Другие нефункциональные требования описывают среду, в которой работает система, например платформу, переносимость, совместимость и ограничения. Многие продукты также должны подчиняться определенным правилам, требованиям регулирующих органов или требовать сертификации. Такими могут быть требования по локализации для продуктов, в которых должны учитываться региональные стандарты, языки, законы, валюты, терминология, орфография и другие характеристики пользователей. Хотя такие требования определяются с использованием нефункциональной терминологии, бизнес-аналитик на их основе может определить много функций, чтобы система обладала всеми необходимыми свойствами и вела себя соответствующим образом в разных ситуациях.

Несмотря на описанные ограничения, в этой книге мы будем придерживаться термина «нефункциональные требования» за неимением подходящего всеобъемлющего альтернативного термина. Не надо волноваться о точности названия всей подобной информации — лучше позаботьтесь, чтобы она вошла в ваши действия по выявлению и анализу требований.

Можно создать продукт, обладающий всей требуемой функциональностью, но пользователи могут невзлюбить его за то, что тот не соответствует их (обычно невысказанным) ожиданиям по качеству.

*Характеристика* (feature) — это набор логически связанных функциональных требований, которые представляют ценность для пользователя и удовлетворяют бизнес-цели. Желательные 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 веб-браузера,

средства проверки орфографии, запись макрокоманды, автоматическое обновление определений вирусов в антивирусной программе. Характеристики могут охватывать множество пользовательских требований, и для каждого варианта необходимо, чтобы множество функциональных требований было реализовано для выполнения пользователем его задач. Рис. 1-2 иллюстрирует *дерево функций* (feature tree) — модель анализа, которая показывает, как функцию можно разложить на иерархию более мелких функций, которые

связаны с конкретными пользовательскими требованиями и ведут к определению наборов функциональных требований (Beatty и Chen, 2012). Чтобы вы лучше восприняли некоторые из различных видов требований, проанализируем проект по разработке следующей версии текстового редактора. Бизнес-требование может звучать так: «Увеличить продажи за пределами США на 25% за следующие полгода». В отделе маркетинга выяснили, что

в продуктах-конкурентах есть только англоязычные средства проверки орфографии, поэтому было принято решение включить возможность проверки орфографии на других языках. Соответствующие требования пользователей могут содержать задачи вроде такой: «Выберите язык проверки орфографии», «Найдите орфографические ошибки» или «Добавьте слово в словарь».

**Глава 11**

**Пишем идеальные требования**

*«Привет, Гаутам. Это звонит Рут из филиала в Остине. Мы получили по-*

*следнюю версию ПО веб-сайта для музыкального интернет-магазина. У меня*

*вопрос о функциональности предварительного прослушивания песен. Она ра-*

*ботает не так, как я себе представляла».*

*«Давай я посмотрю требования, которые ты присылала, — ответил*

*Гуатам. — А, вот они. В пользовательской истории говорится: «Как клиент*

*я хочу прослушать фрагменты песен, чтобы решить, какую приобрести».*

*В моих заметках говорится, что когда мы это обсуждали, ты сказала, что*

*длительность фрагментов песен должна быть 30 секунд и что должен ис-*

*пользоваться наш встроенный проигрыватель MP3-плейер, чтобы клиенту не*

*нужно было запускать другой проигрыватель. Разве все работает не так?»*

*«Ну да, это работает нормально, — ответила Рут, — есть ряд других про-*

*блем. Я могу запустить воспроизведение щелчком по кнопке запуска, но у меня*

*нет способа сделать паузу или остановить воспроизведение. Я вынуждена слу-*

*шать все 30 секунд фрагмента. Кроме того, все фрагменты содержат нача-*

*ло песни. У некоторых песен очень длинное вступление, поэтому по их началу*

*нельзя получить представление о самой песне. Фрагмент должен начинаться*

*где-то в середине этих песен, чтобы пользователи могли понять, что собой*

*представляет песня. Также фрагмент воспроизводится на полной громкости,*

*а затем резко прекращается. Если динамики пользователя настроены на вы-*

*сокий уровень громкости, воспроизведение может быть слишком громким.*

*Я думаю, что будет лучше постепенно увеличивать громкость в начале и*

*уменьшать в конце фрагмента».*

*Гуатам был немного разочарован. «Нужно было мне сказать все это рань-*

*ше, когда мы это обсуждали. Ты дала не очень много информации, поэтому мне*

*пришлось многое додумывать самостоятельно. Я могу все это сделать, но на*

*это потребуется еще несколько дней».*

Самое лучшее хранилище требований бесполезно, если оно не содержит высококачественной информации. Эта глава описывает необходимые характеристики требований и документов требований. Приведено много рекомендаций по написанию требований вместе с примерами неправильных требований и предложений по их исправлению. Эти рекомендации применимык требованиям, создаваемым в проектах независимо от применяемых в них методиках разработки. Авторы требований в каждом проекте должны самостоятельно определять необходимый уровень точности и детализации своих требований, но ничто не заменит ясной и четкой коммуникации.

**Характеристики превосходных требований**

Как отличить хорошие требования от проблемных? В этом разделе описываются несколько характеристик, которыми должны отличаться отдельные требования, после чего рассказывается о желательных характеристиках требований в целом (Davis, 2005; ISO/IEC/IEEE, 2011). Лучший способ сказать, обладают ли требования необходимыми атрибутами, — попросить нескольких заинтересованных лиц проверить их. Разные заинтересованные лица обнаружат разные недостатки. В главе 17 описывается использование контрольных списков для напоминания рецензентам о стандартных ошибках

в требованиях.

**Характеристики отдельных положений спецификации**

**требований**

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

**Полнота**

Каждое требование должно содержать всю информацию, необходимую читателю, чтобы понять его. В случае функциональных требований это означает предоставление всей информации, необходимой разработчику, чтобы реализовать их. Если вы понимаете, что данных определенного рода не хватает, используйте пометку «TBD» (to be determined — необходимо определить) на полях как стандартный флаг для выделения пробелов или зафиксируйте их в системе отслеживания дефектов, чтобы вернуться к ним позже. Восполните все пробелы в каждом фрагменте требований, прежде чем разработчики при-

ступят к разработке этой функции.

**Корректность**

Каждое требование должно точно описывать возможность, которая будет удовлетворять какую-то потребность заинтересованного лица и четко определять функциональность, которую надо построить. Для соблюдения корректности необходима связь с источниками требований, в качестве которых могут выступать пользователь, предоставивший начальное требование, высокоуровневое системное требование, вариант использования, бизнес-правило или другой документ. Низкоуровневое требование, которое конфликтует

со своим родительским требованием нельзя считать корректными. Однако основная оценка здесь — за представителями пользователей или их непосредственным заместителям.

**Осуществимость**

Необходима возможность реализовать каждое требование при известных возможностях и ограничениях системы и рабочей среды, а также в рамках временных, бюджетных и ресурсных ограничений проекта. Разработчик, участвующий на этапе выявления требований может выполнить проверку на предмет того, что можно и что нельзя выполнить с технической точки зрения, а на что требуются только исключительные затраты и усилия. Инкрементальная разработка и прототипы проверки концепции — два способа проверить осуществимость требования. Если требование нужно удалить, потому что оно неосуществимо, нужно понимать его влияние на концепцию и границы проекта.

**Необходимость**

Каждое требование должно отражать возможность, которая действительно предоставит заинтересованным лицам ожидаемую бизнес-пользу, выделит продукт на рынке или необходима для соблюдения внешних стандартов, политик или правил. Каждое требование должно поступать из источника, которое имеет полномочия на предоставление требований. Отследите каждое требование вплоть до стадии сбора мнений пользователей, когда выявлялись варианты использования или пользовательские истории. Вы должны отследить каждое требование до бизнес-целей, которые четко определяют, почему оно необходимо. Если кто-то спросит, почему включено то или иное требование, у вас будет наготове обоснованный ответ.

**Назначение приоритетов**

Определяйте приоритеты бизнес-требований на основании важности для получения требуемой пользы. Назначьте приоритеты реализации каждому функциональному или пользовательскому требованию, потоку вариантов использования или функции, чтобы определить, насколько они важны для конкретного выпуска продукта. Если все требования одинаково важны, менеджеру проекта будет трудно справиться с нарушением сроков, потерей персонала или добавлением новых требований в процессе разработки. Приоритизация требований должна выполняться коллективно, чтобы учесть интересы многих заинтересованных лиц. Подробнее о назначении приоритетов см. главу 16.

**Недвусмысленность**

Естественный язык несет в себе два типа двусмысленности. Первый тип я могу определить самостоятельно, когда прикидываю, нет ли более одного способа интерпретации данного требования. Другой тип двузначности поймать сложнее. Это происходит, когда разные люди читают одно требование, но интерпретируют его по-разному. Они находят требование содержательным, но оно имеет для них разное значение. Рецензирование — хороший способ обнаружить двусмысленности (Wiegers, 2002). Формальное рецензирование коллегами, такое как освидетельствование (в отличие от простой передачи требований коллегам для самостоятельного изучения), предоставляет возможность каждому участнику сравнить свое понимание каждого требования с другими. Понятие «исчерпывающее» связано с понятием «недвусмысленное»: читатели должны понимать, о чем идет речь в требовании. Процесс

дружественного рецензирования описан в главе 17. Вам никогда не удастся полностью устранить двусмысленность из требований — такова природа человеческого языка. В большинстве случаев разумные люди делают правильные выводы даже из немного туманных требований. Но помощь со стороны коллег в виде рецензирования позволит избавиться от многих самых плохих проблем

**Проверяемость**

Сможет ли тестировщик разработать тесты или применить другие приемы, чтобы установить, действительно ли в продукте реализовано каждое требование? Если требование не поддается проверке, вопрос корректности его реализации становится предметом оценки, а не объективного анализа. Неполные, несогласованные, невыполнимые или двусмысленные требования также не поддаются проверке. Тестировщики знают толк в рецензировании требований на пригодность для проверки. Включите их в рецензенты своих требований, чтобы обнаружить проблемы как можно раньше.

**Характеристики наборов требований**

Недостаточно получить прекрасные отдельные формулировки требований. Набор требований, составляющий базовое соглашение о требованиях (baseline) для определенного выпуска или итерации, должен отвечать характеристикам, описанным в следующих разделах, будь они записаны в документе спецификаций требований к ПО, средстве управления требованиями, наборе пользовательских историй и приемочных тестах или любой другой форме.

**Полнота**

Никакие требования или необходимые данные не должны быть пропущены. На практике никогда не документируются *все и каждое* требование к системе. Всегда есть предполагаемые и подразумеваемые требования, хотя они и более рисковые, чем явно сформулированные требования. Недостающие требования сложно обнаружить, потому что их просто нет! В разделе «Предотвращение неполноты» далее в этой главе предлагаются некоторые способы определения отсутствующих требований. Любая спецификация, со-

держащая записи TBD является неполной.

**Согласованность**

Согласованные требования не конфликтуют с другими требованиями того же типа или с высокоуровневыми пользовательскими, системными или бизнес-требованиями. Если не устранить несогласованность требований до начала процесса разработки, ее придется устранять разработчикам. Рекомендуется записывать автора каждого требования, чтобы узнать, кто его высказал, если будет обнаружен конфликт. Выявить несоответствия может быть сложным, если соответствующая информация хранится в разных местах, например в документе концепции и границ проекта и в средстве управления требованиями.

