GeekBrains

Введение в бизнес-анализ

Документирование

На этом уроке

1. Разберёмся со способами документирования требований.
2. Познакомимся с шаблоном документа об образе и границах проекта.
3. Вспомним варианты использования и способы их документирования.
4. Рассмотрим шаблон спецификации требований к ПО.

Оглавление

Документирование требований

К этапу документирования проект приходит, уже располагая достаточным объёмом требований, чтобы можно было получить представление о продукте.

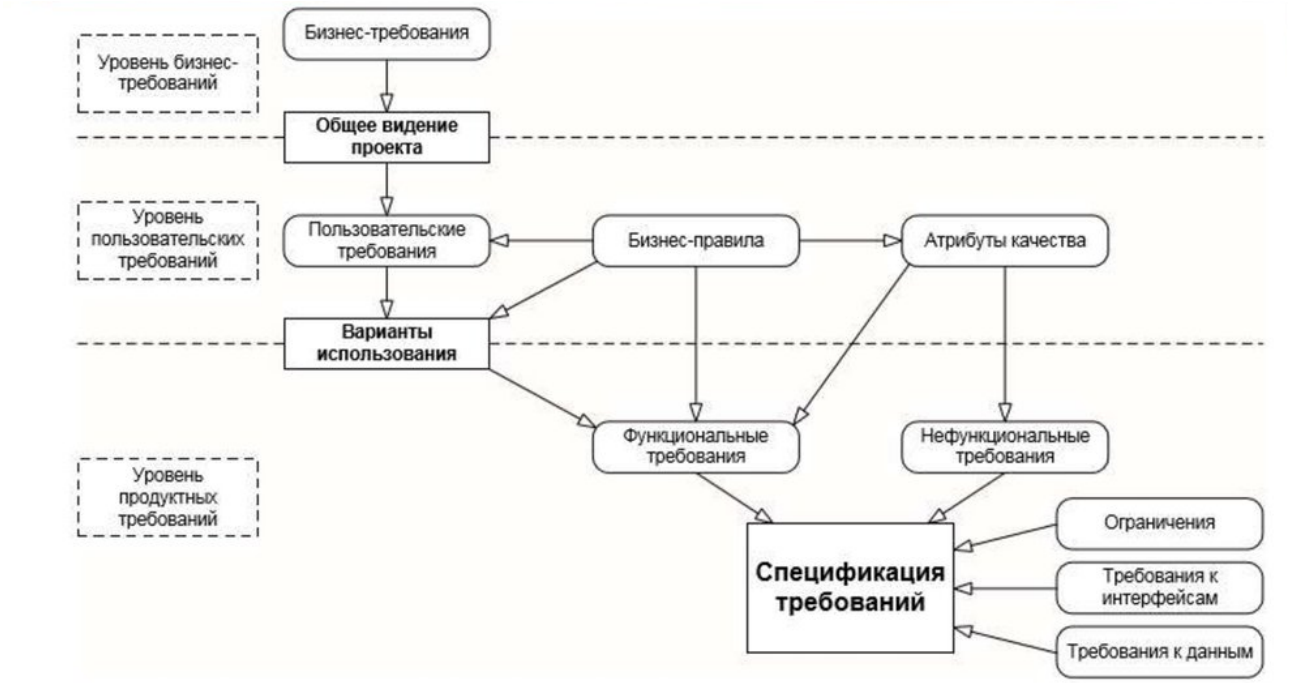
Главный принцип разработки требований — чёткая и эффективная коммуникация сначала с людьми, у которых есть потребности, затем с людьми, которые умеют создавать решения, и в итоге — с теми, кто может реализовать и проверить эти решения.

Опытный бизнес-аналитик выберет самый эффективный способ доведения любого типа требований до любой аудитории.

Итог разработки требований — задокументированное соглашение между заинтересованными лицами о создаваемом продукте.

Каким образом документируются требования

1. Бизнес-требования, наряду с информацией об общем видении проекта, содержатся в **документе об образе и границах проекта**.
2. Пользовательские требования часто фиксируются с помощью вариантов использования продукта (use case) и оформляются в виде **документа пользовательских требований**.
3. Подробные функциональные и нефункциональные требования к продукту записаны в **спецификации требований к ПО**, которая предоставляется тем, кто должен проектировать, разрабатывать и проверять решение.



Фиксация всех требований в виде структурированного и понятного материала, который могут проверить все заинтересованные лица, гарантирует, что они понимают, на что соглашаются.

При создании документов могут возникать следующие трудности:

- сложно хранить описательные атрибуты вместе с требованиями;
- управление изменениями выполняется некорректно;
- сложно сохранять хронологию версий требований;
- непросто выделить часть требований, которые назначены на определённую итерацию, или отслеживать те требования, что были однажды одобрены, но затем отложены или отменены;
- сложно отслеживать связи требований с другими артефактами разработки;
- дублирование требования, которое логически подходит для размещения в нескольких местах, приводит к проблемам с поддержкой.

В качестве альтернативы можно хранить информацию в электронной таблице, у которой те же ограничения, что и у документа, базе данных, вики или серийном средстве управления требованиями. Их надо рассматривать как разные возможные хранилища или контейнеры информации требований. Независимо от формы используемого хранилища, при описании требований всегда нужны одни и те же типы информации. Ниже вы увидите шаблон спецификации требований к ПО. Это удобное средство напоминания, какую информацию надо собрать и как её организовать.

Не все считают, что стоит тратить время на документирование требований. Однако эти затраты ничтожно малы по сравнению со стоимостью получения этих знаний или воссоздания их в какой-то момент в будущем.

