеристики, например предполагаемые качество и производительность. Подробнее об этом — в следующих главах:

- в главе 3 — о процессе формулирования требований;
- в главе 5 — об определении образа и границ проекта;
- в главе 6 — об определении классов пользователей и их характеристик, выборе

сторонника продукта из каждого класса пользователей, наблюдении за пользователями на рабочих местах;

- в главе 7 — о проведении совместных семинаров;
- в главе 8 — о работе с представителями пользователей, определении системных событий и реакции на них;
- в главе 22 — об определении процесса формулирования требований.

Определение процесса формулирования требований. Задокументируйте этапы выявления, анализа, определения и проверки требований. Наличие инструкций по выполнению ключевых операций поможет аналитикам качественно и согласованно выполнить их работу. Кроме того, вам будет проще поставить задачи по созданию требований и графики, а также продумать необходимые ресурсы.

Определение образа и границы проекта. Документ об образе и границах проекта содержит бизнес-требования к продукту. Описание образа проекта позволит всем заинтересованным лицам в общих чертах понять назначение продукта. Границы проекта определяют, что следует реализовать в этой версии, а что — в следующих. Образ и границы проекта — хорошая база для оценки предлагаемых требований. Образ продукта должен оставаться от версии к версии относительно стабильным, но для каждого выпуска необходимо составлять отдельный документ о границах.

Определение классов пользователей и их характеристик. Чтобы не упустить из виду потребности отдельных пользователей, выделите их в группы. Например, по частоте работе с ПО, используемым функциям, уровню привилегий и навыкам работы. Опишите их обязанности, местоположение и личные характеристики, способные повлиять на архитектуру продукта.

Выбор сторонника продукта (product champion) в каждом классе пользователей. Это человек, который сможет точно передавать настроения и нужды клиентов. Он представляет потребности определенного класса пользователей и принимает решения от их лица. В случае разработки внутрикорпоративных информационных систем, когда все пользователи — ваши коллеги, такого человека выбрать проще. При коммерческой разработке порасспросите клиентов или используйте площадки бета-тестирования. Выбранные вами люди должны принимать постоянное участие в проекте и обладать полномочиями для принятия решений, касающихся пользовательских требований.

Создание фокус-групп типичных пользователей. Определите группы типичных пользователей предыдущих версий вашего продукта или похожих. Выясните у них подробности о функциональности и качественных характеристиках разрабатываемого продукта. Фокус-группы особенно значимы при разработке коммерческих продуктов, когда приходится иметь дело с большой и разнородной клиентской базой. В отличие от сторонников продукта, у фокус-групп обычно нет полномочий на принятие решений.

Работа с пользователями для выяснения назначения продукта. Выясните у пользователей, какие задачи им требуется выполнять средствами ПО. Обсудите, как должен клиент взаимодействовать с системой для выполнения каждой такой задачи. Воспользуйтесь стандартным шаблоном для документирования всех задач и для каждой сформулируйте функциональные требования. Похожий способ, часто применяемый в правительственных проектах, — создать документ с концепциями операций (ConOps), где указаны характеристики новой системы с точки зрения пользователя (IEEE, 1998a).

Определение системных событий и реакции на них. Определите возможные внешние события и ожидаемую реакцию системы на них. Это могут быть сигналы и данные, получаемые от внешнего оборудования, а также временные события, вызывающие ответную реакцию, например ежевечерняя передача данных, генерируемых системой, внешнему объекту. В бизнес-приложениях бизнес-события напрямую связаны с задачами.

Проведение совместных семинаров. Совместные семинары по выявлению требований, где тесно сотрудничают аналитики и клиенты — отличный способ выявить нужды пользователей и составить наброски документов с требованиями (Gottesdiener, 2002). Конкретные примеры таких семинаров - Joint Requirements Planning (JRP — совместное планирование требований) (Martin, 1991) и Joint Application Development (JAD-совместная разработка приложений) (Wood и Silver, 1995).

Наблюдение за пользователями на рабочих местах. Наблюдая за работой пользователей, выявляют контекст потенциального применения нового продукта (Beyer и Holtzblatt, 1998). Простые диаграммы рабочих потоков, а также диаграммы потоков данных позволяют выяснить, где, как и какие данные задействовал пользователь. Документируя ход бизнес-процесса, удается определить требования к системе предназначенной для поддержки этого процесса. Иногда даже выясняется, что для выполнения деловых задач клиентам вовсе и не требуется новое ПО (McGraw и Harbison, 1997).

Изучение отчетов о проблемах работающих систем с целью поиска новых идей. Поступающие от клиентов отчеты о проблемах и предложения о расширении функциональности — отличный источник идей о возможностях, которые можно реализовать в следующей версии или новом продукте. За подобной информацией стоит обратиться и к персоналу службы поддержки.

Повторное использование требований в разных проектах. Если необходимая клиенту функциональность аналогична уже реализованной в другом продукте, подумайте, готовы ли клиенты гибко пересмотреть свои требования для использования существующих компонентов. Требования, соответствующие бизнес-правилам компании, можно применить в нескольких проектах. Это требования к безопасности определяющие порядок доступа к приложениям, и требования, соответствующие постановлениям правительства, например Закон о гражданах США с ограниченными возможностями (Americans with Disabilities Act).

Анализ требований

Анализ требований подразумевает их детализацию, гарантирующую, что требования понимают все заинтересованные лица, а также тщательное исследование требований на предмет ошибок, пробелов и других недостатков. Кроме того, анализ включает создание прототипов, анализ осуществимости и согласование приоритетов. Цель анализа — достаточно качественно и подробно описать требования, позволяющие менеджерам реалистично оценить все затраты на проект, а техническому персоналу — начать проектирование, сборку и тестирование.

Зачастую отдельные требования стоит представить несколькими способами, например в текстовой и графической форме. Это позволит выявить их особенности и проблемы, не заметные при представлении одним способом (Davis, 1995). Также это помогает всем заинтересованным лицам выработать согласованное представление об итогах разработки продукта. Подробнее об этих темах — в следующих главах:

- в главе 5 — о создании контекстной диаграммы;
- в главе10 — о создании словаря данных;
- в главе 11 — о моделировании требований;
- в главе 13 — о создании пользовательского интерфейса и технических прототипов;
- в главе14 — об определении приоритетов требований;
- в главе 17 — о распределении требований по подсистемам.

Создание контекстной диаграммы. Контекстная диаграмма — простая 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.

Создание пользовательского интерфейса и технических прототипов. Если разработчики или пользователи не совсем уверены насчет требований, создайте прототип — частичную, возможную или предварительную версию продукта, которая сделает концепции и возможности более осязаемыми. Оценка прототипа поможет всем заинтересованным лицам достичь взаимопонимания по решаемой проблеме.

Анализ осуществимости требований. Проанализируйте, насколько реально реализовать каждое требование при разумных затратах и с приемлемой производительностью в предполагаемой среде. Рассмотрите риски, связанные с реализацией каждого требования, включая конфликты с другими требованиями, зависимость от внешних факторов и препятствия технического характера.

Определение приоритетов требований. Воспользуйтесь аналитическим подходом и

определите относительные приоритеты реализации функций продукта, решаемых задач или отдельных требований. На основании приоритетов установите, в какой версии будет реализована та или иная функция или набор требований. Подтверждая изменения, распределите все их по конкретным версиям и включите в план выпуска этих версий затраты, необходимые на внесение изменений. В ходе работы над проектом периодически корректируйте приоритеты в соответствии с потребностями клиента, условиями рынка и бизнес-целями.