**Способность к модификации**

Требование всегда можно переписать, но нужно поддерживать историю изменений каждого требования, особенно после достижения базового соглашения о требованиях. Нужно также знать о связях и зависимостях между требованиями, чтобы можно было найти те, что должны меняться одновременно. Возможность модификации диктует, что каждое требование должно

уникально именоваться и выражаться отдельно от других, чтобы на него можно было однозначно ссылаться из других требований. Подробнее о разных способах именования требований см. главу 10. Для сохранения возможности модификации избегайте излишней информации в требованиях. Повторяя требования в нескольких местах спецификации требований к ПО, где они уместны, вы облегчаете чтение документа, но затрудняете его поддержку. Одинаковые требования придется изменять одновременно, в противном случае возникнет несогласованность. Перекрестные ссылки на связанные между собой элементы в спецификации помогут вам синхронизировать последние при внесении изменений. Храня отдельные требования в средстве управления требованиями или в базе данных только

один раз, вы решите проблему избыточности и упростите повторное использование требований, общих для нескольких проектов. Подробнее о повторном использовании требований см. главу 18.

**Отслеживаемость**

Отслеживаемое требование можно отследить как назад, к первоисточнику, так и вперед, к производным требованиям, элементам дизайна и исходному коду, который его реализует, а также к тестам, которые позволяют проверить корректность реализации. Заметьте, что на самом деле не обязательно определять все эти связи для требования, чтобы получить свойства, обеспечивающие возможность отслеживания требования. Отслеживаемые требования обозначаются постоянными уникальными идентификаторами. Они записаны в структурированной, детализированной форме, в противоположность длинным абзацам в повествовательной форме. Избегайте соединения нескольких требований в одну формулировку, потому что разные требования могут быть связаны с различными элементам разработки. Подробнее об отслеживании связей требований см. главу 29. Вам никогда не удастся создать идеальную спецификацию, в которой бы у *всех* требований присутствовали *все* эти атрибуты. Однако если вы будете помнить об этих характеристиках при написании и проверке требований, вы создадите более качественную спецификацию требований, а значит, и программный продукт.

**Принципы создания требований**

Не существует выверенного способа написания идеальных требований, а лучший учитель — это опыт, который нарабатывается со временем. Конструктивная критика проницательных коллег — огромное подспорье, потому что позволяет понять, когда ваш текст выполняет свою задачу, а когда нет. Вот почему рецензирование документов требований коллегами так важно.

Начните процесс рецензирования с дружбы с коллегой бизнес-аналитиком и взаимного рецензирования требований. Вы узнаете, как другой бизнес аналитик пишет требования, и внесете свой вклад в успех команды за счет более раннего обнаружения ошибок и возможностей улучшения. В следующих разделах приводится много советом по написанию требований —особенно функциональных — так, чтобы читатели их понимали. Benjamin

Kovitz (1999), Ian Alexander и Richard Stevens (2002) и Karl Wiegers (2006) предоставляют многие другие рекомендации и примеры написания хороших требований. Говоря «написание требований», люди обычно подразумевают текстовое представление требований на естественном языке. Но лучше мысленно перефразировать «написание требований» на «представление знания о требованиях». Во многих случаях альтернативные варианты представления информации доводят ее более эффективно, чем простой текст (Wiegers, 2006). Бизнес-аналитик должен выбрать подходящее сочетание методов коммуникации, которые гарантируют ясное и единое понимание как потребностей заинтересованного лица, так их решения, которое планируется создать. Представленные здесь примеры требований всегда можно улучшить, и всегда есть эквивалентные способы их представления. При написании требований преследуются две важных цели:

-любой, прочитавший требование, должен придти к той же интерпретации,

что и любой другой читатель этого требования

-интерпретации всех читателей требования должны совпадать с тем, что

автор хотел передать.

Эти результаты важнее, чем чистота стиля и догматическое следование каким-то заданными правилам и соглашениям.

**Системная или пользовательская точка зрения**

Функциональные требования можно писать с точки зрения того, что делает система или что делает пользователь. Так как важнее всего эффективно передать информацию, нет ничего страшного в смешении этих подходов и формулировании каждого требования в максимально понятной форме. Требования следует излагать последовательно, например «Система должна» или «Пользователь должен», затем — активный глагол, а после — наблюдаемый результат. Укажите инициирующие условия или триггеры, вследствие которых система ведет себя определенным образом. Общий шаблон требования, написанного с точки зрения системы, таков (Mavin et al., 2009): [*необязательное предварительное условие*] [*необязательный триггер события*] *система должна [ожидаемая реакция системы*]*.* Шаблон взят из статьи «Easy Approach to Requirements Syntax (EARS)».

EARS также содержит дополнительные конструкции шаблонов для нежелательного поведения, необязательных и сложных требований, а также требований, управляемых событиями или состояниями. Вот пример простого функционального требования, описывающего действие системы, на основе этого шаблона:

*«Если запрошенный химикат есть на складе химикатов, система должна отобразить список всех хранимых на складе контейнеров с указанным химикатом».*

Этот пример содержит предварительное условие, но в нем нет триггера. Некоторые авторы требований опускают из требований фразу «система должна», аргументируя это тем, что поскольку требования описывают поведение системы, нет необходимости постоянно упоминать, что «система должна». В данном примере удаление фразы «система должна» не изменит смысла. Иногда естественнее выразить требование в терминах действий пользователя, а не с точки зрения системы. Включение «должен» и использование активного

залога позволяет четко сказать, какая сущность выполняет действие. При написании функциональных требований с точки зрения пользователя хорошо работает следующая общая структура (Alexander и Stevens, 2002): [*класс пользователя или имя действующего лица*] *должен иметь возможность* [*выполнить что-то*] [*с каким-то объектом*] [*условия выполнения, время отклика или декларация качества].* *Альтернативная формулировка выглядит так: «Система должна позволять (или разрешать или давать возможность) [название класса пользователя] [делать что-то].» Вот пример функционального требования, написанного* *с точки зрения пользователя:*

*Химик должен иметь возможность повторно заказать любой химикат, который он ранее заказывал, путем извлечения и редактирования параметров ранее введенного заказа.*

*Заметьте, что в требовании используется название класса пользователя — Химик, а не общий термин «пользователь». Явная формулировка максимально снижает вероятность неверного истолкования.*

**Язык и стиль**

Написание требований не похоже на создание художественных и нехудожественных произведений. В данном случае не подходит стиль письма, к которому вы привыкли в школе и институте и в котором сначала излагается основная идея, потом описываются подкрепляющие основную идею факты и, наконец, формулируется заключение. Надо изменить стиль письма: первой формулируется суть — формулировка потребности или функциональности, после чего следуют детали (обоснование, источник, приоритет и другие атрибуты требования). Такая структура удобная как читателям, которые бегло просматривают документ, так и тем, кому нужны все детали. Включение таблиц, структурированных списков, диаграмм и других визуальных элементов помогает оживить монотонное перечисление функциональных требований и предоставляет более богатую информацию тем, кому проще осваивать материал нетрадиционными методами. Документы требований также не являются местом, где можно упражняться в креативности и изобретательности в плане стиля. Избегайте смешения активного и пассивного залогов в попытке сделать материал более интересным для чтения. Не обозначайте одно понятие разными терминами, чтобы разнообразить свой текст (клиент, покупатель, посетитель, пользователь, заказчик). Простота чтения и понимания — жизненно важный элемент качественно написанного требования, а вот интересность и увлекательность, честно говоря, не так важны. Если у вас нет опыта, следует ожидать, что ваши читатели не всегда смогут понять, что вы хотели до них донести. Следуйте

приведенным ниже советам при работе над формулировками своих требований, чтобы они максимально эффективно передавали информацию целевой аудитории.

**Ясность и лаконичность**

Пишите требования полными предложениями, с правильной грамматикой, правописанием и пунктуацией Предложения и абзацы должны быть краткими и ясными. Пишите требования простым и прямолинейным языком, характерным для соответствующей предметной области, но не используйте профессиональный жаргон. Определения используемых терминов размещайте в словаре терминов. Еще одна хорошая привычка — писать кратко. Фразы типа «должна предоставлять пользователю возможность сделать» можно свести к одному слову — «должна». Для каждой включаемой в требования порции информации задавайте себе вопрос: «Что читатель будет делать с этой информацией?» Если вы не уверены, что ваша информация будет полезной каким-либо заинтересованным лицам, скорее всего она не нужна. Вместе с тем ясность важнее лаконичности. Точно сформулированные требования повышают вероятность того, что люди получат то, что ожидали; менее точные требования оставляют разработчику больше пространства для свободы интерпретации. Иногда подходит

и не очень большая точность, но в других случаях это может приводить к слишком большому отклонению от желаемого результата. Если разработчик, рецензирующий спецификацию требований к ПО, не совсем понимает цели заказчика, следует включить дополнительную информацию, чтобы снизить риск возникновения проблем в дальнейшем.

**Ключевое слово «должна»**

Традиционно при описании какой-либо возможности системы используют слово «должна». Некоторым это не нравится. Ну и что? Формулировки со словом «должна» четко говорят о требуемой функциональности — в полном соответствии с самой важной целью ясного и четкого доведения информации. Вам может больше понравиться другие слова, такие как «обязана», «следует» или что-то похожее, но нужно придерживаться одного термина. Иногда мне приходится читать спецификации, содержащие произвольные и вводящие в заблуждения наборы глаголов: должна, может, будет, надлежит, должна бы, могла бы, обязана, надо бы, должна бы обеспечить и т. п. Никогда не знаешь, есть различия в значении этих глаголов или нет. Нюансы смысла различных глаголов также сильно усложняют единообразную интерпретацию документа в командах, члены которых относятся к разным культурам. Лучше всего придерживаться ключевого слова «должна» (shall). Некоторые авторы требований специально пользуются разными глаголами для обозначения небольших смысловых различий. Для обозначения приоритета они используют определенные ключевые слова: «должна» означает обязательность, «будет» означает желательную, а «может» — необязательную функциональность (ISO/IEC/IEEE 2011). Мы считаем такие условности опасными. Намного понятнее, когда четко формулируется — «должна» и явно задается приоритет требования: высокий, средний или низкий. Также приоритеты могу меняться по мере завершения итераций, поэтому не нужно их указывать в формулировке требований. Сегодняшнее «должна» завтра может стать «надо бы». Другие авторы используют «должна» в требованиях, а для обозначения ожидаемого поведения дизайна применяют «будет».

При использовании таких соглашений есть риск, что некоторые читатели не поймут разницы между словами, которые в повседневном общении применяются практически в одном значении, поэтому такой разноголосицы нужно избегать.

**Внимание!** Один остроумный консультант предложил мысленно заменить «может» на «скорее всего не» и посмотреть на такое видоизмененное требование. Если такая формулировка неверна, то «может» надо заменить на более точное слово.

**Активный залог**

Используйте активный залог, чтобы дать точно понять, какая сущность выполняет описываемое действие. Много научных и бизнестекстов написаны с использованием пассивного залога, но он никогда не бывает таким ясным и прямым, как активный залог. Следующее требование написано с использованием пассивного залога:

*После доставки обновления продукта будет изменен регистрационный но-*

*мер в соответствующей строке договора.*

Фраза «будет изменен» — признак пассивного залога. Она обозначает адресата (регистрационный номер), но не исполнителя действия. То есть эта фраза ничего не говорит о том, кто или что обновляет регистрационный номер. Выполняет ли это система автоматически или регистрационный номер должен обновить пользователь? Изменение формулировки этого требования с использованием активного залога явно показывает действующее лицо и уточняет инициирующее событие:

*Когда отдел обслуживания подтвердит, что они отправили обновление*

*продукта, система должна обновить клиентский договор, задав в нем новый*

*регистрационный номер продукта.*

**Раздельные требования**

Избегайте длинных повествовательных абзацев, которые содержат несколько требований. Читатели не должны вылавливать изюминки отдельных требований из единой безликой массы текста. Четко отделяйте требования от вводной и контекстной информации. Такая информация полезна читателям, но они должны четко отличать формулировки требований. Как-то мне пришлось рецензировать объемную спецификацию требований, написанную в форме длинных абзацев. Я читал страницы «цельнолитого» текста и понимал его, но было очень сложно выловить в нем конкретные требования. Другие читатели могут придти к неправильным выводам относительно того, где в этом тексте скрываются зерна требований.