Акт спецификации и моделирования помогает участникам проекта обдумывать и точно формулировать важные вещи, которые в устном обсуждении могут остаться невыясненными. Если есть стопроцентная уверенность, что какая-то информация никому из заинтересованных лиц никогда не будет нужна, тогда её можно не регистрировать. В противном случае лучше сохранить данную информацию где-то в общем доступе для всех участников проекта.

Невозможно создать идеальные требования, но нужно помнить, что требования пишутся для определённой аудитории. Объём подробностей, типы предоставляемой вами информации и способ её структурирования должны соответствовать потребностям целевой аудитории. Аналитики естественным образом пишут требования со своей точки зрения, но они должны их писать так, чтобы требования были максимально понятными тем, кто их должен понимать и выполнять на их основе свою работу. Вот почему важно, чтобы представители этих аудиторий рецензировали требования, проверяя их соответствие своим потребностям.

Ключевой принцип эффективной разработки требований — последовательное уточнение деталей.

В большинстве проектов нереально, да и не нужно собирать абсолютно все требования на ранних этапах проекта. Наоборот, надо думать в терминах слоёв. Сначала нам достаточно знать о требованиях столько, чтобы в первом приближении определить их приоритеты и назначить требования на конкретные будущие выпуски и итерации. После этого можно уточнять группы требований в режиме just in time, предоставив разработчикам достаточно информации, чтобы они не выполняли лишних переделок.

Даже самая точная документация по требованиям не сможет заменить обсуждения в процессе выполнения проекта. Должны быть открыты все каналы коммуникации: между бизнес-аналитиками, командой разработчиков, представителями клиента и другими заинтересованными лицами, чтобы они могли быстро разобраться со всеми возникающими по ходу проекта вопросами.

Основные способы представления требований

1. Документация, в которой используется чётко структурированный и аккуратно используемый естественный язык.
2. Графические модели, иллюстрирующие процессы преобразования, состояния системы и их изменения, отношения данных, а также логические потоки и т. п.
3. Формальные спецификации, где требования определены с помощью математически точных, формальных логических языков.

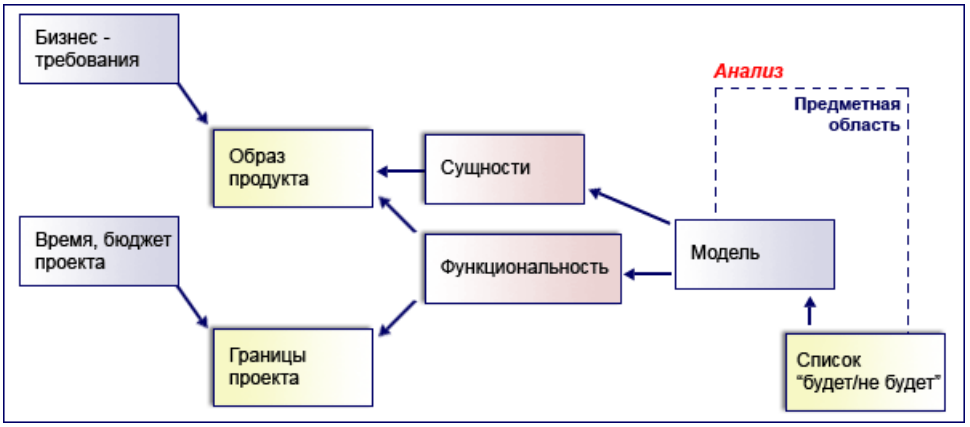
Последний метод обеспечивает наивысшую степень точности, однако с ним знакомы немногие разработчики и ещё меньше клиентов. В большинстве проектов не требуется такого уровня

формализма, но для проектирования высокорисковых систем, таких как системы управления АЭС, применяются формальные методы спецификации. Структурированный естественный язык, усиленный визуальными моделями и другими приёмами представления, такими как таблицы, рабочие модели, фотографии и математические выражения, остаётся самым практичным способом документирования требований в большинстве проектов по разработке ПО.

Документ об образе и границах проекта

Сведения об образе и границах проекта должны быть оформлены в виде отдельного документа, которым владеют лица, финансирующие проект.

Документ об образе и границах проекта (vision) определяет границу и связи системы с остальным миром.



Образ продукта (product vision) — это определение стратегического образа системы, позволяющей выполнять бизнес-задачи.

Образ продукта выстраивает работу всех заинтересованных лиц в одном направлении и описывает, что продукт представляет собой сейчас и каким он станет впоследствии. Образ будет основой принятия решений в течение всего жизненного цикла продукта, так как он содержит описание долгосрочных целей и назначения продукта, которое удовлетворяет различных заинтересованных лиц, основано на существующих или прогнозируемых рыночных факторах, учитывает структуру и стратегию развития организации.

Границы проекта (project scope) показывают, к какой области конечного долгосрочного образа продукта будет направлен текущий проект.

Границы проекта могут относиться к определённой итерации проекта или версии продукта. В положении о границах определена черта между тем, что входит в проект, и тем, что остаётся вовне. То

есть указанные рамки также определяют ограничения проекта. Более детально эти сведения изложены в базовой версии требований, которую разрабатывает для данного проекта команда.

Документ об образе и границах проекта — это описание исходных данных проекта в заданной структуре. Стандартного шаблона такого документа нет, но есть основные разделы, которые должны быть прописаны. Этот документ также может называться уставом проекта, описанием бизнес-требований, документом о концепции и границах проекта.

Информация, которую должен содержать документ об образе и границах проекта

Бизнес-требования

1. Обоснование и содержание нового продукта. Причины, по которым было принято решение о создании продукта.
2. Описание ситуации на рынке, где продукту придётся конкурировать с другими продуктами.
3. Для корпоративной системы приводится описание бизнес-проблемы, которую решает продукт, среды, в которой он будет использован, сравнительная оценка продукта и его конкурентов, преимущества продукта.
4. Описание бизнес-целей и критерии оценки их достижения в количественном и измеряемом виде, рисков, возможных потерь от них, способов минимизации рисков.