Моделирование требований. В отличие от подробной информации, представленной в спецификации требований к ПО или пользовательского интерфейса прототипа, графическая модель анализа отображает требования на высоком уровне абстракции. Модели позволяют выявить некорректные, несогласованные, отсутствующие и избыточные требования. К таким моделям относятся диаграммы потоков данных, диаграммы «сущность — связь», диаграммы перехода состояний, называемые также ***автоматами*** (statecharts), карты диалогов, диаграммы классов, диаграммы последовательностей, диаграммы взаимодействий, таблицы решений и деревья решений.

Создание словаря терминов. В нем соберите определения всех элементов и структур данных, связанных с системой, что позволяет всем участникам проекта использовать согласованные определения данных. На стадии работы над требованиями словарь должен содержать определения элементов данных, относящихся к предметной области, чтобы клиентам и разработчикам было проще общаться.

Распределение требований по подсистемам. Требования к сложному продукту, включающему несколько подсистем, следует соразмерно распределять между программными, аппаратными и операторскими подсистемами и компонентами (Nelsen, 1990). Как правило, это осуществляет системный инженер или разработчик.

Применение технологий развертывания функций качества. Технология развертывания функций качества (Quality Function Deployment, QFD) — точная методика, соотносящая возможности и атрибуты продукта с их значимостью для клиента (Zultner, 1993; Pardee, 1996). Она позволяет аналитически выявить функции, которые максимально удовлетворят потребности клиента.

Технология развертывания функций качества рассчитана на три класса требований: ожидаемые, о которых клиент может не упомянуть, но будет расстроен, если их не окажется в продукте, обычные требования и отдельные, специальные требования, которые обеспечивают удобство работы клиентам, но отсутствие которых не влечет санкций со стороны клиента.

Спецификации требований

Независимо от способа выявления требований, документировать их нужно так, чтоб это обеспечивало удобный доступ и просмотр. Зафиксировать бизнес-требования можно в положении об образе и границах проекта. Пользовательские требования обычно представляют в виде вариантов применения или таблиц «событие — реакция». Спецификация требований содержит подробные функциональные и нефункциональные требования к ПО. Подробнее о приемах по спецификациям к требованиям — в следующих главах:

- в главе 9 — о документировании бизнес-правил;
- в главе 10 — об использовании шаблона спецификации требований к ПО, о присвоении уникальных идентификаторов всем требованиям;
- в главе 12 — об указании атрибутов качества.

Использование шаблона спецификации требований к ПО. Создайте стандартный шаблон для документирования требований к ПО в вашей организации. Шаблон предоставляет согласованную структуру, позволяющую фиксировать описания нужной функциональности, а также прочую информацию, касающуюся требований. Вместо того чтобы изобретать новый шаблон, модифицируйте один из существующих в соответствии со спецификой проекта. Многие компании начинают с использования шаблона спецификации требований к ПО, описанного в стандарте IEEE 830-1998 (IEEE, 1998b); подробнее о работе с ним — в главе 10. Если ваша компания занимается разными проектами, например проектирует новое крупное приложение и параллельно дорабатывает версии старых программ, создайте соответствующие шаблоны для всех

типов проектов. Шаблоны и процессы должны быть масштабируемыми.

Определение источников требований. Чтобы гарантировать, что все заинтересованные лица понимают, почему то или иное требование зафиксировано в спецификации требований к ПО, и упростить последующее прояснение требований, выявите источники всех требований. Это может быть вариант использования или другая информация от пользователей, системное требование высокого уровня, бизнес-правило или иной внешний фактор. Указав всех лиц, заинтересованных в каждом требовании, вы будете знать, к кому обратиться при поступлении запроса на изменение. Источники требований устанавливают на основе связей или определяют для этой цели атрибут требования, Подробнее об атрибутах требований — в главе **18.**

Присвоение уникальных идентификаторов всем требованиям,

Выработайте соглашение о присвоении уникальных идентификаторов требованиям, зафиксированным в спецификации требований к ПО Соглашение должно быть устойчивым к дополнению, удалению элементов и изменениям, вносимым в требования. Присвоение идентификаторов позволяет отслеживать требования и фиксировать вносимые изменения.

Указание атрибутов качества. Выявляя качественные характеристики, удовлетворяющие потребности клиента, не ограничивайтесь только обсуждением функциональности. Выясните ожидаемые производительность, эффективность, надежность, удобство использования и др. Информация от клиентов об относительной важности тех или иных качественных характеристиках позволит разработчику принято правильные решения, касающиеся архитектуры приложения.

Документирование бизнес-правил. К бизнес-правилам относятся корпоративные политики, правительственные распоряжения и алгоритмы вычислений. Ведите список бизнес-правил отдельно от спецификации требований к ПО, поскольку правила обычно существуют вне рамок конкретного проекта. Для выполнения некоторых приходится создавать реализующие их функциональные требования, и поэтому необходимо определить связь между этими требованиями и соответствующими правилами.

Проверка требований

Проверка гарантирует, что все положения требований корректны, отражают желаемые качественные характеристики и удовлетворяют потребностям клиента. Может оказаться, что требования, в спецификации требований к ПО выглядевшие превосходно, при реализации чреваты проблемами. В большинстве случаев удается выявить двусмысленности и неопределенности, написав для требований сценарии тестирования. Если требования должны стать надежной основой для проектирования и итоговой проверки системы посредством системного тестирования или тестирования на приемлемость для пользователей, эти проблемы необходимо устранить. Подробнее о проверке требований — в главе 15.

Изучение документов с требованиями. Официальная проверка документирования требований — один из наиболее ценных способов, проверки качества ПО. Соберите небольшую команду, члены которой представляют различные направления (например, аналитик, клиент, разработчик и специалист по тестированию), и тщательно изучите спецификацию требований к ПО, модель анализа и соответствующую информацию на предмет недостатков. Также полезно провести в ходе формулирования требований их неофициальный предварительный просмотр. И хотя реализовать это на практике непросто, данный прием — один из самых ценных, так что начинайте внедрять проверку требований в вашей организации прямо сейчас.

Тестирование требований. На основе пользовательских требований создайте сценарии функционального тестирования и задокументируйте ожидаемое поведение продукта в конкретных условиях. Совместно с клиентами изучите сценарии тестирования и убедитесь, что они отражают нужное поведение системы. Проследите связь сценариев тестирования с функциональными требованиями и удостоверьтесь, что ни одно требование не пропущено и что для всех требований есть соответствующие сценарии тестирования. Запустите сценарии, чтобы удостовериться в правильности моделей анализа и прототипов.

Определение критериев приемлемости. Предложите пользователям описать, как они собираются определять соответствие продукта их потребностям и его пригодность к работе. Тесты на приемлемость следует основывать на сценариях использования (Hsia, Kung и Sell, 1997).

Управление требованиями

Начальные требования неизбежно корректируются в процессе работы клиентами, менеджерами, специалистами по маркетингу, разработчиками и другими лицами. Для эффективного управления требованиями необходим процесс, позволяющий предлагать изменения и оценивать их возможную стоимость и влияние на проект. Решения о внесении изменений принимает *совет по управлению изменениями* (change control board, CCB), в который входят все заинтересованные лица. Контроль за выполнением требований на разных стадиях разработки и тестирования системы позволяет лучше понять состояние проекта в целом.

Для эффективного управления проектом жизненно необходимы выверенные способы управления конфигурацией. Для управления требованиями годятся те же утилиты для контроля версий, которые вы используете для управления базой кода. Подробнее о способах управления требованиями — в следующих главах:

- в главе 18 — о создании базовой версии и управлении версиями требований, о контроле за состоянием всех требований;
- в главе 19 — об определении процесса управления изменениями, создании совета управления изменениями, оценке изменяемости требований, анализе влияния изменений требований;
- в главе 20 — о создании матрицы связей требований;
- в главе 21 — об использовании средств управления требованиями.