Наличие в требовании таких слов, как «и», «или», «дополнительно» и «также», предполагает, что несколько требований могли быть объединены. Это не означает, что нельзя использовать союз «и», но если вы делаете это, проверяйте, соединяет ли он две части одного требования или два отдельных требования. Если для проверки двух частей требуются разные тесты, разбейте предложение на отдельные требования. Никогда не используйте «и/или» в требованиях; это оставляет читателю свободу интерпретации, вот пример:

*Систем должна поддерживать поиск по номеру заказа, номеру счета и/или*

*по номеру пользовательского заказа на покупку.*

Такое требование позволило бы пользователю ввести одно, два или три числа при выполнении одной операции поиска. Скорее всего, это не то, что планировалось. Такие выражения, как «пока не» и «кроме» также указывают на наличие нескольких требований:

*Кредитная карточка покупателя должна считаться действительной для платежей до тех пор, пока не истечет ее срок действия.*Отсутствие информации о том, что происходит, когда выполняетсяусловие «пока не», — обычная причина наличия недостающих требований.

Разделите это положение на два — для двух условий: когда кредитная карточка действительна и когда срок ее действия истек: *Если кредитная карточка покупателя действительна, система должна* *выполнить платеж по этой карточке.* и *Если срок действия кредитной карточки покупателя истек, система* *должна предоставить покупателю возможность обновить информацию своей кредитной карты или ввести для платежа информацию другой кредитной* *карточки.*

**Уровень детализации**

Требования должны определяться настолько подробно, чтобы предоставлять разработчикам и тестировщикам ровно столько информации, сколько необходимо для их реализации.

**Достаточный уровень детализации**

Важной частью анализа требований является разбиение высокоуровневого требования на части, столь мелкие, чтобы они объясняли и дополняли суть происходящего. Нет универсального ответа на обычно задаваемый вопрос: «Насколько детальными должны быть

требования?» Требования должны быть сформулированы достаточно подробно, чтобы риск непонимания был минимальным, для этого необходимо учесть знания и опыт разработчиков. Чем меньше возможностей для обсуждения проблем с текущими требованиями, тем больше деталей нужно включить в набор требований. Если разработчики предлагают несколько способов удовлетворения требования и все они приемлемы, значит, особенности продукта и детализация изложения выбраны верно. Дополнительные детали требуются, когда (Wiegers, 2006):

-работа выполняется для внешнего клиента;

-разработка или тестирование будет передаваться на сторону;

-члены проектной команды распределены географически;

-тестирование системы будет выполняться на основе требований;

-необходимы точные оценки;

-требуется отслеживать связи требований.

Можно сократить объем деталей, когда:

-работа выполняется для внутренних нужд компании;

-клиенты активно участвуют в разработке;

-у разработчиков значительный опыт и знания предметной области;

-есть прецеденты и когда новое приложение создается на замену предыду-

щему;

-будет использоваться пакетное решение.

**Единый уровень детализации**

Создатели документации зачастую тратят массу сил, чтоб «поймать» нужный уровень детализации. Не обязательно описывать все ваши требования с одним уровнем детализации. В частности, можно подробнее описывать те области, которые характеризуются повышенным риском. Тем не менее, в наборе связанных требований лучше придерживаться одного уровня детализации при описании функциональных требований.

Попробуйте описать требования, которые можно протестировать по отдельности. Число поддающихся тестированию требований были предложены в качестве метрики, определяющей размер ПО (Wilson, 1995). Если вам удастся придумать небольшое число взаимосвязанных вариантов тестирования, чтобы проверить правильность реализации требования, скорее всего необходимый уровень детализации достигнут. Если ваши тесты многочисленны и разнообразны, вероятно, несколько требований соединены вместе. Их

следует разделить на более простые. Мне приходилось видеть в одной и той же спецификации положения, которые значительно варьировались в их границах. Например, следующие две функции были разбиты на отдельные требования:

1. *Комбинация клавишей Ctrl+S должна интерпретироваться как «Сохранить файл».*

2. *Комбинация клавишей Ctrl+P должна интерпретироваться как «Печать файла».*

Это очень детальные требования. Для проверки правильного поведения потребуется очень мало тестов. Представьте себе несуразно длинный список подобных требований, который лучше выразить в виде таблицы, перечисляющей сочетания клавиш и соответствующие действия системы. Вместе с тем та же спецификация требований содержала крупное функ-

циональное требование:

*Продукт должен реагировать на команды редактирования, введенные голосом.* Это простое требование — по виду не больше и не меньше остальных мелких требований в спецификации требований — предусматривает включениесложной подсистемы распознавания речи — практически целый самостоятельный продукт! Проверка одного этого требования в работающей системепотребует сотен тестов. Такая формулировка требования уместна на высокомуровне абстракции, который уместен в документе концепции или маркетинговых требований, но требование по распознаванию речи совершенно точно

требует больше подробностей функциональности.

Попробуйте описать требования, которые можно протестировать по отдельности. Число поддающихся тестированию требований были предложены в качестве метрики, определяющей размер ПО (Wilson, 1995). Если вам удастся придумать небольшое число взаимосвязанных вариантов тестирования, чтобы проверить правильность реализации требования, скорее всего необходимый уровень детализации достигнут. Если ваши тесты многочисленны и разнообразны, вероятно, несколько требований соединены вместе. Их

следует разделить на более простые.

Мне приходилось видеть в одной и той же спецификации положения, которые значительно варьировались в их границах. Например, следующие две функции были разбиты на отдельные требования:

1. *Комбинация клавишей Ctrl+S должна интерпретироваться как «Сохранить файл».*

2. *Комбинация клавишей Ctrl+P должна интерпретироваться как «Печать файла».*

Это очень детальные требования. Для проверки правильного поведения потребуется очень мало тестов. Представьте себе несуразно длинный список подобных требований, который лучше выразить в виде таблицы, перечисляющей сочетания клавиш и соответствующие действия системы. Вместе с тем та же спецификация требований содержала крупное функциональное требование:

*Продукт должен реагировать на команды редактирования, введенные голосом.* Это простое требование — по виду не больше и не меньше остальных мелких требований в спецификации требований — предусматривает включениесложной подсистемы распознавания речи — практически целый самостоятельный продукт! Проверка одного этого требования в работающей системепотребует сотен тестов. Такая формулировка требования уместна на высокомуровне абстракции, который уместен в документе концепции или маркетинговых требований, но требование по распознаванию речи совершенно точнотребует больше подробностей функциональности.

**Способы представления**

Глаза читателя не цепляются за плотную массу объемного текста или длинного списка единообразных требований. Подумайте над тем, как самым эффективным образом донести каждое требование до соответствующей целевой аудитории. Альтернативами выражения требований на естественном языке являются списки, таблицы, наглядные модели анализа, графики, математические формулы, фотографии, звуковые и видеоклипы. Во многих случаях

альтернативы не избавляют от необходимости текстового представления требований, но они служат превосходной вспомогательной наглядной информацией, способствующей более глубокому пониманию предмета читателем. Как-то я проверял требования, которые были представлены в следующем формате:

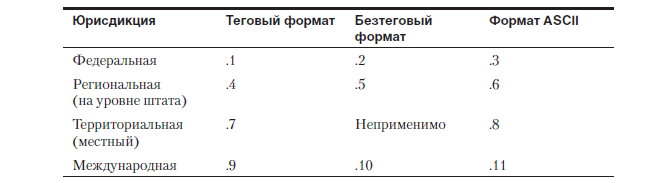
*Текстовый редактор должен уметь анализировать документы следующего*

*<формат>, определяющих законы <юрисдикции>.*

Предлагалось три возможных значения <*формат*> и четыре возможных значения <*юрисдикции*> для двенадцати схожих требований. Спецификация требований к ПО действительно содержала 12 требований, однако из этих двенадцати требований одного не хватало, а одно повторялось. Найти ошибку можно было, составив таблицу всех возможных вариантов, которая компактнее и менее однообразная, чем список требований. Общее требование можно было сформулировать так:

**Editor.DocFormat** *Текстовый редактор должен уметь анализировать документы нескольких форматов, определяющие законы юрисдикций, как показано в табл. 11-1.*

**Табл. 11-1.**Требования по анализу документов



Ячейки таблицы содержат только суффикс, который присоединяется к идентификатору основного требования. Например, третье требование в верхней строке преобразуется в следующее:

**Editor.DocFormat.3** *Текстовый редактор должен уметь анализировать документы ASCII, определяющие федеральные законы.* Если по определенным причинам у каких-то сочетаний нет соответствующих функциональных требований, укажите в соответствующих ячейкахтаблицы «Неприменимо». Это намного понятнее, чем пропустить соответствующее сочетание в списке и заставить читателя гадать, почему нет требования по анализу документов местных законов в безтеговом формате. Этотприем также гарантирует полноту набора требований — если нет пустых ячеек, можно быть уверенным, что вы ничего не пропустили.

**Предотвращение неопределенности**

Качество требований определяет читатель, а не автор. Бизнес-аналитик может считать, что написанное им требование кристально-ясное, свободно от неоднозначностей и других проблем. Но если у читателя возникают вопросы, требование нуждается в дополнительном совершенствовании. Рецензирование — лучший способ поиска мест, где требования непонятны всем целевым аудиториям. В этом разделе рассказывается о популярных причинах неоднозначности требований.

**Неоднозначные слова**

Последовательно используйте термины и именно так, как они определены в словаре. Остерегайтесь синонимов и слов, близких по значению. Я видел проект, в котором для обозначения одного элемента в одном и том же документе требований использовались четыре разных термина. Выберите один термин и последовательно используйте только его, а

синонимы укажите в словаре терминов, чтобы люди, привыкшие к другому названию, могли связать его с вашим термином. При использовании местоимения, ссылающегося на что-то упомянутое ранее, обеспечьте, чтобы было четко понятно, на что вы ссылаетесь. Наречия привносят субъективность, а, значит, неоднозначность. Избегайте таких слов, как *разумно*, *уместно*, *вообще*, *приблизительно*, *обычно*, *систематично* и *быстро*, потому что нет уверенности, что читатель сынтерпретирует их правильно. Требования, изложенные неясным языком, не поддаются проверке, поэтому избегайте двусмысленных и субъективных терминов. В табл. 11-2 перечислены многие из них, а также приводятся рекомендации, как исправить такие неясности. Некоторые из этих слов могут применяться в бизнес-требованиях,

но только не в пользовательских или конкретных функциональных требованиях, описывающих решение, которое планируется создать.