Положение об образе проекта

1. Целевая аудитория пользователей, их потребности или возможности.
2. Имя, категория и ключевое преимущество. Это основы для использования.
3. Отличия от конкурентов или текущего бизнес-процесса, описание основного отличия и преимущества продукта.
4. Основные функции и возможности продукта, предоставляемые пользователю. Функции и возможности должны быть идентифицированы, а те из них, которые отличают продукт от конкурентов, подчёркнуты.
5. Предположения и зависимости. Перечисляются зависимости проекта от внешних факторов, документируются все предположения, сделанные заинтересованными лицами при разработке данного документа.

Масштабы и ограничения проекта

Описывается объём первоначальной версии. Туда включают наиболее важные для широкой аудитории функции, которые можно реализовать как можно раньше, и объёмы последующих версий продукта. Здесь же перечисляются те возможности и характеристики продукта, которые могут ожидать

заинтересованные лица, но включение которых не предусмотрено в продукт или определённую версию.

Бизнес-контекст

Описываются профили заинтересованных лиц. Они включают данные об основной ценности или преимуществе продукта, о вероятном отношении к продукту. Перечисляются наиболее интересные функции, характеристики и другие преимущества нового продукта.

Приоритеты проекта

Перечисляются приоритеты факторов, которыми может оперировать менеджер проекта для достижения успеха. Например, каждый проект имеет пять измеряемых параметров: объём (функции), качество, график, затраты и кадры. Каждый из параметров может относиться к одной из категорий:

- ограничение — лимитирующий фактор, в рамках которого должен оперировать менеджер;
- ключевой фактор — важный фактор успеха, ограниченно гибкий при изменениях;
- степень свободы — фактор, который можно в некоторой степени изменять.

Тогда расстановка приоритетов — это соотнесение параметров и категорий параметров.

Операционная среда

Приводится описание среды, в которой будет использоваться система, и определяются требования к доступности, надёжности, производительности и целостности.

Документ об образе и границе проекта определяет границу и связи системы с остальным миром. Использование в документе контекстной диаграммы позволяет графически иллюстрировать эту границу, описывая внешние сущности, расположенные вне системы, а также потоки данных: управляющие и материальные потоки, протекающие между системой и внешним миром. Диаграммы позволяют уточнить взаимодействие между заинтересованными в проекте лицами.

Документ пользовательских требований

Пользовательские требования часто фиксируются с помощью вариантов использования продукта (use case) и оформляются в виде **документа пользовательских требований**. Вспомним определение варианта использования из третьего урока.

Вариант использования *(use case) — это отдельное, независимое действие, которое действующее лицо может выполнить для получения определённого значимого результата.*

Варианты использования применяют при описании пользовательских требований к системе, выясняя, **какие задачи собирается с её помощью решать пользователь.**

Описание варианта использования — это таблица, в которой относительно заданных параметров для вариантов использования прописывают действия пользователя и реакцию системы.

Диаграммы вариантов использования (use case diagrams), разработанные, например, в нотации UML, позволяют получить визуальное представление о требованиях пользователей. Полезно сочетать оба способа — табличный и графический — для более полного представления кейса.

Подробную методику описания вариантов использования вы найдёте в методичке «Урок 3. Заинтересованные лица и участники проекта».

Спецификация требований к ПО

Спецификация требований к ПО (software requirements specification, SRS) — согласно IEEE Std 1012:2004, структурированный набор требований к программному обеспечению и его внешним интерфейсам. В него входят функциональность, производительность, конструктивные ограничения и атрибуты.

Этот документ предназначен, чтобы установить базу для соглашения между заказчиком и разработчиком о том, как должен функционировать программный продукт. Спецификацию требований к ПО в различных компаниях называют по-разному:

- документом бизнес-требований;
- функциональной спецификацией;
- спецификацией продукта;
- просто документом о требованиях.

В спецификации требований к ПО указываются функции и возможности, которыми должно обладать ПО. Она должна содержать функциональные требования — достаточно подробное описание поведения системы при различных условиях, а также нефункциональные требования — необходимые качества системы и накладываемые на неё ограничения, такие как производительность, безопасность и удобство использования.

Спецификация требований служит основой для дальнейшего планирования, дизайна и кодирования, а также базой для тестирования пользовательской документации. Однако она не должна содержать подробностей дизайна, проектирования, тестирования и управления проектом, за исключением особенностей дизайна и реализации, относящихся к ограничениям.

Важно понимать, что один материал требований часто не удовлетворяет потребностям всех типов заинтересованных лиц. Одним нужно знать только бизнес-цели, другие хотят получить

высокоуровневую общую картину, третьих интересует только видение с точки зрения пользователя, но есть и такие, которым требуются все подробности. Вот почему рекомендуется создавать такие материалы, как документ об образе и границах проекта, документ пользовательских требований и спецификация требований к ПО. При этом не стоит ожидать, что все классы пользователей прочтут подробную спецификацию требований к ПО, а разработчики узнают всё, что им нужно, из набора вариантов использования или пользовательских историй.

Спецификация требований к ПО необходима различным участникам проекта.

Участники проекта	**Необходимость**
Клиенты, отдел маркетинга и специалисты по продажам	Для представления о конечном продукте
Менеджеры проекта по данным спецификации	Для составления графиков работ, а также расчёта затрат и ресурсов
Команда разработчиков ПО	Для получения представления о том, какой продукт надо создавать
Тестировщики	Для составления основанных на требованиях тестов, планов тестирования и процедур
Специалисты по обслуживанию и поддержке	Для получения представления о функциональности каждой составной части продукта
Технические писатели	Для создания дизайна пользовательского интерфейса, руководств пользователей и окон справки
Специалисты по обучению	Для разработки обучающих материалов
Персонал, занимающийся юридической стороной проекта	Для проверки соответствия требований к продукту существующим законам и постановлениям
Субподрядчики	Для определения зоны юридической ответственности при выполнении работ по проекту

Если нужной функциональности нет в соглашении о требованиях, нет оснований ожидать, что они появятся в продукте.