Определение процесса управления изменениями. Определить процесс представления, анализа и утверждения или отклонения изменений. Применяйте его для управления всеми предлагаемыми изменениями. В контексте процесса управления изменениями полезно использовать коммерческие средства отслеживания недостатков.

Создание совета по управлению изменениями. Из представителей заинтересованных в проекте лиц организуйте совет по управлению изменениями, который будет получать информацию о предполагаемых изменениях требований, оценивать ее, решать, какие изменения принять, а какие отклонить, и определять, в какой версии продукта будет внедрена та или иная функция.

Анализ влияния изменений требований. Анализ влияния изменений помогает совету по управлению изменениями принимать обоснованные решения. Оцените, как каждое предлагаемое изменение требований повлияет на проект. На основе матрицы связей выявите другие требования, элементы архитектуры, исходный код и сценарии тестирования, которые, возможно, придется изменить. Определите, что необходимо для реализации изменений, и оцените затраты на реализацию.

Создание базовой версии и управление версиями требований.

Базовая версия содержит требования, утвержденные для реализации в конкретной версии продукта. После определения базовых требований изменения можно вносить только в соответствии с процессом управления изменениями. Присвойте всем версиям спецификации требование уникальные идентификаторы, чтобы избежать путаницы между черновыми вариантами и базовыми версиями, а также между предыдущей и текущей версиями требований. Более надежное решение — управлять версиями документов с требованиями при помощи соответствующих средств управления конфигурацией.

Ведение журнала изменений требований. Фиксируйте даты изменения спецификаций требований, сами коррективы, их причины, а также лиц, вносивших изменения. Автоматизировать эти задачи позволяет утилита управления версиями или коммерческая утилита управления требованиями.

Контроль за состоянием всех требований. Создайте БД, включающую по одной записи для каждого дискретного функционального требования. Занесите в БД ключевые атрибуты каждого требования, включая его состояние (например «предложено», «одобрено», «реализовано» или «проверено»), чтобы в любой момент вы могли узнать количество требований в каждом состоянии,

Оценка изменяемости требований. Еженедельно фиксируйте количество требований,

внесенных в базовую версию, а также число предложенных и одобренных изменений (добавлений, модификаций и удалений). Если требования формируются не самим клиентом, а от его лица, может оказаться, что проблема понята плохо, границы проекта определены нечетко, бизнес стремительно меняется, при сборе информации многие требования были упущены или внутрикорпоративные политики меняются в худшую сторону.

Использование средств управления требованиями. Коммерческие утилиты управления требованиями позволяют хранить различные типы требований в БД. Для каждого требования можно определить атрибуты, отслеживать его состояние, а также выявить связи между требованиями и другими рабочими продуктами. Данный прием поможет вам автоматизировать прочие задачи по управлению требованиями, описанные ниже.

Создание матрицы связей требований. Создайте таблицу, сопоставляющую все функциональные требования с элементами архитектуры и кода, которые реализуют данное требование, и с тестами, проверяющими его. Матрица связей требований позволяет также сопоставить функциональные требования с требованиями более высоких уровней, на основе которых они созданы, и с другими родственными требованиями. Заполняйте эту таблицу в ходе, а не в конце работы над проектом.

Управление проектом

Способы управления проектом ПО тесно связаны с работой над требованиями к нему. Планируйте ресурсы, графики и обязательства по проекту на основании требований, которые собираетесь реализовать. Изменения требований влияют на планы реализации, поэтому в планах следует предусмотреть возможность частичного изменений требований и расширения границ проекта. Подробнее о способах управления проектом, касающихся создания требований — в следующих главах:

- в главе 17 — о планах, связанных с проектом, о их реализации на основе требований;
- в главе 18 — о контроле объема работ по созданию требований;
- в главе 23 — о документировании и управлении рисками, связанными с требований.

Выбор цикла разработки ПО. Вашей компании следует определить, несколько жизненных циклов разработки для проектов различного типа и различных степеней неопределенности требований (McConnel, 1996). Каждый менеджер проекта должен выбрать и использовать цикл, оптимальным образом подходящий для его проекта. Включите в цикл операции по созданию требований. Если на ранних этапах работы над проектом требования или границы проекта определены нечетко, разрабатывайте продукт постепенно (небольшими этапами), начиная с наиболее понятных требований и устойчивых элементов архитектуры. По возможности реализуйте наборы функций, чтобы периодически выпускать промежуточные версии продукта и как можно раньше предоставлять клиенту работоспособные образцы приложение (Gilb. 1988; Cockburn, 2002).

Планы реализации проекта должны быть основаны на требованиях. Разрабатывайте планы и графики работы над проектом постепенно, по мере прояснения границ и подробных требований. Начните с оценки затрат, необходимых на реализацию функциональных требований, определенных на основе первоначальных образа и границ продукта. Графики и оценка затрат, построенные на основе нечетких требований, окажутся крайне неточными, однако по мере детализации требований их следует уточнить.

Пересмотр обязательств по проекту при изменении требований.

Добавляя в проект новые требования, оцените, удается ли соблюдать обязательства, касающиеся графика и требований к качеству, при доступном объеме ресурсов. Если нет, обсудите реалии проекта с менеджерами и согласуйте новые, достижимые обязательства (Humphrey, 1997; Fisher, Ury, и Patton, 1991; Wiegers, 2002). Если переговоры не увенчаются успехом, сообщите менеджерам и клиентам о их результатах, чтобы нарушение планов в реализации проекта не стало для них неожиданностью.

Документирование и управление рисками, связанными с требованиями. Одна из составляющих управления рисками проекта — выявление и документирование рисков, связанных с требованиями. Уменьшайте или предотвращайте их посредством мозговых штурмов, реализуйте

корректирующие действия и отслеживайте их эффективность.

Контроль объема работ по созданию требований. Фиксируйте усилия, прилагаемые вашей командой на разработку требований и управление проектом. Эти данные позволят оценить соответствие планам и эффективнее спланировать необходимые ресурсы для будущих проектов. Также отслеживайте, как ваши действия по регламентации требований влияют на проект в целом. Это позволит оценить отдачу от этой работы,

Извлечение уроков из полученного опыта. Для этого в организации следует провести *ретроспективу* проектов, называемую также *изучением законченных проектов* (Robertson и Robertson, 1999; Kerth, 2001; Wiegers и Rothman, 2001). Ознакомление с опытом в области проблем и способов создания требований, накопленным в ходе работы над предыдущими проектами, помогает менеджерам и аналитикам требований более эффективно работать в будущем.

Начинаем применять новые приемы

В табл. 3-2 описанные выше приемы создания требований сгруппированы по их относительному влиянию на большинство проектов, а также по относительной сложности реализации. И хотя полезны все способы, начать можно с простейших, наиболее влияющих на успех проекта и относительно простых в реализации.

Таблица 3-2. Реализация приемов формулирования требований

Влияние	**Сложность**		
	Высокая	***Средняя***	***Низкая***
Сильное	• Определите процесс формулирования требований. • Планируйте на основании требований. • Пересматривайте обязательства.	• Определите варианты использования • Укажите атрибуты качества. • Определите приоритеты требований. • Используйте шаблон спецификации требований к ПО. • Определите процесс управления изменениями. • Создайте совет управления изменениями. • Изучите документы с требованиями. • Распределите требования по подсистемам. • Задокументируйте бизнес-правила.	• Обучите разработчиков основам предметной области. • Определите образ и границы проекта. • Определите классы пользователей. • Нарисуйте контекстную диаграмму. • Определите источники требований. • Определите базовую версию и управляйте версиями требований.