**Табл. 11-2.** Некоторые неоднозначные термины, которых следует избегать в спецификации

к требованиям

|  |  |
| --- | --- |
| **Неоднозначные термины** | **Способы улучшения** |
| Приемлемый, адекватный | Определите, что понимается под приемлемостью и как система это может оценить |
| И/или | Укажите точно, что имеется в виду — «и» или «или», чтобы не заставлять читателя гадать |
| Практически выполнимо | Не заставляйте разработчиков определять, что под этим понимается. Поставьте пометку «TBD» и определите дату, к которой эту проблему следует разрешить |
| По меньшей мере, как минимум, не  более чем, не должно превышать | Укажите минимальное и максимальное допустимые значения |
| Наилучший, самый больший, большинство | Укажите, какой уровень требуется, а также минимальный приемлемый уровень |
| Между, от X до Y | Укажите, входят ли конечные точки в диапазон |
| Зависит от | Определите природу зависимости. Обеспечивает ли другая система ввод данных в вашу систему, надо ли установить другое ПО до запуска вашей системы и зависит ли ваша система от другой при выполнении определенных расчетов или служб? |
| Эффективный | Определите, насколько эффективно система использует ресурсы, насколько быстро она выполняет определенные  операции и как быстро пользователи с ее помощью могут выполнять определенные задачи |
| Быстрый, скорый, моментальный | Укажите минимальное приемлемое время, за которое система выполняет определенное действие |
| Гибкий, универсальный | Опишите способы адаптации системы в ответ на изменения условий работы, платформ или бизнес-потребностей |
| Улучшенный, лучший, более быстрый, превосходный, более качественный | Определите количественно, насколько лучше или быстрее должны стать показатели в определенной функциональной области или аспект качества |
| Включает; включает в себя, но  не ограничен этим; и т. д.; и т. п.; такой  как, в частности | Перечислите все возможные значения или функции, а не только примеры, или укажите читателю, где можно найти  исчерпывающий список. В противном случае разные читатели могут по-разному понимать, что должен содержать полный список и где он должен заканчиваться |
| В большинстве случаев, обычно,  как правило, практически всегда | Уточните, когда указанные условия или сценарии неприменимы и что должно происходить в таком случае. Опишите,  как пользователь или система должны различать разные случаи |
| Соответствует, равняется, согласуется, представляет то же самое | Определите, должно ли сравнение текста быть чувствительным к регистру и что означает эта фраза — «содержит», «начинается с» или «точно совпадает». Для действительных  чисел определите точность при сравнении |
| Максимизируйте, минимизируйте,  оптимизируйте | Укажите минимальное и максимальное допустимые значения определенного параметра |
| Обычно, в идеальном варианте | Опишите нештатные или неидеальные условия и как система должна вести себя в таких ситуациях |
| Необязательно | Укажите, кто делает выбор: система, пользователь или разработчик |
| Возможно, желательно, должно | Должно или не должно? |
| Разумный, при необходимости,  когда уместно, по возможности | Объясните четко, как разработчик или пользователь должен оценивать разумность и уместность |
| Устойчивый к сбоям | Определите, как система должны обрабатывать исключения и реагировать на неожиданные условия работы |
| Цельный, прозрачный, корректный | Что означает «цельный» или «корректный» для пользователя? Выразите ожидания пользователя, применяя характеристики продукта, которые можно наблюдать |
| Несколько, некоторые, много,  немного, множественный | Укажите сколько или задайте минимальную и максимальную границы диапазона |
| Не следует | Старайтесь формулировать требования в позитивной форме, описывая, что именно система будет делать |
| Современный | Поясните этот термин для заинтересованного лица |
| Достаточный | Укажите, какая степень чего-либо свидетельствует о достаточности |
| Поддерживает, позволяет | Дайте точное определение, из каких действий системы состоит «выполнение» конкретной возможности |
| Дружественный, простой, легкий | Опишите системные характеристики, которые будут отвечать потребностям пользователей и его ожиданиям,  касающимся легкости и простоты использования продукта |

**Конструкция типа «A/B»**

Многие спецификации требований содержат выражения вида «A/B», то есть двух связанных терминов (синонимов или антонимов) разделенных косой чертой. Такие выражения как правило неоднозначны. Вот пример:

*Система должна обеспечивать автоматический сбор информации о лицензионных ключах при массовом выпуске продукта отделом доставки/исполнения.* Это предложение можно истолковать несколькими способами:

-«отдел доставки/исполнения» — это название подразделения;

-доставка и исполнение являются синонимами;

*-*в некоторых проектах упомянутое подразделение называется отделом доставки, а в других — отделом исполнения;

-массовый выпуск продукта может выполнять отдел доставки или отдел исполнения, так что косая черта означает «или»;

-массовый выпуск продукта может выполнять отдел доставки или отдел

исполнения, так что косая черта означает «или»;

Иногда авторы применяют конструкцию «A/B», потому что не совсем уверены, что им хочется сказать. К сожалению, это означает, что каждый читатель делает собственные выводы, в чем собственно заключается требование. Лучше точно решить, что хочется сказать, и выразить это правильными словами.

**Пограничные значения**

Много неоднозначности возникает на границах числовых диапазонов как в требованиях, так и бизнес-правилах. Посмотрите на следующее:

*Отпуск длительностью до 5 дней не требует одобрения. Запросы на отпуск длительностью от 5 до 10 дней требуют одобрения непосредственного начальника. Отпуска длительностью 10 дней и более требуют одобрения директора.* При такой формулировке непонятно, в какую категорию попадают отпуска длительностью точно 5 и 10 дней. Если ввести дробные числа, скажем 5,5дней отпуска, то дело еще больше запутается. Слова «от и до», «включительно» и «свыше» вносят четкость и ясность насчет пограничных значений: *Отпуск длительностью 5 дней и меньше не требует одобрения. Запросы на*

*отпуск длительностью более 5 и до 10 дней включительно требуют одобрения непосредственного начальника. Отпуска длительностью свыше 10 дней требуют одобрения директора.*

**Негативные требования**

Иногда люди пишут в требованиях не то, система должна, а то, что она *не должна* делать. Как реализовать такие «негативные» требования? Особенно сложны в расшифровке двойные и тройные отрицания. Попытайтесь переформулировать негативные требования в позитивном

стиле, которые описывает ограничение на поведение. Вот пример: *Пользователь не должен иметь возможность активизировать договор,* *если он не сбалансирован.* Лучше перефразировать это двойное отрицание («не должен» и «не сбалансирован») как позитивное утверждение: *Система должна позволять пользователю активировать договор, только*

*если этот договор сбалансирован.*

Вместо применения негативных требований для указания того, что определенная функциональность выходит за рамки проекта, включите это ограничение в раздел «Ограничения и исключения» документа концепции и границ проекта, как описано в главе 5. Если какое-то требование было в границах проекта, но потом было удалено, лучше не терять его из вида — оно может потребоваться когда-нибудь в будущем. Если требования размещаются в документе, используйте зачеркивание, чтобы отметить удаленное требование. Лучше всего при работе с такими удаленными требованиями использовать атрибут состояния в системе управления требованиями (подробнее об отслеживании атрибутов и состояния требований см. главу 27).

**Предотвращение неполноты**

Нет никакого способа узнать наверняка, что обнаружены все и каждое требование. В главе 7 описывается несколько способов определения отсутствующих требований. Ориентация на выявление пользовательских задач, а не функций системы помогает не упустить функциональность. Использование моделей анализа также помогает обнаружить пропущенные требования (см. главу 12).

**Симметрия**

Симметричные операции — типичный источник пропущенных требований. Как-то я обнаружил в рецензируемой спецификации требований к ПО следующее требование:

*Пользователь должен иметь возможность сохранить договор в любой мо-*

*мент ручного ввода договора.* Мне не удалось найти в спецификации требование, предусматривающеедля пользователя возможность открыть ранее сохраненный неполный договор для завершения работы над ним — скорее всего это требование упусти-ли. Также ничего не говорилось о том, должна ли система перед сохранением проверять данные, введенные в договор. Может это неявное требование?Разработчикам нужно точно знать ответ на этот вопрос.

**Сложная логика**

Сложные логические выражения часто оставляют неопределенными некоторые значения, необходимые для принятия решения. Посмотрите на следующее требование:

*Если не выбран план «Премиальный» и не предоставлено подтверждение страховки, клиент должен автоматически получать план «Базовый».* Это требование ссылается на два двоичных выбора, которые дают четыревозможных сочетания. Но спецификация описывает только одно сочетание.В ней не говорится, что нужно делать, если:

-выбран план «Премиальный», а подтверждение страховки не предоставлено;

-выбран план «Премиальный» выбран и предоставлено подтверждение страховки;

-не выбран план «Премиальный», а подтверждение страховки предоставлено.

Читатель вынужден думать, что в таких ситуациях система не должна предпринимать никаких действий. Это может быть верно, но лучше это сформулировать явно, а не подразумевать. Для представления сложной логики используйте таблицы и деревья принятия решений, чтобы быть уверенным, что не пропустили какие-то варианты.

**Отсутствующие исключения**

У каждого требования, описывающего, как система должна работать, когда все хорошо, должно быть сопутствующее требование, рассказывающее, как поступать в случае исключения. Посмотрите на следующее: *Если пользователь работает над существующим файлом и решит сохранить его, система должна сохранить файл под тем же именем.*

Само по себе это требование ничего не говорит о том, что система должна делать, если не может сохранить файл под тем же именем. Соответствующее второе требование, идущее в паре с первым, может формулироваться так: *Если система не может сохранить файл под определенным именем, она* *должна предоставить пользователю возможность сохранить файл под другим именем или отменить операцию сохранения.*

**Примеры требований: до и после**

В начале главы перечислено несколько характеристик качественно сформулированных требований. Поскольку наличие требований, не отвечающим этим характеристикам, приводит к неразберихе, напрасно затраченным усилиям и последующим переделкам, старайтесь разрешить проблемы на ранних стадиях работы. Далее я покажу несколько, далеких от совершенства требований, взятых из реальных проектов. Исследуйте каждое из них,

используя перечисленные ранее характеристики качества, и попытайтесь определить проблемы. Для начала, например, выясните, поддаются ли они проверке. Если вам не удастся составить тесты, чтобы точно сказать, были ли требования реализованы соответствующим образом, возможно, они неясно сформулированы или не хватает необходимой информации. Мы высказали свои соображения о проблемах для каждого требования и предложили несколько путей их решения. Дополнительные проверки еще их

улучшат, однако на определенном этапе вы должны приступить к непосредственному написанию ПО. Больше примеров по исправлению неудачных требований изложено Ivy Hooks и Kristin Farry (2001), Al Florence (2002), Ian Alexander и Richard Stevens (2002), а также Karl Wiegers (2006). Заметьте, что извлечение требований из подобного контекста показывает их самые неприглядные стороны. Возможно, в исходной среде эти требования выглядели лучше. Мы также предполагаем, что бизнес-аналитики (и другие члены команды) приходят каждый день на работу, чтобы выполнить свою работу наилучшим образом в меру своего опыта и знание, так что наша цель здесь не в придирках к авторам.

**Внимание!** Старайтесь избегать паралича аналитического процесса. Все примеры требований в состоянии «после» можно дополнительно улучшать, но нельзя тратить слишком много времени на попытки сделать требования идеальными. Помните, что ваша цель — написать требования, которые *достаточно хороши*, чтобы команд могла приступить к конструированию продукта при приемлемом уровне риска.

**Пример 1.** *Диспетчер фоновых задач должен предоставлять сообщения о состоянии через регулярные интервалы, составляющие не менее 60 секунд.* Что понимается под сообщениями о состоянии? При каких условиях икак именно они поставляются пользователю? Как долго они должны отображаться на экране? Достаточно ли отображения на протяжении половины секунды? Продолжительность временного интервала не сформулированаточно, а слово «каждый» только вносит дополнительную неясность. Один

из способов оценить требование — проверить, устраивают ли пользователя нелепые, но имеющие право на существование интерпретации этого требования. Если нет, то над требованием необходимо еще поработать. В этом примере интервал должен равняться *не менее* 60 секунд; таким образом, если сообщение будет появляться раз в год, это нормально? А если промежуток *не должен превышать* 60 секунд, то не будет ли интервал, составляющий

одну миллисекунду, слишком коротким? Эти утрированные интерпретациине выходят за рамки первоначального требования, но совершенно очевидно, что пользователь хотел совершенно другого. По этим причинам требование нельзя проверить

.

А вот возможные способы исправления недостатков этого требования, ко- торыми вы можете воспользоваться после того, как получите дополнительную информацию от клиента.