Сколько нужно спецификаций

В большинстве проектов создают одну спецификацию требований к ПО, но для крупных проектов это непрактично. В проектах крупных систем часто пишут спецификацию требований системы, к которой прилагаются отдельные спецификации требований к ПО и оборудованию (ISO/IEC/ IEEE, 2011).

Например, если разрабатывается очень сложное приложение управления процессами, где задействовано более сотни сотрудников, в спецификации требований этого проекта может быть около 800 высокоуровневых требований. В таком случае проект должен быть разбит на 20 подпроектов, в каждом из которых будут собственные спецификации требований к ПО.

В простых проектах может возникать ситуация, когда разрабатывают лишь по одному руководящему документу для каждого проекта среднего размера и называют его просто «спецификация». Она содержит всю возможную информацию о проекте: требования, оценки, проектные планы, планы качества, планы тестирования. Управлять изменениями и версиями такого всеобъемлющего документа затруднительно. Рекомендуется разделять выявленные требования по уровням, приведённым в разделе «Документирование требований».

При разработке документации всегда стоит оценивать её объём с точки зрения необходимости и достаточности. Лучшим вариантом будет хранение требований в средстве управления требованиями, которое оказывается очень кстати для решения дилеммы: создавать одну или несколько спецификаций требований в проекте, где планируется несколько выпусков продукта или итераций разработки (Wiegers, 2006). В этом случае спецификация требований к ПО для части продукта или определённой итерации представляет собой всего лишь отчёт, созданный на основе запроса определённого содержимого базы данных.

Не обязательно составлять спецификацию для всего продукта ещё до начала разработки, но необходимо зафиксировать требования для каждой итерации перед её созданием. Это удобно, если в начале работы участники проекта не смогли определить все требования, а некоторую функциональность необходимо быстро передать в руки пользователей. Отклики об использовании ранних итераций позволяют скорректировать оставшуюся часть проекта. Однако при работе над любым проектом необходимо достичь базового соглашения по каждому набору требований до начала их реализации разработчиками. Выработка базового соглашения или базовой версии требований (baselining) — это процесс преобразования реализуемых требований к ПО в то, что будет изучаться и одобряться. При работе с согласованным набором требований снижается вероятность недопонимания и ненужных переделок.

При разработке спецификации требований к ПО следует структурировать информацию таким образом, чтобы все заинтересованные в проекте лица смогли в ней разобраться.

Как сделать требования ясными и понятными

1. Для структурирования необходимой информации использовать соответствующие шаблоны.
2. Разделы, подразделы и отдельные требования должны именоваться единообразно.
3. Использовать последовательно и в разумных пределах средства визуального выделения: полужирное начертание, подчёркивание, курсив и различные шрифты. Цветовые выделения могут быть не видны дальтоникам или при чёрно-белой печати.
4. Создать оглавление, чтобы облегчить пользователям поиск необходимой информации.
5. Пронумеровать все рисунки и таблицы, озаглавить их, при упоминании их в документе использовать присвоенные номера.
6. При использовании ссылок на другие части документа применять перекрёстные ссылки в редакторе, а не сложную кодировку страниц.
7. При использовании документов применять гиперссылки, чтобы можно было быстро перейти к соответствующим разделам спецификации или другим файлам.
8. При хранении требований в специализированном средстве использовать ссылки, чтобы облегчить читателю навигацию.
9. Включать графическое представление информации для облегчения понимания, где это возможно.
10. Воспользоваться услугами опытного редактора, чтобы обеспечить последовательность документа и единообразие терминологии и структуры.

Шаблон спецификации требований

Каждая компания, занимающаяся разработкой ПО, принимает один или несколько стандартных шаблонов спецификации требований к ПО для использования в проектах. Существуют шаблоны спецификаций для проектов различных типов и размеров. Приведённый шаблон подходит для многих проектов.

1. Введение
- 1.1. Назначение.
- 1.2. Соглашения, принятые в документах.
- 1.3. Границы проекта.
- 1.4. Ссылки.

2. Общее описание
- 2.1. Общий взгляд на продукт.
- 2.2. Классы и характеристики пользователей.
- 2.3. Операционная среда.
- 2.4. Ограничения дизайна и реализации.
- 2.5. Предположения и зависимости.

3. Функции системы
- 3.x. Функция системы X.
- 3.x.1. Описание.
- 3.x.2. Функциональные требования.

4. Требования к данным
- 4.1. Логическая модель данных.
- 4.2. Словарь данных.
- 4.3. Отчёты.
- 4.4. Получение, целостность, хранение и утилизация данных.

5. Требования к внешним интерфейсам
- 5.1. Пользовательские интерфейсы.
- 5.2. Интерфейсы ПО.
- 5.3. Интерфейсы оборудования.
- 5.4. Коммуникационные интерфейсы.

6. Атрибуты качества
- 6.1. Удобство использования.
- 6.2. Производительность.
- 6.3. Безопасность.
- 6.4. Техника безопасности.
- 6.x. [Другие].