Среднее	• Ознакомьте представителей пользователей и менеджеров с требованиями. • Моделируйте требования. • Управляйте рисками, связанными с требованиями. • Используйте утилиту управления требованиями. • Создайте матрицу связей требований. • Проводите семинары для уточнения требований.	• Обучите аналитиков • Требований. • Выберите ярых сторонников продукта. • Создайте фокус-группы. • Создайте прототипы. • Определите критерии приемлемости. • Анализируйте влияние изменений. • Выберите соответствующий цикл.	• Проанализируйте осуществимость. • Создайте словарь бизнес-терминов. • Создайте словарь данных. • Наблюдайте за пользователями на рабочих местах. • Определите системные события и реакцию на них. • Задайте каждому требованию уникальный.
Слабое	• Используйте требования повторно. • Воспользуйтесь технологией развертывания функций качества. • Оцените изменяемость требований.	• Ведите журнал изменений. • Отслеживайте объем работ по реализации требований.	• Изучите отчеты о проблемах.

Не пытайтесь применять в своем следующем проекте сразу все эти приемы. Скорее, их стоит рассматривать как новые компоненты вашего инструментария для работы над требованиями. Некоторые способы, например касающиеся управления изменениями, можно начать использовать независимо от того, на какой стадии разработки находится ваш проект. Приемы выявления требований наиболее полезны в начале работы над очередным проектом или при повторении. Прочие способы могут оказаться неприемлемыми из-за ограничений текущего проекта, корпоративной культуры или объема доступных ресурсов. В главе 22 рассказывается, как оценить способы создания требований, используемые вами в настоящий момент. Кроме того, описанные выше приемы помогут вам разработать план реализации сбора и документирования требований.

Процесс создания требований

Не ждите, что все действия по выявлению, анализу, спецификации и проверке требований удастся выполнить последовательно и за один проход. На практике эти действия выполняются попеременно, поэтапно и повторяются (рис. 3-1). Работая с клиентами в качестве аналитика, вы будете задавать вопросы, выслушивать ответы и наблюдать за действиями клиентов (выявление требований). Далее вы обработаете полученную информацию, классифицируете по различным категориям и соотнесете потребности клиентов с возможными требованиями к ПО (анализ). Затем вы оформите информацию от клиентов и выработанные требования в виде письменных документов и диаграмм (спецификация), предложите представителям пользователей подтвердить,

что написанный вами текст точен и полон, и попросите их исправить возможные ошибки (проверка). Этот итерационный процесс и есть процедура создания требований.

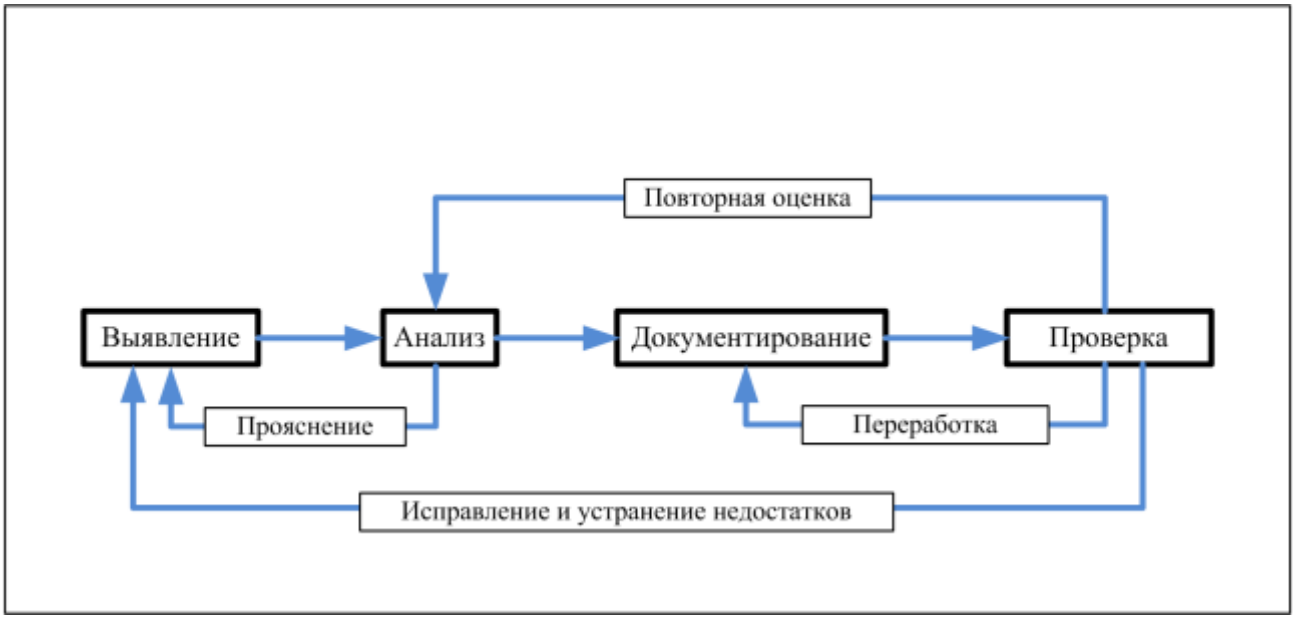


Рис. 3-1. Итеративный процесс формулирования требований

Из-за разнообразия проектов по разработке ПО и организационных культур единого, шаблонного подхода к созданию требований не существует. На рис. 3-2 показана схема создания требований, которая с разумными исправлениями подойдет для большинства проектов. Как правило, действия выполняются в основном по порядку, однако сам процесс не является строго последовательным. Первые семь действий обычно однократно выполняются на ранних стадиях работы нал проектом (тем не менее, команде разработчиков придется периодически изменять приоритеты). Остальные необходимы для каждого очередного выпуска или этапа работы над проектом.

Оценив доступные вам способы общения с представителями пользователей, выберите подходящие приемы выявления требований (круглые столы, исследования, опросы и т.д.) и спланируйте время и ресурсы, необходимые для сбора информации (этап 5 на рис. 3-2). Многие системы создаются поэтапно, и поэтому любой команде, работающей над проектом, необходимо определить приоритеты вариантов использования и других пользовательских требований (этап 7). Расставив приоритеты, вы решите, на каком этапе следует реализовать те или иные варианты использования. В случае новых систем или значительных усовершенствований можно на этапе 14 определить или уточнить архитектуру, а на этапе 15 распределить функциональные требования по конкретным подсистемам. Этапы 12 и 17 — это операции по контролю качества, в результате которых вам, возможно, придется вернуться, чтобы исправить ошибки, улучшить модели анализа или выявить упущенные ранее требования. Прототипы, создаваемые на этапе 13, зачастую выявляют необходимость усовершенствовать и модифицировать определенные ранее требования. Завершив для какой-либо части требований этап 17 можно приступать к реализации соответствующей части системы. Повторите этапы 8—17 для следующих наборов вариантов использования, которые могут войти в более позднюю версию продукта.