1. Диспетчер фоновых задач (ДФЗ) должен отображать сообщения о состоянии в определенной области пользовательского интерфейса.

1.1. ДФЗ должен обновлять сообщения каждые 60 плюс/минус 5 секунд после запуска фоновой задачи.

1.2. Сообщения должны оставаться видимыми все время, пока работает фоновая задача.

1.3. Если взаимодействие с фоновой задачей возможно, ДФЗ должен отображать процент выполнения фоновой задачи.

1.4. По завершению фоновой задачи ДФЗ должен отобразить сообщение «Выполнено».

1.5. Если фоновая задача «зависла», ДФЗ должен отобразить соответствующее сообщение.

При переписывании несовершенного требования его формулировка становится длиннее из-за необходимости включения отсутствовавшей информации. Разделение требования на несколько дочерних разумно, потому что для каждого понадобится отдельный тест, кроме того, так каждое проще отслеживать. Скорее всего будут дополнительные сообщения о состоянии, отображаемые ДФЗ. Если они задокументированы в другом месте, например в спецификации интерфейса, добавьте эту информацию по ссылке, не дублируя ее. Перечисление сообщений в таблице условий и соответствующих сообщений позволит предоставить информацию более лаконично, чем при написании множественных функциональных требований.

В измененном требовании не указан способ отображения сообщения о состоянии — указано просто «в определенной области пользовательского интерфейса». Такая формулировка делает размещение сообщений задачей дизайна, что допустимо в большинстве ситуаций. Если вы сейчас определите место отображения сообщений здесь, разработчики воспримут ее как ограничение. Излишние ограничения на дизайн расстраивают разработчиков, кроме того, в этом случае вряд ли можно рассчитывать на оптимальный продукт. Но представьте, что мы добавляем эту функциональность в существующее приложение, у которого уже есть строка состояния, где пользователи привыкли видеть важные сообщения. Для единообразия имеет большой смысл предусмотреть условие, что сообщения о состоянии ДФЗ должны отображаться в строке состояния. То есть вы можете специально наложить ограничение дизайна, имея на то очень серьезную причину.

**Пример 2.** *Если возможно, номера счетов следовало бы проверять по списку корпоративных счетов.*

Что означает «если возможно»? Значит ли это «технически осуществимо» (вопрос для разработчика) или «когда доступен основной список счетов»? Если вы не уверены, можно ли реализовать функцию, сделайте пометку «TBD», чтобы указать, что эта проблема еще не решена. После проверки одно из двух будет ликвидировано — либо пометка «TBD», либо требование. В требовании не указано, что произойдет, если проверка пройдет успешно

или окончится неудачей. Избегайте неточных слов вроде «следовало бы». Вот как выглядит исправленный вариант требования. *«В момент ввода номера счета система должна отобразить сообщение об* *ошибке, если этого номера счета в основном корпоративном списке счетов».* Связанное требование будет относиться к условию исключения: основной

список счетов не доступен во время проверки.

**Пример 3.** *Тестер должен позволять пользователю легко подключать дополнительные компоненты, в том числе импульсный генератор, вольтметр, измеритель емкости и нестандартные тестовые платы.* Это требования к продукту, содержащему встроенное ПО, которое используется для тестирования различных типов измерительных приборов. Слово«легко» подразумевает требование легкости и простоты использования, нооно не поддается ни измерению, ни проверке. «В том числе» не дает ясности,полный ли это список внешних устройств, которые должны подключаться киспытательному устройству, или существует еще множество других приборов, о которых мы не знаем. Вот какие альтернативные требования содержатнекоторые преднамеренные ограничения дизайна.

*1. Тестер должен содержать USB-порт, чтобы пользователь смог подключить любое измерительный прибор, у которого есть USB-разъем.*

*2. USB-порт должен быть установлен на передней панели для того, чтобы*

*позволить квалифицированному оператору подключить измерительный*

*прибор за 10 секунд или менее.*

Бизнес-аналитик не должен по собственной инициативе переписывать требования так, чтобы они налагали ограничения дизайна. Вместо этого нужно стремиться обнаруживать несовершенные требования и обсуждать их с соответствующими заинтересованными лицами для уточнения требований.

**Пример 4.***Система должна проверять наличие несоответствий данных счетов между журналом активных счетов и архивом диспетчера счетов. Логика, применяемая для создания этих сравнений, должна основываться на логике существующего средства проверки соответствия. Иначе говоря, код не нужно создавать с нуля. В качестве основы разработчики должны использовать код существующего средства проверки соответствия. Однако нужно добавить дополнительную логику определения базы данных, являющейся полномочным источником. Новая функциональность будет включать запись данных в таблицы, указывающие как и где разрешать несоответствия. Кроме того код должен проверять наличие исключительных сценариев по базе данных средств безопасности. При обнаружении несоответствий в команду безопасности должны отправляться автоматизированные уведомления о сообщениях электронной почты*.

Это хороший пример для практики. Мы укажем на некоторые проблемы в этом разделе, а вы можете попытаться переписать его в поле правильной форме, внося свои предложения по заполнению пробелов. Вот некоторые проблемы, которые вы можете устранить.

-Есть ряд требований, который нужно разбить на отдельные требования.

-Если логика сравнения «основывается» на логике существующего средства проверки соответствия, то какая конкретно часть может повторно использоваться и как ее нужно изменить? Какие функции различаются в новой системе и в существующем средстве? Какую «дополнительную логику» нужно добавить? Как конкретно система может определить, какая

база данных является «полномочным источником»?

-Новая функциональность «включает» запись данных в таблицы — это все или «включена» другая функциональность, о которой не сказано явно?

-Уточните, что означает «как и где», когда речь идет о разрешении несоответствий.

-В некоторых местах нужно использовать слово «должен».

-Какое соотношение между «исключительным сценарием» и «несоответствием»? Если это синонимы, выберите один термин и придерживайтесь именно его. В словаре терминов можно уточнить что это: одно и то же или разное и какая между ними связь.

-Какую информацию система должна отправлять в команду безопасности при обнаружении несоответствия?

-Как уже говорилось ранее, вы никогда не сможете создать идеальные требования. Но опытный бизнес-аналитик может практически всегда помочь сделать требования лучше.

**Что дальше?**

Проведите обсуждение со своими клиентами, разработчиками и тестировщикам, цель которого — оценить текущий уровень проектной документации по требованиям на предмет того, не нужно ли добавить или убрать часть подробностей в тех или иных областях и как представить

эти требования.

Проверьте страницу функциональных требований в спецификации требований к ПО вашего проекта и убедитесь, что каждое положение демонстрирует характеристики отличных требований. Проверьте, нет ли в требованиях проблем, описанных в этой главе. Перепишите любые требования, которые не отвечают этому уровню.

Пригласите от трех до шести заинтересованных в проекте лиц для проверки спецификации требований к ПО для вашего проекта (Wiegers, 2002). Убедитесь, что каждое требование отвечает соответствующим характеристикам, о которых говорилось в этой главе. Проверьте спецификацию на наличие конфликтов между различными требованиями в спецификации, недостающих требований и недостающих разделов. Убедитесь, что обнаруженные недостатки исправлены в спецификации требований к ПО и в любых предыдущих продуктах, созданных

на основе этих требований.

стр 11, 237-259 Вигерс Карл, Битти Джой  Разработка требований к программному обеспечению, 3е Изд.

1. **Написание и анализ требований**

Писать просто и ясно так же трудно*,* как быть искренним и добрым*.*

Сомерсет Моэм, (William Somerset Maugham)

писатель, 1874–1965

# **Введение**

Разработка требований это технический процесс. Поэтому написание требований в корне отличается от написания других текстов, т.е. этот процесс совершенно непохож на написание романов или книг. Написание требований также непохоже и на написание обычной технической документации, например, такой как инструкция по эксплуатации или руководство для пользователей.

Целью данной главы является ознакомление читателя с такими характерными аспектами написания требований, которые являются общими для любого уровня разработки. Как бы не выглядел этот ваш основной процесс разработки, каким бы изменениям он не подвергался, определенная техника и принципы формулирования и структуризации требований остаются неизменными.

Необходимо помнить, что в процессе написания требований два очень важных аспекта должны быть аккуратно сбалансированы:

* документ с требованиями должны быть удобным для чтения;
* наборы требований должны быть удобными для работы с ними.

Под первым подразумевается, что документ с требованиями должен быть структурирован таким образом, чтобы пользователю было легко понять формулировку каждого индивидуального требования в контексте всего документа.

Под вторым аспектом подразумевается качество каждого отдельного требования, каким языком оно написано, насколько оно четко и точно отражает суть, насколько требование может быть представлено в виде некоего элемента, с которым удобно устанавливать связи от других требований.

Опытные специалисты, работающие в области написания требований, хорошо понимают, что как такового текстового редактора явно недостаточно для управления набором требований, создания структуры, классификации, определения свойств и установления связей между требованиями. В качестве иллюстрации неудобства использования текстового редактора можно привести пример, когда нумерация требований осуществляется с помощью номеров параграфов. Попытка вставить в середину документа новое требование приведет к тому, что все последующие требования автоматически изменят свою нумерацию.

Аналогичным образом, те, кто пытается работать с требованиями с помощью баз данных, вскоре приходят к выводу, что наличие многочисленных таблиц, набитых индивидуальными требованиями, все равно практически не позволяет управлять ими как

единой и цельной структурой. Т.е. несмотря на то, что с помощью базы данных легко нумеровать, классифицировать и сортировать требования, жизненно важный смысл всего документа утрачивается, поскольку после таких операций каждое конкретное требование теряет свой смысл, будучи вырванным из общей структуры документа.

Таким образом, оба аспекта – целостность документа и качество отдельного требования должны постоянно находиться под пристальным вниманием.

При этом следует отметить, что написание требований и их анализ (рецензирование) должны происходить параллельно. По той простой причине, что критерии, используемые для написания хороших требований, те же самые, что и для рецензирования (анализа) последних.

Именно поэтому написание и анализ (рецензирование) требований рассматриваются нами вместе в одной главе.

# **Требования для требований**

Прежде чем приступать к обсуждению того, как лучше формулировать требования и составлять документы, необходимо вначале проанализировать общие цели и задачи разработки требований. Это поможет понять суть предлагаемых принципов, которые излагаются в данной главе.

Отправной точкой при разработке требований является определение заинтересованных сторон (stakeholders), как это показано в таблице 4.1.

Таблица **4.1** Заинтересованные стороны (stakeholders) для требований

Заинтересованная сторона Роль

Автор Создает требования и оформляет изменения

Издатель Выпускает и архивирует документ требований

Цензор Рецензирует требования и предлагает изменения

Конструктор, системный аналитик, разработчик

Анализирует требования и обсуждает изменения

В таблице 4.2 представлены те возможности (полезные свойства), предоставляемые требованиями, которые необходимы различным заинтересованным сторонам.

В таблице перечислены все основные действия, которые могут быть совершены с требованиями, включая определение, классификацию, оценку, контроль статуса, контроль связей, рассмотрение в контексте всего документа, анализ и рецензирование. То, насколько хорошо написаны требования, и насколько хорошо организован набор требований, существенным образом влияет на удобство работы с этим набором требований.