7. Требования по интернационализации и локализации

8. Остальные требования

Приложение А. Словарь терминов
Приложение Б. Модели анализа

Иногда фрагмент информации логически подходит для нескольких разделов шаблона. В этом случае нужно выбрать один раздел и использовать именно его для информации такого типа в своём проекте. Не стоит дублировать информацию в нескольких разделах, даже если она логически туда ложится. Для быстрого поиска в документе хорошо подойдут перекрёстные ссылки и гиперссылки. При создании спецификации требований к ПО применяют приёмы и средства управления версиями, чтобы все читатели чётко понимали, какую версию они читают в тот или иной момент времени. Также полезно вести журнал изменений, в котором фиксируется суть изменений, автор, дата и причина.

Описание разделов шаблона спецификации требований

1. Введение

Введение представляет собой обзор, помогающий читателям разобраться в структуре и принципе использования спецификации требований к ПО.

1.1. Назначение

Определение продукта или приложения, требования для которого указаны в данном документе, в том числе редакция или номер выпуска. Если эта спецификация требований к ПО относится только к части системы, её необходимо идентифицировать. А также описать типы читателей, которым адресован этот документ, например разработчикам, менеджерам проектов, маркетологам, пользователям, тестировщикам или составителям документации.

1.2. Соглашения, принятые в документах

Указание на все стандарты или типографические соглашения, включая значение стилей текста, особенности выделения или нотацию. Можно определить принятый формат на случай, если кому-нибудь позже понадобится добавить требование.

1.3. Границы проекта

Краткое описание ПО и его назначения. Описание, как связан продукт с пользователями или корпоративными целями, а также с бизнес-целями и стратегиями. Если есть отдельный документ о концепции и границах проекта, не стоит повторять его содержимое, достаточно ссылки. Если спецификацию требований к ПО предполагается разрабатывать постепенно, она должна содержать собственное положение о концепции и границах продукта в качестве подраздела долгосрочной стратегической концепции. Можно предоставить высокоуровневую сводку главной функциональности выпуска или функций, которые он должен выполнять.

1.4. Ссылки

Перечисление всех документов или других ресурсов, на которые есть указания в этой спецификации, в том числе гиперссылки на них, если их местоположение не будет меняться. Это могут быть руководства по стилям пользовательского интерфейса, контракты, стандарты, спецификации к системным требованиям, спецификации интерфейса и спецификации требований к ПО связанных продуктов. Объём информации должен быть достаточным, чтобы пользователь сумел при необходимости получить доступ к каждому указанному материалу: название, имя автора, номер версии, дата, источник, место хранения или URL-адрес.

2. Общее описание

В этом разделе представлен общий обзор продукта и среды, в которой он будет применяться, предполагаемая пользовательская аудитория, а также известные ограничения, предположения и зависимости.

2.1. Общий взгляд на продукт

Описание контекста и происхождения продукта. Пояснения, является ли он новым членом растущего семейства продуктов, новой версией существующей системы, заменой существующего приложения или совершенно новым продуктом. Если спецификация требований определяет компонент более крупной системы, необходимо показать взаимосвязь продукта с другими подсистемами и основные интерфейсы между ними с помощью визуальных моделей, контекстной диаграммы или карты экосистемы.

2.2. Классы и характеристики пользователей

Описание различных классов пользователей, которые, как предполагается, будут работать с данным продуктом, их характеристики. Некоторые требования могут относиться только к определённым классам. Необходимо выделять привилегированные классы пользователей. Класс пользователей — это подмножество заинтересованных в проекте лиц, их описание приводится в документе концепции и границ проекта. Если есть главный каталог классов пользователей, можно включить описания классов пользователей, просто указав их в каталоге.

2.3. Операционная среда

Описание рабочей среды, в которой будет работать ПО, включая аппаратную платформу, операционные системы и их версии, а также географическое местоположение пользователей, серверов и баз данных вместе с организациями, в которых располагаются соответствующие базы данных, серверы и веб-сайты. Перечисление всех остальных компонентов ПО или приложений, с которыми система должна быть совместима. Если в связи с разработкой новой системы нужно произвести значительную работу с технической инфраструктурой, стоит подумать о создании отдельных требований к инфраструктуре, в которой детально изложить подробности этой работы.

2.4. Ограничения дизайна и реализации

Бывает, что нужно использовать вполне определённый язык программирования или библиотеку, на разработку которой уже потрачено время. В этом случае делают описание и логическое обоснование всех факторов, которые ограничат возможности, доступные разработчикам. Требования должны включать потребности, а не идеи по решению, потому что идеи накладывают ограничения на дизайн, часто неоправданные.

2.5. Предположения и зависимости

Предположение (assumption) — это утверждение, которое предполагается верным в отсутствие знаний или доказательств иного.

Некоторые предположения можно отнести к группе рисков проекта. Разработчик может думать, что определённый набор функций написан специально для этого приложения, бизнес-аналитик — что он будет взят из предыдущего проекта, а менеджер проекта — что предполагается приобрести коммерческую библиотеку функций. Включаемые здесь предположения относятся к системной функциональности. Предположения, относящиеся к бизнесу, представлены в документе об образе и границах проекта.

Определите все зависимости проекта или создаваемой системы от внешних факторов или компонентов вне её контроля. Например, до установки продукта может требоваться установить Microsoft .NET Framework 4.5 или более позднюю версию — это зависимость.

3. Функции системы

Шаблон структурирован по функциям системы — это ещё один способ систематизации функциональных требований. Другие методы классификации: по функциональным областям, рабочим потокам, вариантам использования, режимам работы, классам пользователей, стимулам и реакциям. Возможны также иерархические комбинации этих элементов, например варианты использования внутри классов пользователей. Не существует единственно правильного метода организации. Выбирайте тот, при котором пользователям будет легче понять предполагаемые возможности продукта.

3.x. Функция системы X

Наименование и описание особенностей функции несколькими словами, например «3.1. Проверка правописания». Аналогично называют подразделы с 3.x.1 по 3.x.3 для каждой функции системы.