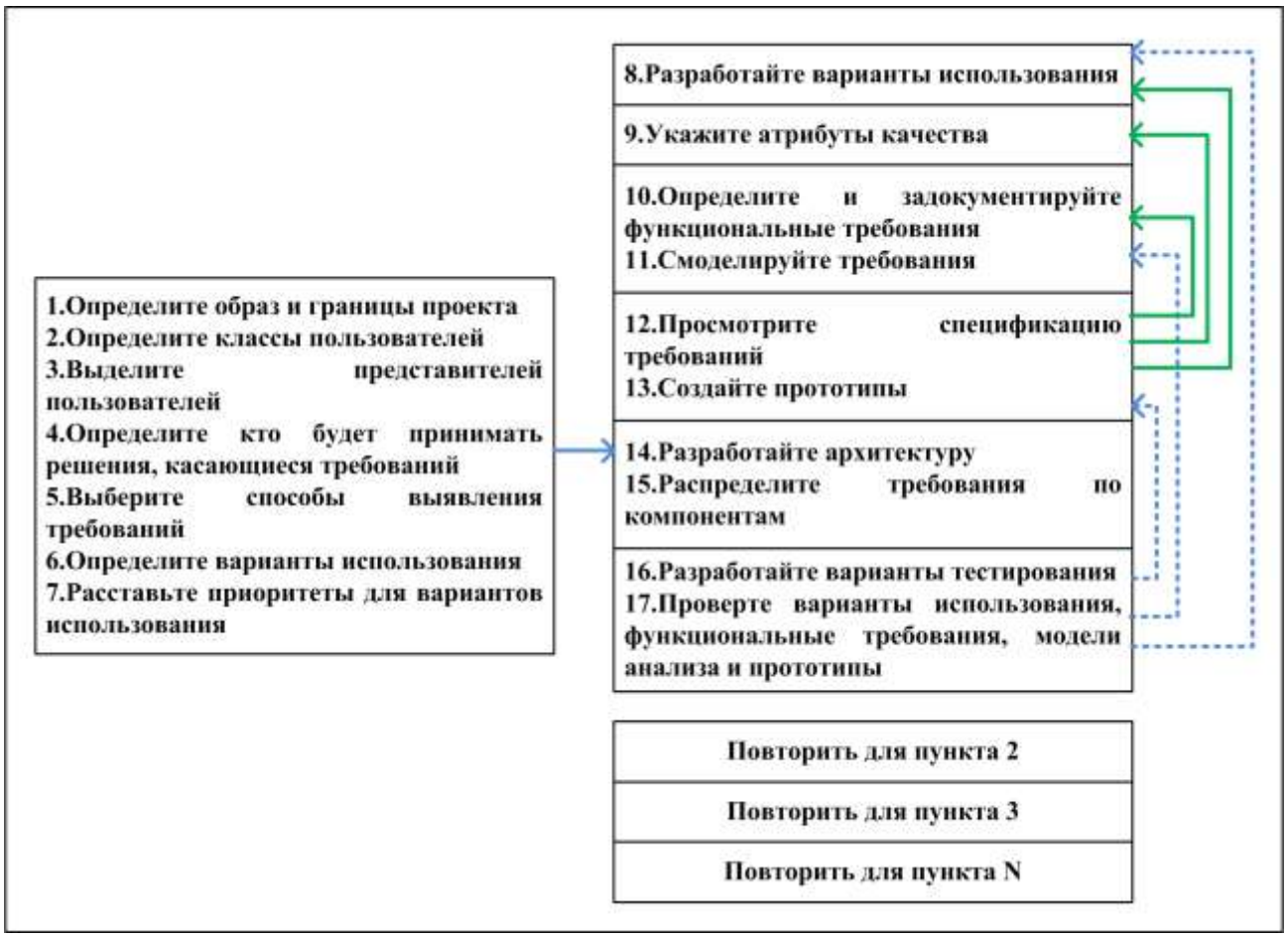


Рис. 3-2. Поэтапный процесс создания требований

Что теперь?

- Вернитесь к проблемам с требованиями, которые вы определили, выполняя задания раздела «Что теперь?» главы 1. Выберите описанные в этой главе приемы, которые помогут вам решить все эти проблемы. Или же воспользуйтесь руководством по выявлению и устранению проблем из приложения В. Сгруппируйте приемы по их влиянию на вашу организацию (сильное, среднее и слабое). Определите, какие препятствия в организации или культуре способны помешать вам воспользоваться тем или иным приемом. Кто поможет вам устранить эти препятствия?
- Оцените приемы, которые считаете наиболее ценными. Позволят ли они уменьшить число некорректных требований, выявляемых на поздних стадиях работы над проектом, уменьшить объем ненужных доработок, точнее следовать графику проекта или предоставят какие-то другие преимущества?
- Перечислите все приемы формулирования требований, определенные на первом этапе. Укажите, насколько члены вашей команды овладели на сегодняшний момент тем или иным приемом: квалифицированные специалисты, опытные специалисты, новички или не знакомы с данным приемом. Если квалификация членов команды ниже, чем по крайней мере опытные специалисты, попросите одного из них изучить данный прием и поделиться полученными знаниями с остальными членами команды

Глава 4 Аналитик требований

Среди участников любого проекта по разработке ПО обязательно есть человек, явно или неявно выполняющий роль аналитика требований. Крупные фирмы для решения подобных задач привлекают специалистов такого профиля — бизнес-аналитиков. Их также называют ***системными аналитиками, инженерами по требованиям, менеджерами по требованиям*** и просто ***аналитиками.*** В организации, разрабатывающей продукты, функции аналитика зачастую выполняет менеджер по продукту или специалист отдела маркетинга. Задача аналитика — отразить мнения заинтересованных сторон и лиц в спецификации требований и передать информацию другим заинтересованным в проекте лицам. Аналитик помогает участникам проекта прояснить, действительно ли пожелания, которые они высказывают вслух, — это то, что им на самом деле нужно. Аналитик обучает, задает вопросы, слушает, организует и учится. Это сложная работа.

В этой главе я познакомлю вас с функциями аналитика требований, требованиями, которые предъявляются к его знаниям и опыту, а также расскажу, как воспитать аналитика в своей организации (Wiegers, 2000). Пример должностных обязанностей аналитика требований опубликован на: *http://www.processimpact.com/goodies.shtml.*

Роль аналитика требований

Аналитик требований — это основное лицо, отвечающее за сбор, анализ, документирование и проверку требование к проекту. Это основной коммуникативный канал между группой клиентов и командой разработчиков (рис. 4-1), хотя, конечно, не единственный: есть и другие, Аналитик отвечает за сбор и распространение информации о продукте, а менеджер проекта — за обмен информацией о проекте.