Таблица **4.2** Возможности, которые должны предоставлять требования

Возможность

* + - Возможность однозначно идентифицировать каждое положение требования
    - Возможность классифицировать каждое положение требования различными способами, например:
      * по важности
      * по типу (например, функциональность, производительность, ограничение, безопасность)
      * по срочности (дата реализации)
    - Возможность отслеживать статус каждого требования с разных точек зрения, таких как:
      * статус рецензирования (анализа)
      * статус удовлетворения
      * статус проверки
    - Возможность оценивать требование с разных сторон, таких как:
      * информации о производительности
      * стратегия проверки
      * критерии тестирования
      * рациональность
      * комментарии
    - Возможность анализировать каждое требование в контексте целого документа, т.е. в окружении других требований
    - Возможность нахождения в документе определенного требования по контексту, классификации или другим признакам
    - Возможность установления связей между требованиями и легкого перехода по этим связям между требованиями

# Разработка структуры требований

Документация с требованиями может занимать очень большие объемы. Так, например, полный набор требований для авианосца в печатном виде может занимать несколько полностью набитых шкафов. Для того, чтобы доставить такой объем документов от поставщика к заказчику, потребуется даже не один грузовик. В такой ситуации ясная хорошо продуманная структура требований и документов имеет очень большое значение для облегчения управления и реализации всего комплекса требований.

Создание хорошей структуры требований может помочь:

* минимизировать общее количество требований;
* лучше осмыслить большой объем информации;
* отыскать наборы требований, относящихся к определенной теме;
* выявить пробелы и повторения;
* исключить конфликт (противоречия) между требованиями;
* управлять этапами реализации (например, вначале отложенных требований);
* отклонить малоинформативные требования;
* оценить требования (напр., с точки зрения стоимости или времени реализации);
* повторно использовать требования в последующих проектах.

Обычно документ, содержащий требования, состоит из множества секций и подсекций, т.е. уже имеет иерархическую структуру. Такая иерархия секций достаточно удобна для начальной классификации требований, поскольку позволяет использовать структуру заголовков для распределения требований по категориям. При таком подходе та позиция, которую занимает требование в общей структуре документа, является его первичной классификацией. (Вторичная классификация требования может быть организована посредством связей с требованиями других секций или же с помощью атрибутов).

В главе 3 описывается, как часто в системном анализе используются иерархические структуры при моделировании. В качестве иллюстрации можно привести следующие примеры:

* декомпозиция целей и возможностей в пользовательских сценариях;
* функциональная декомпозиция в диаграммах потоков данных;
* декомпозиция состояний системы в диаграммах состояний.

В случае, когда требования формируются на основе анализа моделей, структура этих моделей может использоваться как часть структуры документа.

Помимо формулировки самих требований документ, в котором собраны требования, может также содержать большое количество разного рода технического и нетехнического текста, помогающего лучшему пониманию сути требований.

К такого рода текстам может относиться:

* Сопроводительная информация, которая помогает правильно позиционировать требование в контексте документа;
* Описание внешнего окружения системы или, как это часто называют, «знания о предметной области»;
* Определение границ требований (что включать, а что нет – т.е. границы проекта);
* Определение терминологии*,* используемой в документе;
* Описательный текст, который служит для связи разделов документа между собой;
* Характеристика заинтересованных сторон;
* Краткое описание моделей*,* используемых для получения требований;
* Ссылки на другие документы.

# Понятие о ключевых требованиях

Многие организации начинают применять концепцию «ключевых требований» практически уже на уровне пользовательских требований.

Такие требования, часто называемые ключевыми пользовательскими требованиями KURs (key user requirements) или ключевыми показателями производительности KPIs (key performance indicators), являются небольшой «выжимкой» из общих требований, описывающих суть системы (ее основные функций).

При выборе ключевых требований нужно руководствоваться той же философией, что и герои романа Джерома К. Джерома «Трое в лодке, не считая собаки», которые, собираясь в путешествие, пришли к следующему умозаключению:

Несомненно*,* Темза в своем верхнем течении недостаточно судоходна*,* и по ней не сможет подняться судно*,* которое вместит все*,* что мы сочли необходимым взять с собой*.*

*...*

Джордж сказал*: - «…* Нужно думать не о том*,* что нам может пригодиться*,* а только о том*,* без чего мы никак не сможем обойтись*»*

Каждое ключевое требование должно подразумевать отрицательный ответ на вопрос:

Если решение не предполагает *<*эту*>* возможность *(*функцию*,* опцию*,* т*.*д*.),* стану ли я в этом случае его приобретать*?*

или то же, но на системном уровне:

Если система не обеспечивает *<*этого*>,* будет ли система все еще нужна мне*?*

В этом случае ключевыми требованиями смогут называться только те, которые абсолютно необходимы. (Разумеется, любые требования могут обсуждаться и корректироваться, - в том числе и ключевые, - однако надо помнить, что обсуждение ключевых требований всегда должно проходить с большим вниманием и осторожностью).

Там, где это уместно, каждому ключевому требованию KUR должен быть поставлен в соответствие показатель производительности. В этом случае ключевые требования могут быть использованы как ключевые показатели производительности KPIs для оценки эффективности предлагаемых альтернативных решений.

# Использование атрибутов

Из содержания предыдущих глав, а также из возможностей требований, перечисленных в таблице 4.2, становится ясным, что наличие простого текстового описания явно недостаточно, чтобы полно и однозначно определить требование. Для полного определения требования необходима также и другая информация, которая ему присуща, - признаки классификации, статуса и др.

Чтобы не перегружать формулировку требования излишними деталями, специалисты рекомендуют выносить всю дополнительную информацию в атрибуты, жестко привязанные к требованию.

**[SH234]** Система управления скорой помощью должна быть способна принимать до 100 ста вызовов одновременно

Автор***:*** R. Thomas

Приоритет***:*** Обязательное

Релиз***:*** 1

Статус рецензирования***:*** Одобрено

Возможность проверки***:*** Да

Способ проверки***:*** Симуляция, затем системные тесты

Рис **4.1** Атрибуты требования.

Используя содержимое атрибутов, намного легче обрабатывать требования, осуществлять поиск и выборку, сортировку и т.п. Атрибуты могут быть использованы для поддержания большинства возможностей требований, перечисленных в таблице 4.2, делая сам процесс разработки требований более наглядным, управляемым и удобным.

На рис. 4.1 показан пример требования [SH234] с присущими ему атрибутами.

Использование того или иного конкретного набора атрибутов зависит от используемого компанией процесса разработки требований, который необходимо поддержать. При этом значения некоторых атрибутов могут заполняться автоматически, например, последовательная нумерация или дата; значения других заполняются пользователями (или со слов пользователей), например, приоритет или номер релиза; суть значения третьих – это флаг, устанавливаемый после аналитической работы с требованиями, например, пригодность для проверки.

В таблице 4.3 в качестве наглядного примера, иллюстрирующего вышесказанное, приведены те категории атрибутов требований, которые использовались английским подразделением INCOSE (The International Council on Systems Engineering), работавшим над одним из проектов.

# Связанность и согласованность требований

При работе с большими наборами требований зачастую достаточно трудно идентифицировать те требования, которые могут противоречить друг другу по смыслу. Согласитесь, что, если не иметь специальных средств для выявления подобных конфликтов, не так-то просто понять, что требование, находящееся через несколько страниц от данного, имеет противоположный смысл. Что же может помощь в этом случае?

Ответ достаточно прост. Необходимо иметь возможность классифицировать, фильтровать и сортировать требования с тем, чтобы иметь возможность получать относительно небольшую выборку требований, относящихся к одной теме, для последующего анализа.

При этом многие требования могут одновременно затрагивать различные аспекты функционирования системы. Например, требование, относящееся в основном к вопросу производительности двигателя, может затрагивать и вопросы безопасности. В этом случае данное требование должно рассматриваться как в контексте производительности двигателя, так и в контексте безопасности.

Для поддержания такой возможности требования должны иметь первичную и вторичную классификацию (как обсуждалось в разделе 4.3). Обычно каждое требование имеет единственную первичную классификацию (например, его месторасположение в контексте документа) и множественное количество вторичных классифицирующих свойств, использующих возможности атрибутов и связей.

Эта техника существенным образом помогает при анализе и рецензировании требований, позволяя находить все связанные между собой по смыслу требования с помощью фильтрации и сортировки по ключевым словам и используя признаки основной и дополнительных классификаций.

Например, вначале, чтобы сузить поле возможного поиска, вы строите выборку всех требований, относящихся к безопасности. А затем уже, среди отобранных, вы анализируете схожие требования на предмет наличия конфликтов между ними.

Таблица **4.3** Категории атрибутов

Категория Примеры значений

Идентификация

* + Идентификатор Уникальный номер требования (ID)
  + Название Уникальное краткое название, характеризующее требование

Внутренние характеристики

* + Основной тип Функциональность, производительность, качество, окружение, интерфейс, ограничение, не требование
  + Качественный подтип Доступность, гибкость, целостность, ремонтопригодность, портативность,

легкость поддержки, легкость использования, квалификация

* + Тип продукта/процесса Продукт, процесс, данные, сервис
  + Количественный/качественный тип Количественный, качественный
  + Фаза жизненного цикла Предварительная концепция, окончательная концепция, разработка,

производство, интеграция/тестирование, внедрение/поставка/установка, функционирование, поддержка, удаление/демонтаж

Приоритет и важность

* + Приоритет Ключевое, необходимое, дополнительное, желательное

(Key, mandatory, optional, desirable)

или

Обязательное, рекомендуется, возможное, желательно

(Must, Should, Could, Wish)

* + Важность Шкала от 1 до 10

Источник и владелец

* + Способ получения Назначение, декомпозиция
  + Источник Название документа или имя заинтересованного лица
  + Владелец Имя заинтересованного лица
  + Согласовано Имя человека

Контекст

* + Набор требований/документ (наилучшим образом управляется с помощью правильного расположения

требования в структуре требований)

* + Объект
  + Границы (рамки)

Проверка и утверждение ***(verification & Validation,*** или ***V&V)***

* + V&V метод Анализ, инспекция, системный тест, модульный тест
  + V&V стадия (см. Фаза жизненного цикла)
  + V&V статус В очереди, проверено, отклонено, не завершено
  + Критерий успешности проверки Зависит от выбранной декомпозиции
  + Критерий утверждения Зависит от выбранного V&V метода

Поддержка процесса

* + Статус согласования Предложено, на согласовании, согласовано
  + Статус проверки Проверено, не проверено, подозрительно
  + Статус удовлетворения Неудовлетворенно, удовлетворено, подозрительно
  + Статус рецензирования Ожидает анализа, принято, отклонено

Уточнение

* + Необходимость Описание того, почему возникла необходимость в данном требовании
  + Комментарии Текстовое уточнение требования
  + Вопросы Вопросы, которые нужны для уточнения требования
  + Ответы Ответы, полученные при уточнении

Прочее

* + Зрелось (стабильность) Количество изменений/время
  + Уровень риска Высокий, средний, низкий
  + Оценочная стоимость
  + Фактическая стоимость
  + Релиз продукта Версия продукта, в которой реализовано данное требование

# Важность требования

Некоторые требования можно отнести к категории «не обсуждаемых».

Т.е. их следует не обсуждать, а именно выполнять, потому как, если конечный продукт не удовлетворяет таким требованиям, то он просто не будет использовать.

Соответственно, другие требования могут обсуждаться и корректироваться.

Так, например, если в соответствии с требованиями система управления работой скорой помощи должна обеспечивать одновременную работу как минимум 100 пользователей, а готовое решение поддерживает только 99 пользователей, работающих одновременно, то такое решение, вероятнее всего, будет все-таки признано удовлетворительным и полезным заказчику и пользователям.

Оценить степень важности (удовлетворенности) требования может быть само по себе трудной задачей. Возможно, что для предыдущего примера достижение показателя в 75 одновременно работающих пользователей будет приемлемой величиной, а вот любая величина ниже 50 будет уже категорически неприемлема, но показатель в 200 пользователей скорей всего будет очень хорошим результатом, который для заказчика будет даже более ценен, чем 100.

Одним из подходов, облегчающих решение этой проблемы, является определение нескольких значений производительности для одного показателя. Ниже приведен пример для трех значений:

* О (обязательный): Обязательный верхний (или нижний) предел значения величины;
* Ж (желаемый): Желаемое значение;
* Н (наилучший): Наилучшее значение.