3.x.1. Описание

Краткое описание функции системы и указание её приоритета: высокий, средний или низкий. Приоритеты — динамичная характеристика, они могут изменяться в ходе проекта. При использовании средства управления требованиями определяется атрибут требований для обозначения приоритета.

3.x.2. Функциональные требования

1. Перечисление по пунктам конкретных функциональных требований, связанных с этой функцией, которые нужно реализовать, чтобы пользователь мог использовать сервисы этой функции или реализовать вариант использования.
2. Описание, как продукт должен реагировать на ожидаемые ошибки, неправильный ввод информации или неверные действия.
3. Присвоение каждому функциональному требованию уникального имени.

4. При использовании средства управления требованиями создают много атрибутов для каждого функционального требования, таких как основание, источник и состояние.

4. Требования к данным

Ценность информационных систем заключается в том, что они предоставляют возможность манипулировать данными. Этот раздел шаблона предназначен для описания различных аспектов данных, которые будет потреблять система в качестве входной информации, как-то обрабатывать и возвращать в виде выходной информации.

4.1. Логическая модель данных

Модель данных — это визуальное представление объектов и наборов данных, которые будет обрабатывать система, а также отношений между ними.

Существует много видов нотаций для моделирования данных, в том числе диаграммы отношений «сущность–связь» и диаграммы классов UML. Можно включить модель данных для бизнес-операций, выполняемых системой, или логическое представление данных, с которыми будет работать система. Это не то же самое, что модель данных реализации, которая реализуется в виде дизайна базы данных.

4.2. Словарь данных

Словарь данных определяет состав структур данных, а также их значение, тип данных, длину, формат и разрешённые значения элементов данных, из которых состоят эти структуры. Серийные средства моделирования данных часто включают словарь данных в качестве компонента. Во многих случаях словарь данных лучше хранить как отдельный артефакт, не внедряя его в спецификацию требований к ПО. Это облегчает возможность его повторного использования в других проектах.

4.3. Отчёты

Если приложение будет генерировать отчёты, их перечисляют и описывают характеристики. Если отчёт должен соответствовать определённому готовому макету, можно указать это как ограничение, возможно, с примером. В противном случае делают логическое описание с указанием порядка сортировки, уровней суммирования и т. п., отложив подробный макет до этапа дизайна.

4.4. Получение, целостность, хранение и утилизация данных

Если это важно, описывают процесс получения и обслуживания данных. Например, при открытии канала номенклатуры данных может требоваться первым делом выполнить начальный дамп всей номенклатуры данных в принимающую систему, а после этого использовать каналы для передачи только изменений. Указывают все требования, относящиеся к защите целостности данных системы, и все процедуры, которые могут потребоваться, например резервное копирование, создание контрольных точек, зеркальное отображение или проверка корректности данных. Также перечисляют все политики, которые должна применять система для хранения или утилизации данных, в том числе

временных данных, метаданных, остаточных данных (таких, как удалённые записи), данных в кеше, локальных копий, архивов и промежуточных архивов.

5. Требования к внешним интерфейсам

В этом разделе указывается информация, которая гарантирует, что система будет правильно взаимодействовать с пользователями и компонентами внешнего оборудования и ПО. Выработка согласия по внешнему и внутреннему интерфейсу системы признана оптимальным приёмом в области разработки ПО (Brown, 1996). В сложной системе с множеством подкомпонентов следует использовать раздельные спецификации для интерфейсов или спецификацию системной архитектуры. В документацию по интерфейсу можно включить ссылки на материал из других документов. Например, ссылка может указывать на руководство по работе с устройством, где перечислены коды ошибок, которые устройство может отправить программе.

5.1. Пользовательские интерфейсы

Это описание логических характеристик каждого пользовательского интерфейса, который необходим системе. Особенные характеристики пользовательских интерфейсов могут упоминаться в разделе «6.1. Удобство использования». Некоторые из них:

1. Ссылки на стандарты графического интерфейса пользователей или стилевые рекомендации для семейства продуктов, которые необходимо соблюдать.
2. Стандарты шрифтов, значков, названий кнопок, изображений, цветовых схем, последовательностей полей вкладок, часто используемых элементов управления, графики фирменного стиля, уведомления о зарегистрированных товарных знаках и о конфиденциальности и т. п.
3. Размер и конфигурация экрана или ограничения разрешения.
4. Стандартные кнопки, функции или ссылки перемещения, одинаковые для всех экранов, например кнопка справки.
5. Сочетания клавиш.
6. Стандарты отображения и текста сообщений.
7. Стандарты проверки данных. Например, какие ограничения накладываются на вводимые значения и когда нужно проверять содержимое полей.
8. Стандарты конфигурации интерфейса для упрощения локализации ПО.
9. Специальные возможности для пользователей с плохим зрением, проблемами с различением цветов и другими ограничениями.

5.2. Интерфейсы ПО

1. Описание связи продукта и других компонентов ПО, идентифицированных по имени и версии. Это могут быть другие приложения, базы данных, операционные системы, средства, библиотеки, веб-сайты и интегрированные серийные компоненты.
2. Назначение, форматы и содержимое сообщений, данных и контрольных значений, обмен которыми происходит между компонентами ПО.
3. Соответствия между входными и выходными данными между системами. Описание всех преобразований, которые должны происходить с данными при перемещении между системами.
4. Службы, необходимые внешним компонентам ПО, природа взаимодействия между компонентами.
5. Данные, которыми будут обмениваться и к которым будут иметь общий доступ компоненты ПО.
6. Нефункциональные требования, влияющие на интерфейс, такие как уровни обслуживания для времени и частоты отклика или меры и ограничения безопасности. Часть этой информации может быть определена как требования к данным в разделе 4 или как требования к взаимодействию в разделе «6. Атрибуты качества».