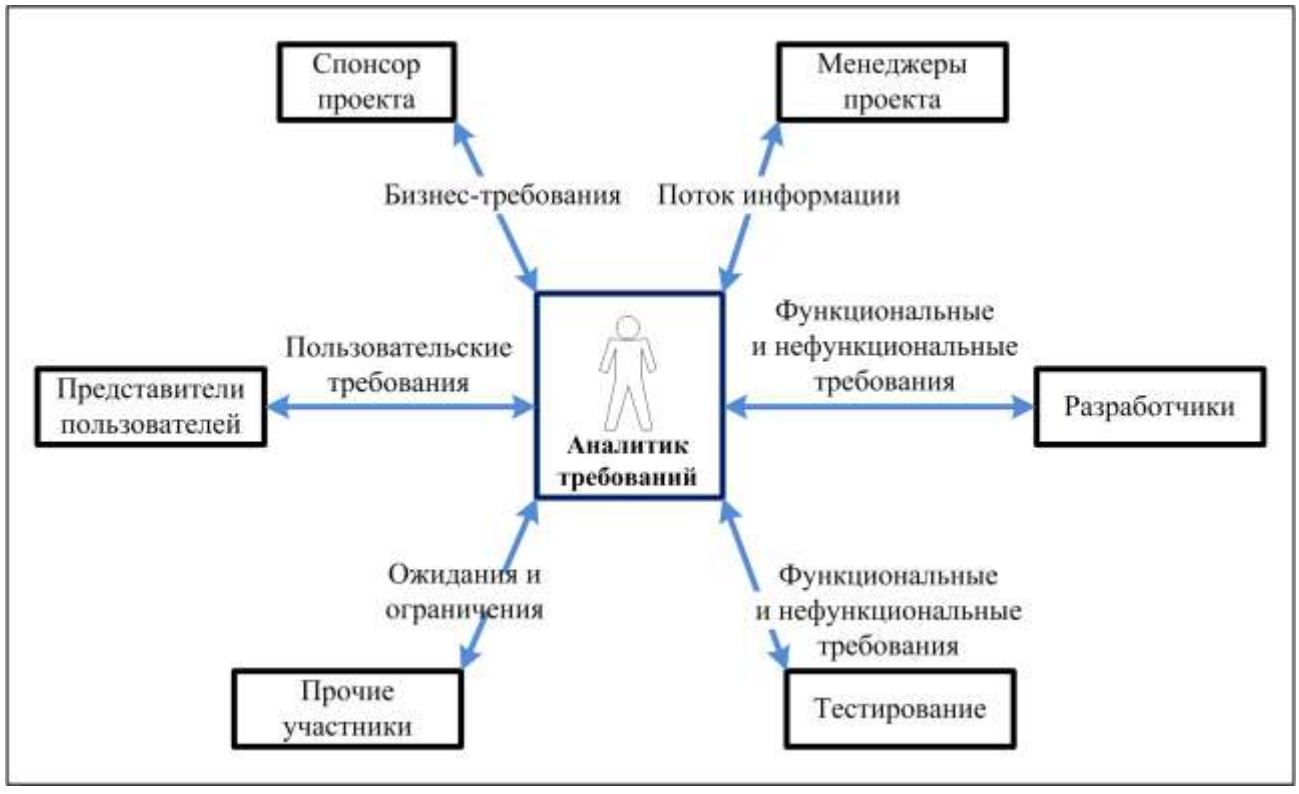


Рис. 4-1. Обязанности аналитика требований: наведение коммуникативных мостов между различными группами клиентов и разработчиков проекта

Аналитик требований — это одна из ролей участников проекта, а не обязательно название должности. На эту роль можно назначить одного или нескольких специалистов. Кроме того, функции аналитика могут выполнять другие члены команды параллельно со своими обязанностями, например менеджер проекта, менеджер по продукту, ***профильный специалист*** (subject matter

expert), разработчик и даже пользователь. В любом случае аналитик должен обладать всеми навыками, знаниями и личными качествами, необходимыми для эффективной работы.

> **Ловушка**
>
> Не думайте, что любой талантливый разработчик или опытный пользователь автоматически, без обучения, чтения литературы и тренировок, станет профессиональным аналитиком требований. Все эти роли требуют разных навыков, знаний и личных качеств.

От таланта аналитика зависит успех проекта. Один из клиентов, которых я консультировал, пришел к выводу, что спецификации с требованиями, созданные опытными аналитиками, удается изучать вдвое быстрее, чем созданные новичками, поскольку в первых меньше недостатков. В модели Cocomo II широко применяемой для оценки проектов, указано, что опыт и способности аналитика требований сильно влияют на материальные и трудовые затраты, связанные с реализацией проекта (Boemn et all., 2000). Привлекая опытных специалистов можно на треть снизить связанные с проектом трудозатраты по сравнению с аналогичными проектами, где заняты неопытные аналитики. Способности аналитика оказывают даже большее влияние, чем его опыт: трудозатраты удается сократить вдвое.

Задачи аналитика.

Аналитик — это посредник в общении, проясняющий смутные представления пользователей и обращающий их в четкие спецификации, которыми руководствуется команда разработчиков продукта. Задача аналитика — прежде всего выяснить, для чего нужна пользователям новая система, и затем определить функциональные и качественные требования, на основе которых менеджеры проекта смогут оценить разработчики — спроектировать и создать, а специалисты по тестированию — проверить продукт. Далее описаны некоторые стандартные обязанности аналитика.

Определить бизнес-требования. Ваша работа в качестве аналитика начинается, когда вы помогаете спонсору, менеджеру продукта или менеджеру по маркетингу определить бизнес-требования к проекту. Возможно, первое, что следует спросить: «Зачем мы начинаем этот проект?» Бизнес-требования включают бизнес-цели организации и представление о внешнем виде и функциональности системы. Можно разработать шаблон документа об образе и границах (см. главу 5) и, расспросив людей, имеющих представление о внешнем виде системы, получить у них необходимую информацию.

Определить заинтересованных лиц и классы пользователей. Документ об образе и границах поможет вам выявить важные классы пользователей и прочих заинтересованных в продукте лиц. Затем совместно с заказчиками следует выбрать соответствующих представителей каждого класса, заручиться их поддержкой и согласовать обязанности. Пользователи могут сомневаться, стоит ли участвовать в создании требований, пока не будут точно знать, чего именно вы от них хотите. Запишите, какого именно сотрудничества вы хотите, и ознакомьте их с этим документом. В главе 6 описаны некоторые виды помощи, которая вам может потребоваться от клиентов.

Выявить требования. Требования к программному продукту не лежат на виду и не ждут, когда какой-нибудь аналитик придет и соберет их, Профессиональный аналитик помогает пользователям четко обрисовать функции системы, необходимые им для достижения бизнес-целей. (Подробнее об этом — в главах 7 и 3). Для этого у него в арсенале припасено множество способов сбора информации:

- интервью;
- семинары;
- анализ документов;
- опросы;

- посещение рабочих мест клиентов;
- анализ бизнес-процессов;
- анализ документооборота и задач;
- списки событий;
- анализ конкурирующих продуктов;
- исследование существующих систем;
- ретроспективы развития предыдущего проекта.

Обычно пользователи уделяют особое внимание функциональным требованиям к системе, поэтому на дискуссии следует обсудить качественные атрибуты, задачи производительности, бизнес-правила, внешние интерфейсы и ограничения. Нормально, если аналитик спорит с пользователями, однако не стоит навязывать им свое мнение. Некоторые пользовательские требования могут показаться абсурдными, но если клиент утверждает, что они верны, лучше уступить ему.

Анализировать требования. Ищите производные требования, логически проистекающие из запросов клиентов, а также невысказанные ожидания, которые, как считают клиенты, и так будут реализованы. Сразу же проясните неясные и неубедительные слова, порождающие двусмысленность (примеры см. в главе 10). Выявите конфликтующие требования и области, требующие подробной детализации. Определите функциональные требования с такой степенью подробности, которая необходима разработчикам. От проекта к проекту степень подробности различается. Для Web-узла, постепенно создаваемого небольшой и дружной командой, достаточно документации с ограниченным перечнем требований. Напротив, для разработки сложной встроенной системы, которую будет строить внешний поставщик, потребуется точная и подробная спецификация требований к ПО.

Создавать спецификации с требованиями. В результате формулировки требований формируется коллективный взгляд на систему. Аналитик отвечает за хорошую организацию спецификаций, в которых этот взгляд четко отражен. В результате применения стандартных шаблонов для вариантов использования продукта и спецификации требований к ПО создание документации ускоряется, поскольку у аналитика перед глазами постоянно находится перечень тем, которые нужно обсудить с представителями пользователей. Подробнее об области применения — в главе 8, написании функциональных требований — в главе 10 и документировании качественных атрибутов ПО — в главе 12.

Моделировать требования. Аналитик должен определить, полезно ли представлять требования нетекстовыми средствами, например с помощью разнообразных моделей графического анализа (подробнее — в главе 11), таблиц, математических уравнений, раскадровок и прототипов (подробнее — в главе 13). Модели анализа отражают информацию на более высоком уровне абстракции, чем подробный текст. Для максимальной прозрачности и эффективного общения рисуйте модели анализа, используя правила стандартного языка моделирования.

Управлять проверкой требований. Аналитик должен гарантировать, что формулировки требований отвечают всем характеристикам, перечисленным в главе I. и что система на основе этих требований устроит пользователей. Аналитики участвуют в обзорах документов с требованиями. Им также следует изучить архитектуру, код и варианты тестирования, спроектированные на основе спецификаций требований, и убедиться, что требования интерпретированы правильно.