Каждое из этих значений может храниться в собственном атрибуте требования или же они могут быть описаны непосредственно в тексте требования, например, в такой форме:

«Система должна поддерживать функционирование [О:50, Ж:100, Н:200] одновременно работающих пользователей».

Другой подход заключается в том, чтобы графически (с помощью функции) отобразить значение важности требования в зависимости от показателя производительности. В этом случае важность требования обычно находится в пределах от 1 до 100 единиц.

На рис. 4.2 показаны четыре примера, отображающие различную форму функции значения важности требования.

Функция (а) демонстрирует случай, когда заказчику желательно, чтобы число пользователей, работающих одновременно, стремилось к максимуму, но при этом определен и минимально допустимый показатель – 50 пользователей.

Функция (b) иллюстрирует бинарный случай - либо определенная производительность (в

100 единиц) достигается, либо нет. При этом даже 200 одновременно работающих пользователей не приносят никакой дополнительной пользы заказчику.

Функция (с) отражает случай, когда значение показателя должно быть минимизировано (например, вес устройства), но при этом определен и максимально допустимый вес – 50 кг. Функция (d) - это когда значение показателя должно быть оптимизировано (например, обороты двигателя).

Использование графических функций является весьма наглядным способом представления важности требования. Один взгляд на функцию дает представление о сути требования – требуется ли минимизировать, максимизировать, оптимизировать и т.д. показатель.

100 100

важность

важность

0

50 100 200 пользователей

производительность

0

50 100 200 пользователей

производительность

1. максимизация ***(b)*** определенная величина

100 100

3

7

10 о

важность

важность

0

5 10 20 кг

производительность

1. минимизация

0

б/мин

производительность

1. оптимизация

Рис**. 4.2** Типичные функции важности.

В качестве дополнительного преимущества такой подход дает инженерам возможность осознать степень их свободы при разработке решения, т.е. путем согласования значений показателей производительности для отдельных требований получить в результате наилучшие значения производительности всей системы в целом.

Такой подход весьма часто используется при проведении тендеров для сравнения и оценки однотипных критериев альтернативных предложений6.

Для отображения значения требования может использоваться также и атрибут, содержащий конкретное значение функции в зависимости от производительности системы.

# Язык требований

Использование четкого и ясного языка (согласованных терминов) при написании требований позволяет существенном образом облегчить последующее понимание требований и их классификацию.

Простым примером здесь может служить использование в тексте слова «должен» (должна,

должно), как ключевого слова, обозначающего наличие требования. Некоторые подходы

6 В частности, этот принцип использует Telelogic Assessment Management, базирующийся на Doors.

даже предписывают использование отличающихся ключевых слов для характеристики приоритета требований, например – «должно» (*must*), «рекомендуется» (*should*) и

«возможно» (*may*).

Следует заметить, что язык, используемый для написания требований, в значительной степени зависит от «уровня» документа с требованиями, т.е. существует принципиальное отличие между пользовательскими требованиями, которые относятся к проблемной области, и системными требованиями, которые относятся к области решений (см. раздел 1.7).

Как подчеркивалось в главе 5, пользовательские требования в основном описывают возможности (услуги), необходимые пользователям (т.е. потребности пользователей), или ограничения, связанные с этими возможностями или потребностями. И в этом случае требование, описывающее такую потребность, должно описывать только одну потребность, необходимую или для одного пользователя, или группы из нескольких однотипных пользователей.

При этом в тексте требования должен указываться типа пользователя.

Типичное требование, описывающее возможность (потребность), выглядит следующим образом:

**<**Тип пользователя**>** должен иметь возможность **<**описание возможности**>**

Если существуют определенные требования к производительности или ограничения, связанные только с одним конкретным требованием, то, в этом случае, текст требования может быть дополнен и выглядеть уже следующим образом:

**<**Тип пользователя**>** должен иметь возможность **<**описание возможности**>** с **<**показатель производительности**>** от **<**момент отсчета**>,**

находясь в **<**условия эксплуатации**>**

Так, например, выше сформулированное общее требование в частном случае может выглядеть так (содержать условия производительности и ограничения):

Оператор должен иметь возможность произвести выстрел в течение ***3*** секунд с момента обнаружения цели радаром***,*** находясь в сложных морских условиях

Гораздо реже встречается ситуация, когда атрибут с одним и тем же условием производительности связан с несколькими разными требованиями. Можно представить, например, ситуацию, когда несколько различных требований характеризуются одним и тем же временным параметром.

Однако на практике, когда существует иерархия требований и требования более низкого уровня являются детализацией требования более высокого уровня, это зачастую означает, что требуемое значение атрибута производительности фактически связано с требованием более высокого уровня, а, следовательно, все требования, являющиеся его детализацией, просто наследуют это же значение атрибута.

Часто можно видеть, что ограничения описываются не в тесте самого требования, описывающего возможности (потребности), а отдельно. Это может быть связано с тем, что такое ограничение либо относится целиком ко всей системе и нет смысла многократно

повторять его в каждом требовании, либо, наоборот, эти ограничения касаются как разных аспектов системы и их необходимо выделять для последующего контроля.

Обычно ограничения в пользовательских требованиях упоминаются или как минимально приемлемые параметры производительности, заявляемые заказчиком, или являются следствием необходимости взаимодействия создаваемой системы с окружением (сюда же, что характерно, относятся правовые и социальные системы).

Требование типа ограничение обычно выражается в следующей форме:

**<**Тип пользователя**>** не должен попадать под действие **<**соответствующе законодательство**>**

Пример из реальной жизни:

Водитель скорой помощи не должен подпадать под действие законодательства***,*** предусматривающего ответственность за нарушение правил дорожного движения***.***

Поскольку ограничения относятся к области решений, то язык системных требований отличен от языка пользовательских требований. При формулировке системных требований основной акцент делается на описание системные функции и построение ограничений. Конкретная формулировка требования зависит также от типа ограничения или показателя производительности, которые связаны с этим требованием.

Приведем общий пример описания функции (системное требование), которое содержит требуемое значение показателя производительности (в данном случае нагрузки):

**<**Система**>** должна **<**выполняемая функция**>** не менее чем **<**количество**> <**объект**>** функционируя в **<**условия эксплуатации**>.**

Или в реалиях:

Телекоммуникационная система должна обеспечивать телефонную связь

не менее чем с ***10*** абонентами***,***

функционируя в условиях отсутствия внешнего источника электропитания

Приведем другой пример, описывающий периодическое ограничение:

**<**Система**>** должна **<**выполняемая функция**> <**объект**>**

каждые **<**показатель производительности**> <**единица измерения**>**

На языке инструкции это выглядит:

Кофе***-***машина должна производить горячий напиток

каждые ***10*** секунд

Мы продолжим обсуждение этой темы в последующих разделах.

# Шаблоны требований

В предыдущем разделе был проиллюстрирован язык написания требований с использованием шаблонов. В данном разделе мы продолжим обсуждение этой темы применительно к сбору и формулировке требований типа ограничения.

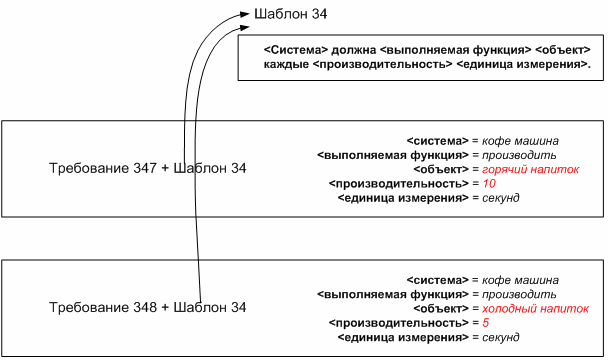
Использование шаблонов, как это показано на примерах, приведенных в предыдущем разделе, является хорошим способом стандартизации языка, применяемого для разработки требований. При этом для того, чтобы иметь возможность писать стандартным способом требования различных типов, необходимо просто собрать определенный набор таких шаблонов. В процессе применения этого набора на практике, он может уточняться, расширяться и корректироваться с тем, чтобы получить более полный набор шаблонов, который в дальнейшем может даже использоваться и в других проектах.

Таким образом, процесс создания требований с помощью шаблонов может быть разбит на два этапа:

* выбор наиболее подходящего шаблона из общего набора шаблонов;
* подстановка конкретных данных для заполнения пустующих полей в шаблоне.

Такой подход дает возможность разделить шаблон и данные, подставляемые в него. Тогда любое требование будет просто содержать ссылку на шаблон, а данные могут фиксироваться отдельно - как атрибуты этого требования.

На рис. 4.3 приведен пример, иллюстрирующий вышесказанное.



Рис**. 4.3** Глобальные шаблоны.

Такой «шаблонный» подход дает возможность сгенерировать текстовое представление требования в любой нужный момент. Использование выделенных шаблонов имеет следующие преимущества7:

* Возможность глобального изменения стиля*:* для изменения формулировки определенных требований необходимо внести изменение только в один или несколько конкретных шаблонов, которые задействованы в этой схеме.
* Возможность более легкой обработки информации*:* например, выделение всех полей, относящихся к <условиям эксплуатации>, в отдельный атрибут требований позволяет более удобно фильтровать и сортировать требования, исходя из конкретных признаков условия эксплуатации.
* Возможность защиты конфиденциальной информации*:* в том случае, когда требования содержат конфиденциальную или секретную информацию, шаблоны могут быть использованы для защиты именно той части текста требования, доступ к которой должен быть защищен.

Последний пункт, несомненно, нуждается в некотором уточнении.

В оборонных и некоторых коммерческих проектах необходимо ограничивать доступ, но не ко всей информации, а только к некоторой ее части. Очень часто текст одного требования содержит информацию, относящуюся к различным уровням секретности.

Например, совершенно очевидно (не секретно), что современный военный корабль будет оснащен ракетами, которые он будет запускать, однако следующая информация о производительности может носить закрытый характер: возможное количество запусков в единицу времени, радиус поражения и т.д.

Вместо того, чтобы ограничивать доступ к целому требованию только из-за того, что некоторая его часть является конфиденциальной, данный подход дает возможность разрешить просматривать всё требование, но без доступа к конфиденциальной информации, содержащейся в некоторых его атрибутах. В действительности, при использовании такого подхода различные участники проекта (с разным уровнем доступа) могут видеть разные наборы атрибутов.

Одним из самых сложных с точки зрения формулирования и, к сожалению, одним из наиболее распространенных типов требований являются ограничения. Так вот как раз для ограничений метод шаблонов существенным образом облегчает задачу формулировки требований.

Авторы предлагают следующий подход для формулирования ограничений:

1. В первую очередь, соберите все возможные требования.
2. Подготовьте список ограничений различных типов, которые могут встретиться при работе над вашим проектом.

Если этот список основан на предыдущем опыте выполнения аналогичного проекта, то, в этом случае, у вас уже есть полный набор шаблонов с ограничениями, который вы можете использовать для описания ограничений текущего проекта. В противном случае, вам придется разрабатывать новые шаблоны.

7 прим*.* переводчика*:* Необходимо также отметить один предполагаемый недостаток метода – для русского языка применение данного метода может осложняться необходимостью согласования падежей.

1. Применительно к каждому требованию рассмотрите полный перечень возможных ограничений из вашего списка и определите, какие именно из них должны быть применимы (зафиксированы) для этого требования.

Для выполнения этой процедуры очень удобно использовать таблицу. Столбцы будут соответствовать различным типам ограничений, а строки – ограничениям.

Если для требования необходимо добавить ограничение определенного типа, то в соответствующей ячейке необходимо сформулировать это ограничение; если же необходимости в ограничении вообще нет, то в соответствующей ячейке нужно поставить «нет» (или N/A = not available).