5.3. Интерфейсы оборудования

Описание характеристик каждого интерфейса между компонентами ПО и оборудования системы. В описание могут входить типы поддерживаемых устройств, взаимодействия данных и элементов управления между ПО и оборудованием, а также протоколы взаимодействия, которые будут использоваться. Перечисляют входные и выходные данные, их формат, разрешённые значения или их диапазоны, а также все временные характеристики, о которых должны знать разработчики. Если такой информации очень много, лучше создать отдельный документ спецификации интерфейса.

5.4. Коммуникационные интерфейсы

Описание требований для любых функций взаимодействия, которые будут использоваться продуктом, включая электронную почту, веб-браузер, сетевые протоколы и электронные формы. Определяют соответствующие форматы сообщений. Описывают особенности безопасности взаимодействия или шифрования, скорости передачи данных и механизмов согласования и синхронизации. Указывают все ограничения этих интерфейсов, например допустимость тех или иных типов вложений в сообщениях электронной почты.

6. Атрибуты качества

В этом разделе описываются нефункциональные требования, помимо ограничений, описанных в разделе 2.4, и требований к внешним интерфейсам, описанных в разделе 5. Эти характеристики должны быть точно определены и должны поддаваться проверке и измерению. Здесь указывают относительные приоритеты различных атрибутов, например приоритет простоты использования над

лёгкостью изучения или приоритет безопасности над производительностью. Необходимые степени качества удаётся гораздо эффективнее описать с помощью подробных нотаций спецификации, например Planguage, чем с помощью простых описательных утверждений.

6.1. Удобство использования

Требования к удобству использования подразумевают лёгкость изучения, простоту использования, предотвращение ошибок и восстановление, эффективность взаимодействия и специальные возможности. Указанные в этом разделе требования к удобству использования помогут дизайнеру интерфейсов создать максимально удобную для пользователя рабочую среду.

6.2. Производительность

Указывают конкретные требования к производительности для различных системных операций. Если у различных функциональных требований или функций разные требования к производительности, то задачи, связанные с производительностью, указывают там же, в разделе соответствующих функциональных требований, а не включают их все в этот раздел.

6.3. Безопасность

Указывают все требования, касающиеся безопасности или конфиденциальности, которые ограничивают доступ или возможности использования продукта. Это может быть физическая безопасность, а также защита данных или ПО. Источник требований к безопасности — это обычно бизнес-правила, поэтому определяют политики или положения, касающиеся защиты или конфиденциальности, которым продукт должен соответствовать. Если они задокументированы в хранилище бизнес-правил, делают ссылку на них.

6.4. Техника безопасности

Указывают требования, связанные с возможными потерями, повреждениями или ущербом, которые могут быть результатом использования продукта. Определяют меры безопасности или упреждающие действия, которые можно предпринять, и потенциально опасные действия, которые можно предотвратить. Определяют сертификаты по безопасности, политики или положения, которым продукт должен соответствовать.

6.x. [Другие]

Для каждого дополнительного атрибута качества продукта в спецификации требований к ПО создают отдельный раздел, чтобы описать характеристики, которые будут важны для клиентов или для разработчиков и сотрудников, ответственных за поддержку. Это может быть доступность, возможность установки, целостность, возможность модификации, переносимость, надёжность, устойчивость, масштабируемость и контролируемость.

7. Требования по интернационализации и локализации

Требования по интернационализации и локализации обеспечивают возможность использовать продукт в других странах, региональных стандартах и географических районах, отличающихся от тех, в которых он был создан.

Такие требования могут быть направлены на разрешение различий в валютах, форматировании дат, чисел, адресов и телефонных номеров, языках, в том числе различных вариантах одного языка (например, американского и британского вариантов английского), используемых символах и наборах символов, личных именах и фамилиях, часовых поясах, международных нормативных актах и законах, культурных и политических традициях, размере используемой бумаги, единицах веса и меры, электрическом напряжении, конфигурации электрических разъёмов и во многом другом. Требования по интернационализации и локализации вполне могут повторно использоваться во многих проектах.

8. [Остальные требования]

В этом разделе определяют все другие требования, которые ещё не были описаны в спецификации требований к ПО. Например, юридические, законодательные или финансовые требования и требования стандартов, требования к установке, конфигурированию, запуску и остановке продукта, а также к журналированию, мониторингу и т. д.

Для различных требований следует добавить новые разделы к шаблону вашего проекта. Например, это могут быть требования к переходу, которые необходимы для миграции с предыдущей системы на новую, если они относятся к создаваемому ПО. Допустим, к программам преобразования данных. В противном случае их можно включить в план управления проектом, например если речь идёт о разработке обучающих материалов или поставке.

Приложение А. Словарь терминов

В этом разделе содержатся все специальные термины, которые читателю необходимо знать для правильного понимания спецификации требований к ПО, включая сокращения и аббревиатуры. Для каждого сокращения должна быть приведена расшифровка и его определение. При необходимости можно создать расширенный общекорпоративный словарь для нескольких проектов, в котором есть ссылки на все термины, относящиеся к данному проекту. В этом случае в спецификации требований к ПО будут определены только те термины, которые относятся лишь к данному проекту и которых нет в общекорпоративном словаре. Определения данных находятся в словаре данных, а не терминов.

Приложение Б. Модели анализа

В этом необязательном разделе упоминаются такие модели анализа, как диаграммы потоков данных, деревья функций, диаграммы переходов состояния и диаграммы «сущность–связь». Часто читателю удобнее, когда определённые модели внедрены в соответствующие разделы спецификации, а не собраны все вместе в конце.

Спецификация требований в проектах гибкой разработки

В проектах гибкой разработки (agile) применяют ряд способов определения требований, которые отличаются от только что описанного метода.