Обеспечить расстановку приоритетов требований. Аналитик обеспечивает общение и взаимодействие различных классов пользователей с разработчиками, что позволяет расставить приоритеты. Возможно, вам будет полезна электронная таблица для назначения приоритетов требованиям, обсуждаемая в главе 14.

Управлять требованиями. Аналитик требований вовлечен во все этапы разработки ПО, его задача — помочь создать, обсудить и осуществить план управления требованиями к проекту. Создав документацию, аналитик управляет требованиями и проверяет, как они реализуются в продукте. Хранение требований с помощью специальных коммерческих утилит упрощает управление ими. Подробнее о средствах управления требованиями — в главе 21.

Управление требованиями подразумевает контроль за состоянием отдельных функциональных требований по мере степени готовности продукта. Расспрашивая коллег, аналитик

собирает информацию о связях требований, сопоставляет их с прочими элементами системы, Аналитик — ключевая фигура в управлении изменениями базовых требований, для этого он применяет средства контроля изменений.

Навыки, необходимые аналитику

Не думайте, что без достаточной подготовки, тренировок и опыта человек сможет стать аналитиком. Ему не только не удастся хорошо выполнять работу, он просто разочаруется в ней. Аналитик должен уметь применять разные средства сбора информации и представлять эту информацию различными способами на нормальном и понятном языке. Профессионал в этой области обладает одновременно развитыми коммуникационными навыками, знанием психологии межличностного общения, техническими знаниями, знаниями предметной области бизнеса и личными качествами, подходящими для этой работы (Ferdinandi, 2002). Основные факторы успеха — терпение и искреннее желание работать с людьми. Не менее важны и навыки, описанные далее.

Умение слушать. Чтобы стать специалистом, научитесь эффективно слушать. Активное слушание подразумевает устранение помех, сохранение вежливой позы и зрительный контакт, а также повторение основных моментов для закрепления их понимания. Вам нужно моментально схватывать, что говорят люди, и уметь читать между строк, что бы обнаружить вещи, о которых они стесняются говорить. Изучите, как ваши коллеги предпочитают общаться и избегайте налагать собственный фильтр понимания на высказывания клиентов. Ищите допущения, которые подчеркивали бы мысли других или их интерпретацию.

Умение опрашивать и задавать вопросы. Большая часть информации о требованиях извлекается в ходе бесед с людьми, и поэтому аналитик должен уметь общаться с разными людьми и группами — только так ему удастся выявить их потребности. Возможно, работать со старшими менеджерами или чрезмерно самоуверенными или агрессивными людьми будет трудно. Для выяснения существенных требований пользователей необходимо задавать правильные вопросы. Например, пользователи обычно делают акцент на ожидаемом поведении системы. Тем не менее значительная часть кода будет обрабатывать исключения, поэтому вам следует определить возможные условия ошибок и реакцию системы на них. По мере приобретения опыта вы научитесь задавать вопросы, которые раскрывают и проясняют неопределенности, расхождения во мнениях, предположения и невысказанные ожидания (Gause и Weinberg, 1989).

Навыки анализа. Эффективный аналитик способен думать на нескольких уровнях абстракции. Иногда требуется перейти от сведений высшего порядка к подробностям. В некоторых случаях на основе потребности одного из пользователей сформулировать набор требований, которые удовлетворят большинство пользователей данного класса. Критически оценивайте информацию, полученную на основе разных источников, чтобы урегулировать конфликты, отделить мимолетные желания пользователей от их реальных потребностей и отличать варианты решений от требований,

Навыки создания комфортных условий общения. Умение организовать дружескую атмосферу на семинарах для уточнения требований — один из необходимых навыков аналитика (Gottesdiener, 2002). Нейтральный наблюдатель, имеющий опыт опроса, наблюдения и создания комфортных условий общения, создаст доверительные отношения в группе и уменьшит напряженность в отношениях между бизнесменами и ИТ-сотрудниками. В главе 7 даны некоторые рекомендации по организации семинаров.

Умение наблюдать. Внимательный аналитик запомнит высказанные мимоходом комментарии, которые могут оказаться важными. Наблюдая за тем, как пользователь выполняет свои обязанности или работает с имеющимся приложением, опытный аналитик выявит моменты, о которых пользователь даже не упомянул. Наблюдательность иногда помогает направить дискуссию в новое русло, чтобы выявить дополнительные требования, о которых никто ничего не сказал.

Навыки написания документации. Основной итог процесса создания требований — письменная спецификация с информацией для клиентов, отдела маркетинга и технического персонала. Аналитик должен отлично владеть языком и ясно выражать сложные идеи.

Я знаком с одной организацией, где аналитиками назначили нескольких специалистов, для которых английский был вторым языком. Написать отличные требования трудно даже на родном

языке. Еще сложнее, когда попутно приходится разбираться в нюансах оборотов речи, двусмысленности слов и местных идиомах. Аналитик же должен уметь читать критично и эффективно, поскольку ему приходится просматривать множество материалов и необходимо быстро уяснять их суть.

Организационные навыки. Аналитик имеет дело с большим объемом беспорядочной информации, собранной на первом этапе. Чтобы справиться с данными и выстроить согласованное целое, вам потребуются исключительные организационные навыки, а также терпение и упорство для вычленения основных идей из хаоса.

Навыки моделирования. Аналитик должен уметь работать с разнообразными средствами, начиная с древних блок-схем и структурированных моделей анализа (диаграммы потоков информации, диаграммы «сущность - связь» и т.д.) и заканчивая современным языком UML (Unified Modeling Language, унифицированный язык моделирования). Некоторые из этих средств полезны при общении с пользователями другие — с разработчиками. Аналитику следует объяснить другим участникам проекта ценность использования этих методов и то, как работать с их данными. В главе 11 дан обзор некоторых типов моделей анализа.

Навыки межличностного общения. Аналитик должен уметь организовать людей с разными интересами для совместной работы, и уверенно чувствовать себя в разговорах с сотрудниками, занимающими разные должности в организации. Подумайте, как сложно иметь дело с сотрудниками из виртуальных групп, различающихся по географическому, временному, культурному или языковому признаку. Опытным аналитикам зачастую приходится наставлять своих коллег-новичков и объяснять клиентам суть процессов создания требований и разработки ПО.

Творческий подход. Аналитик — не просто клерк, записывающий все высказывания клиентов. Лучшие аналитики изобретают требования (Robertson, 2002). Они предлагают инновационные функции продуктов, новые рыночные возможности и возможности для бизнеса и думают, как удивить и удовлетворить своих клиентов. Отличный аналитик творчески подходит к делу: рассказывая о системе, ему удается удивить клиента — тот даже не всегда подозревает, что такая функциональность возможна.

Знания, необходимые аналитику

В дополнение к описанным выше навыкам и личным качествам, аналитику требований требуются обширные знания, большая часть которых приобретается с опытом. Попробуйте понять современные способы создания требований и научитесь применять их на каждом этапе разработки ПО. Эффективный аналитик владеет широким спектром средств и знает когда их стоит использовать, а когда — нет.

Аналитик контролирует требования и управляет проектом на протяжении всего цикла его разработки. Аналитик, знающий тонкости управления проектом и рисками, а также понимающий методы обеспечения качества продукции, не допустит провала проекта из-за проблем, возникающих на этапе формулирования требований. В области коммерческой разработки ПО аналитику полезно знать концепции управления продуктом и основы позиционирования и разработки корпоративного ПО.

Становление аналитика

Знание предметной области — ценное качество для эффективного аналитика. Аналитику, разбирающемуся в бизнесе, легче общаться с клиентами и понимать их, ему удается выявить невысказанные предположения и неявные требования. Он может предложить варианты совершенствования бизнес-процессов, а также ценную функциональность, о которой пользователи даже не думали.