1. Для каждого ограничения необходимо выбрать наиболее подходящий для него шаблон и сформулировать требование с его помощью.
2. Процесс заканчивается в тот момент, когда все ячейки таблицы заполнены.

Использование такого подхода позволяет ответить на два часто задаваемых вопроса:

* Как формулировать требование, в котором должно быть ограничение? (Ответ*:* использовать шаблоны)
* Как убедиться в том, что все ограничения учтены?

(Ответ*:* использовать таблицу для анализа покрытия требований с ограничениями)

В таблице 4.4 показан пример набора шаблонов для требований с ограничениями.

Обратите внимание на тот факт, что для одного типа ограничений могут использоваться различные шаблоны, а так же на то, что ограничения могут иметь сложную классификацию. В приведенных примерах только текст, выделенный жирным шрифтом, относится непосредственно к самому ограничению.

1. Применительно к каждому требованию рассмотрите полный перечень возможных ограничений из вашего списка и определите, какие именно из них должны быть применимы (зафиксированы) для этого требования.

Для выполнения этой процедуры очень удобно использовать таблицу. Столбцы будут соответствовать различным типам ограничений, а строки – ограничениям.

Если для требования необходимо добавить ограничение определенного типа, то в соответствующей ячейке необходимо сформулировать это ограничение; если же необходимости в ограничении вообще нет, то в соответствующей ячейке нужно поставить «нет» (или N/A = not available).

1. Для каждого ограничения необходимо выбрать наиболее подходящий для него шаблон и сформулировать требование с его помощью.
2. Процесс заканчивается в тот момент, когда все ячейки таблицы заполнены.

Использование такого подхода позволяет ответить на два часто задаваемых вопроса:

* Как формулировать требование, в котором должно быть ограничение? (Ответ*:* использовать шаблоны)
* Как убедиться в том, что все ограничения учтены?

(Ответ*:* использовать таблицу для анализа покрытия требований с ограничениями)

В таблице 4.4 показан пример набора шаблонов для требований с ограничениями.

Обратите внимание на тот факт, что для одного типа ограничений могут использоваться различные шаблоны, а так же на то, что ограничения могут иметь сложную классификацию. В приведенных примерах только текст, выделенный жирным шрифтом, относится непосредственно к самому ограничению.

# Детализация требований

Использование шаблонов для «конструирования» требований позволяет практиковать следующее - некоторые ограничения или показатели производительности могут формулироваться и существовать как отдельные подпункты (подклассы) соответствующих функциональных требований.

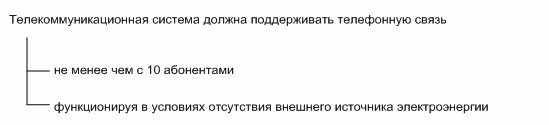
Такой подход позволяет выделять эти подклассы в виде отдельных требований и устанавливать для них связи с соответствующими функциональными требованиям.

Вот тут то, как раз и возникает вопрос о степени (глубине) детализации информации. До какой же степени мы собираемся «расщеплять атом» при разработке требований?

На этот вопрос можно ответить следующим образом: требование может дробиться на подпункты до тех пор, пока средство поддержки работы с требованиями (Requirements Management tool) дает вам возможность видеть каждое требование в нужном контексте.

На рис. 4.4 представлен способ разделения требования, при котором подпункты становятся дочерними требованиями основного требования, образуя, таким образом, определенную иерархию. При этом главное требование остается пригодным для чтения и понимания само по себе, в то время как дочерние требования могут рассматриваться только в контексте

«родительского» требования, хотя в отношении трассировки и связей они могут существовать вполне автономно.



Рис**. 4.4** Показатели производительности и ограничения, как подпункты.

Другими словами, вы можете работать с дочерним требованием как с вполне законченным элементом, пригодным для трассировки, но цитировать (увязывать) дочернее требование лучше все-таки вместе с текстом родительского требования.

Если вновь обратиться к примеру на рис. 4.4, выделив курсивом текст родительского требования в тексте дочерних требований, то имеется в виду следующее:

* Телекоммуникационная система должна обеспечивать телефонную связь
* Телекоммуникационная система должна обеспечивать телефонную связь

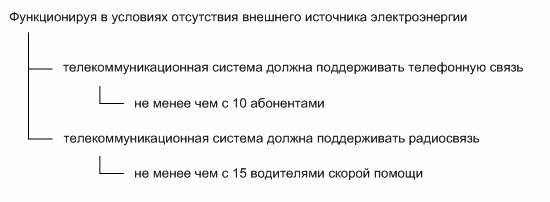
не менее чем с 10 абонентами.

* Телекоммуникационная система должна обеспечивать телефонную связь*,*

функционируя в условиях отсутствия внешнего источника электропитания.

Существуют различные способы организации иерархии таких подклассов. Предположим, что необходимо описать несколько разных возможностей, которые должны быть доступны «в условиях отсутствия внешнего источника электроэнергии».

Вариант организации требований представлен на рис. 4.5.



Рис**. 4.5** Альтернативное представление подпунктов.

Для этого примера взаимосвязь между требованиями будет выглядеть иначе:

* В условиях отсутствия внешнего источника электроэнергии телекоммуникационная система должна поддерживать телефонную связь.
* В условиях отсутствия внешнего источника электроэнергии телекоммуникационная система должна поддерживать телефонную связь*,* не менее чем с 10 абонентами.
* В условиях отсутствия внешнего источника электроэнергии телекоммуникационная система должна поддерживать радиосвязь.
* В условиях отсутствия внешнего источника электроэнергии телекоммуникационная система должна поддерживать радиосвязь*,* не менее чем с 15 водителями скорой помощи.

Другими словами суть такого подхода состоит в том, что при построении иерархии требований (их детализации) родительское требование должно формулироваться таким образом, чтобы обеспечивать полный смысловой контекст для каждого дочернего требования, включая и возможную последующую детализацию дочерних требований на более мелкие блоки информации.

# Критерии для написания текста требований

Независимо от языковых аспектов написания требований, существуют четкие критерии, которым должна удовлетворять формулировка каждого требования. Вкратце эти критерии можно сформулировать следующим образом:

* атомарность: каждое утверждение (формулировка требования) должно представлять собой один элемент иерархии, пригодный для установки связей с ним;
* уникальность: каждое требование должно иметь собственный уникальный идентификатор;
* выполнимость: требование должны быть технически реализуемо в установленные сроки, в рамках выделенного бюджета;
* законность: требование не должно противоречить применимому законодательству;
* ясность: требование должно быть понятно сформулировано (исключать неоднозначное толкование);
* точность: требование должно быть точным и лаконичным;
* проверяемость: должна существовать возможность проверки реализации каждого конкретного требования;
* абстрактность: формулировка не должна навязывать определенные технические решения, характерные для более низких уровней требований (спецификаций).

Дополнительно можно привести критерии, применимые к набору требований:

* полнота: все необходимые требования зафиксированы;
* непротиворечивость: не существует требований, противоречащих друг другу;
* отсутствие избыточности: каждое требование сформулировано только один раз (нет повторов);
* модульность: требования, близкие друг другу по смыслу, содержаться в одном разделе;
* структурированность: наличие ясной и четкой структуры документа с требованиями;
* удовлетворенность: достигнут требуемый уровень покрытия требований связями типа

«удовлетворяется (посредством)»

* тестируемость: достигнут требуемый уровень покрытия требований тестами.

Для иллюстрации того, как не надо делать, ниже приводятся два «жутких» примера формулирования требований:

1. Система должна обеспечивать максимальный уровень производительности в течение всего времени работы, за исключением аварийных ситуаций, при которых она должна обеспечивать уровень производительности до 125%, но только если аварийная ситуации не длится более чем 15 минут, - в противном случае система должна уменьшить уровень производительности до 105%; но в случае, если удается достигнуть уровня производительности только 95%, система должна активировать режим «исключительно малого уровня» и поддерживать этот уровень в пределах 10% от начального значения в течение, как минимум, 30 минут.
2. Система должна обеспечивать основные функции текстового редактора, удобные для использования необученным персоналом, и должна работать в условиях

«тонкого» Ethernet’а, проложенного по воздушной системе кабельных каналов с интегрированными сетевыми адаптерами, поставляемыми с дополнительными модулями памяти, при необходимости.

Эти примеры иллюстрируют классические негативные ситуации, характерные для разработки требований. Для того чтобы избежать этих ошибок, мы рекомендуем следовать простым правилам:

* избегать хаоса: формулируя требование, необходимо сконцентрироваться на самом важном; требование не должно быть похоже на роман;
* избегать *«*лазеек*»*: например, таких выражений, как «если это необходимо», поскольку такие «лазейки» делают требование неоднозначным и зачастую бесполезным;
* избегать размещения более одного требования в один параграф: зачастую, наличие в одном параграфе более одного требования легко определить по наличию союза «и»;
* избегать рассуждений;
* избегать *«*размытых» понятий и слов: обычно, в основном, часто, нормально, типично;
* избегать использования неопределенных терминов: например, удобный в использовании, универсальный, гибкий;
* избегать принятия желаемого за требуемое: напр., 100% надежный, приятный для всех пользователей, безопасный, подходящий для всех платформ, не должен никогда ломаться, обрабатывать все неожиданные сбои, быть готов к модернизациям для любых ситуаций, которые могут возникнуть в будущем и т.д.

Анализ первого примера показывает, что вместо одного требования, нужно писать 12. Развивая эту мысль, лучше всего выделить 4 отдельных условия эксплуатации - нормальное, аварийное, аварийное более 15 минут, режим «исключительно малого уровня»,

- и описать требования для каждого из этих условий.

Обратите внимание на имеющуюся лазейку во втором примере. Совершенно непонятен объем требования. Например, это требование можно интерпретировать и так: «Система должна обеспечивать основные функции текстового редактора … при необходимости».

Так требуется ли, в конце концов, текстовый редактор или нет? – старайтесь избегать таких ситуаций.

# Заключение

Один из самых трудных моментов в разработке требований это начало – что называется,

*«*лиха беда*-*начало*»*.

Разумеется, наличие разного рода методик по конструированию требований и умение ими пользоваться играют важную роль, но также весьма важно и фиксировать каждое требование с самого первого дня работы над проектом, и давать их для анализа и рецензирования другим участникам проекта.

Авторы предлагают использовать следующую «безопасную» последовательность шагов в процессе работы с требованиями:

* Определите структуру требований и пытайтесь совершенствовать ее в процессе работы с требованиями.
* Фиксируйте (записывайте) требования как можно раньше, даже если они весьма далеки от совершенства.
* Заранее определите атрибуты требований, которые в последующем будут использоваться для классификации и детализации требований.
* Как можно быстрее подготовьте первый вариант требований, для того чтобы стимулировать получение отзывов.
* Постоянно совершенствуйте (улучшайте) требования в процессе работы, удаляя повторения, преждевременные и недозволенные технические решения, противоречивость.
* Постоянно обсуждайте требования, собирайте замечания и, не откладывая в долгий ящик, корректируйте требования, создавая новые их версии (принцип «мозгового штурма»).
* Демонстрация пользователям ваших наработок гораздо лучше, чем «экспертный анализ специалистов».

А для написания (формулирования) требований можно следовать нижеприведенным правилам:

* Используйте простой и прямой (однозначный) язык;
* Пишите такие требования, которые могут быть проверены;
* Используйте определенную и согласованную терминологию;
* В одном требовании формулируйте только одно утверждение.
* Начав работать над одним требованием, постарайтесь закончить с ним прежде, чем переходить к другому.

# 84-102 Элизабет Халл, Кен Джексона, Джереми Дика  Разработка и управление требований (практическое руководство пользователя)