Великих аналитиков взращивают, а не обучают. Для работы аналитиком требуется множество личностных черт, а не знаний каких-либо технологий. Стандартного обучающего курса или описания обязанностей такого специалиста не существует. В аналитики приходят из разных профессий, и, скорее всего, у всех новичков есть пробелы в знаниях и навыках. Тому, кто

собирается заниматься этим делом, следует определить, какие именно из обсуждаемых в этой главе требований относятся к нему, и постараться активно восполнить пробел, чтобы первоклассно выполнить работу. Патриция Фердинанди (Patricia Ferdinandi) (2002) описывает требования к профессиональному уровню начинающего, опытного и ведущего аналитика требований в разных областях: практический опыт, разработка, управление проектами, средства и способы, качество и личностные характеристики. Аналитику-новичку пригодятся советы и наставления опытных коллег, выраженные, скажем, в форме обучения. Давайте посмотрим, как люди с разным профессиональным опытом становятся аналитиками.

Бывший пользователь

Во многих корпоративных отделах информационных технологий есть сотрудники, пришедшие в бизнес-аналитики из обычных пользователей. Они отлично понимают особенности бизнеса и рабочей среды и легко завоевывают доверие бывших коллег. Они знают язык пользователей, а также существующие системы и бизнес-процессы.

К сожалению, бывшие пользователи зачастую имеют весьма поверхностные знания о разработке ПО и взаимодействии с техническими специалистами. Если они не знакомы с методами моделирования анализа, то по привычке выражают всю информацию в текстовой форме. Пользователям, ставшим аналитиками требований, следует побольше выяснить о технической стороне разработки ПО, чтобы представлять информацию в наиболее подходящей для разных аудиторий форме. Некоторые бывшие пользователи считают, что они лучше, чем те, кто работает с ПО теперь, понимают, что на самом деле необходимо, и поэтому не обращаются к этим сотрудниками или неуважительно от носятся к информации, предоставленной теми, кто будет работать с новой системой. Недавние пользователи могут запутаться в текущих особенностях работы, да так, что не увидят возможности усовершенствовать бизнес-процессы посредством новой информационной сие темы. Еще одна опасность: бывший пользователь может запросто думать о требованиях только с точки зрения пользовательского интерфейса. Концентрация внимания на вариантах решения с самого начала создает ненужные ограничения архитектуры и зачастую не позволяет устранить реальную проблему.

От специалиста по медицинской технике к аналитику требований

У старшего менеджера отдела медицинских устройств в большой компании возникла проблема. «Два года назад я принял на работу в свой отдел трех специалистов по медицинскому оборудованию, чтобы те представляли потребности наших клиентов. Они проделали огромную работу, но уже не владеют особенностями современных медицинских технологий и поэтому не могут точно сказать, что же сегодня нужно клиентам. Какую работу мы можем предложить этим специалистам теперь?»

Бывшие специалисты по медицинскому оборудованию, работающие под началом этого менеджера могут стать кандидатами в аналитики требований. И хотя они не знают о новинках медицинской техники, они смогут общаться на одном языке с другими специалистами в этой области. За два года они усвоили принципы работы отдела. Возможно, им потребуется пройти обучение способам документации требований, тем не менее они накопили ценный опыт, который позволит им эффективно выполнять работу аналитиков.

Бывший разработчик

Менеджеры проекта, которым не хватает профессионального аналитика требований, зачастую ожидают, что его функции будет выполнять разработчик. К сожалению, навыки и личные качества, необходимые разработчику, отличаются от тех, что необходимы аналитику. Шаблонный

«компьютерщик» — не самый социально приятный из людей. Мало кто из разработчиков терпелив с пользователями, считая их необходимым злом, с которым нужно разобраться как можно быстрее, чтобы скорее вернуться к реальной работе — программированию. Конечно многие разработчики осознают важность процесса создания требований и высказывают желание работать аналитиками, когда потребуется. Те, кому нравится общаться с пользователями — хорошие кандидаты для специализации в области анализа требований.

Разработчику, ставшему аналитиком, вероятно, придется более подробно ознакомиться с предметной областью бизнеса. Однако разработчики легко переходят на техническое мышление и жаргон, концентрируясь вместо потребностей пользователей на программном продукте, который надо создать. Им будет полезно дополнительное обучение в области межличностных коммуникаций, которыми искусно владеют лучшие аналитики — умение эффективно слушать, вести переговоры и создавать комфортные условия общения.

Профильный специалист

Ralph Young (2001) рекомендует, чтобы аналитик требований был экспертом в предметной области или профильным специалистом, а не обычным пользователем: «Спецификация требований к ПО может определить, насколько разумны требования, как они расширяют существующую систему, как следует проектировать предполагаемую архитектуру и какое влияние они окажут на пользователей». Некоторые организации-разработчики ПО нанимают опытных пользователей их продуктов, обладающих большим опытом в предметной области, в качестве аналитиков или представителей пользователей.

Аналитик требований, будучи экспертом в предметной области, зачастую определяет требования к системе, которые соответствуют его личным предпочтениям, а не обоснованным потребностям различных классов пользователей. Иногда профильные специалисты увлекаются созданием универсальной, всеобъемлющей системы, когда на самом деле большую часть потребностей пользователей удовлетворит менее сложное решение. Зачастую лучше, чтобы аналитик требований из команды разработчиков взаимодействовал с профильным специалистом, который кроме того выбран в качестве ключевого представителя пользователей (сторонника продукта). Подробнее о роли сторонников продукта в разработке проекта — в главе 6.

Создание атмосферы тесного сотрудничества

Иногда в проектах по разработке ПО возникают напряженные отношения между аналитиками, разработчиками, пользователями, менеджерами и специалистами по маркетингу. Стороны не всегда доверяют мотивации других участников и не ценят потребности и ограничения других. Однако на самом деле у разработчиков и потребителей программного продукта общая цель. При создании корпоративных информационных систем все стороны работают на одну компанию и все они получают выгоду от улучшения результатов ее работы. В случае успеха коммерческого продукта налицо счастливые клиенты, получивший прибыль производитель и удовлетворенный разработчик. Аналитик требований — основное лицо, отвечающее за обеспечение тесного доброжелательного сотрудничества с представителями пользователей и прочими сторонами, заинтересованными в проекте. Эффективный аналитик принимает во внимание трудности, с которыми сталкиваются представители бизнеса и технических служб, и постоянно демонстрирует коллегам свое уважение. Аналитик добивается от участников проекта согласования требований, в результате чего выигрывают все:

- клиенты довольны продуктом;
- организация-разработчик получает прибыль;
- разработчики гордятся работой, выполненной в ходе сложного проекта.

Что теперь?

- Если вы — аналитик требований, сравните свои знания и навыки с описанными в этой главе. Если вы выявили какие-либо пробелы, выберите две темы и проработайте их читая, практикуясь, работая с наставником или посещая специализированные курсы.
- Выберите из этой книги один новый способ создания требований, о котором вы знаете недостаточно, и начните применять его со следующей недели — в буквальном смысле! Выберите два или три дополнительных способа и начните использовать их через месяц Другие оставьте на потом, чтобы изучить и применять их через пять-шесть месяцев. Определите, когда вы хотели бы применять каждый из новых способов, преимущества, которые, как вы считаете, он даст вам, а также какая помощь или дополнительная информация может потребоваться. Подумайте, чье сотрудничество понадобится вам, чтобы начать использовать новые способы. Выявите препятствия, которые могут помешать применять конкретный способ, и подумайте, кто в силах помочь вам преодолеть их.