Во многих проектах гибкой разработки в процессе выявления требований используются пользовательские истории. Каждая такая история — это формулировка потребности пользователя или функциональности, которая может быть полезной пользователю или покупателю системы (Cohn, 2004; Cohn, 2010). В проектах гибкой разработки команда может начать спецификацию, записав достаточно информации по каждой пользовательской истории, чтобы заинтересованные лица получили общее представление, о чём история, и смогли определить приоритеты историй друг относительно друга. Это также позволяет команде начать планировать назначение историй на конкретные итерации. Команда может собирать связанные истории в минимально поддающуюся для продвижения на рынке функцию, которую нужно реализовать целиком к выпуску продукта, чтобы клиент мог получить от неё пользу.

Пользовательские истории накапливаются и приоритизируются в резерве продукта (product backlog), который меняется на протяжении всего проекта. Крупные истории, которые охватывают значительную функциональность и которые нельзя реализовать в одной итерации, стоит разбить на более мелкие, которые назначаются конкретным итерациям. Пользовательские истории можно записывать на чём-то простом, например на карточках каталога, а не в традиционном документе. Одни команды гибкой разработки сохраняют свои истории в средстве управления историями после реализации, а другие не делают этого.

Когда команда приступает к очередной итерации, каждая история, назначенная на эту итерацию, уточняется и наполняется деталями в процессе обсуждения между владельцем продукта, людьми, выполняющими роль бизнес-аналитиков, аналитиками, тестировщиками и пользователями. То есть спецификация подразумевает постепенное уточнение подробностей на правильном этапе проекта. Это хорошая практика в любом проекте. Обычно эти подробности соответствуют тому, что мы называли функциональными требованиями в спецификации требований к ПО.

Однако в проектах гибкой разработки эти подробности часто представляют в форме приёмочных тестов, описывающих, как система должна вести себя при правильной реализации истории. Тесты истории проводят во время итерации, в которой реализуется эта история, и в будущих итерациях в рамках регрессионного тестирования. Как и в любых других тестах, в них должны выполняться исключительные условия и ожидаемое поведение. Эти приёмочные тесты можно записывать на карточках или регистрировать в более постоянной форме, например в средстве тестирования. Тесты нужно автоматизировать, чтобы обеспечить быстрое и полное регрессионное тестирование. Если команда решит отказаться от исходных пользовательских историй, тогда приёмочными тестами может быть постоянная документация требований, если они хранятся в специализированном средстве.

Нефункциональные требования также можно записывать на карточках, но не как пользовательские истории, а как ограничения (Cohn, 2004). Альтернативный вариант — указывать нефункциональные требования, относящиеся к определённой пользовательской истории, в форме критериев приёмки как демонстрацию достижения определённого атрибута качества. Например, тесты безопасности могут демонстрировать, что только определённым пользователям разрешён доступ к функциональности, описанной в соответствующей пользовательской истории, а остальным пользователям доступ закрыт. Команде гибкой разработки не возбраняется применять другие методы представления знаний о требованиях, такие как модели анализа или словарь данных. Они должны выбрать метод представления, который привычен и уместен в их культуре и проекте.

Выбор надлежащих форм спецификации требований к ПО — исключительная прерогатива команды проекта. **Главная цель разработки требований** — собрать согласованное понимание требований, качество которых достаточно, чтобы приступить к разработке следующей части продукта и продолжить работу при приемлемом уровне риска.

Надлежащий уровень формальности и подробности документа требований зависит от многих факторов, в числе которых:

- объём оперативного неформального вербального и визуального общения между клиентами и разработчиками, который необходим для получения подробностей, необходимых для правильной реализации каждого пользовательского требования;
- предел, до которого неформальные методы общения могут обеспечить эффективную синхронизацию во времени и пространстве внутри команды;
- граница, до которой представляет ценность или необходимость сохранение знаний о требованиях для будущих улучшений, реинжиниринга приложения, проверки, аудита, проверки на соответствие нормативам, сертификации продукта или в соответствии с условиями контракта;
- граница, до которой приёмочные тесты могут эффективно заменять описания ожидаемых возможностей и поведения системы;
- предел, до которого человеческая память может заменить письменное представление.

Независимо от типа создаваемого командой продукта, используемой методики разработки или используемых бизнес-аналитиком приёмов выявления требований, эффективная спецификация требований жизненно важна для успеха. Существует много путей достижения этой цели. Но если не определить высококачественные требования, полученный в результате продукт будет представлять собой коробку с сюрпризами. Вы никогда не будете знать, что от него можно ожидать.

Что дальше

1. Необходимо проверить набор требований проекта на предмет соответствия шаблону, чтобы определить, собраны ли все требования в разделах, относящихся к вашему проекту. Шаблон — всего лишь напоминание, что собрана вся информация, необходимая для успеха проекта.
2. Если в компании нет стандартного шаблона спецификации требований к ПО, необходимо его принять. Можно начать со стандартного и затем изменить его так, чтоб он наилучшим образом соответствовал потребностям выполняемых проектов и продуктов. Также выработать стандарт именования отдельных требований.
3. Если требования хранятся в форме, отличной от обычного документа, например в средстве управления требованиями, следует изучить шаблон спецификации требований к ПО и определить, нет ли каких-либо категорий информации требований, которые не выявляются и не фиксируются. После чего следует изменить своё хранилище, включив эти категории, чтобы в будущем хранилище напоминало о них при выявлении требований.

Практическое задание

1. Достаём все предыдущие практические задания к урокам № 2–7.
2. Заполняем шаблон документа об образе и границах проекта.
3. Заполняем шаблон документа пользовательских требований.
4. Заполняем шаблон спецификации требований к ПО.

Глоссарий

Вариант использования (use case) — это отдельное, независимое действие, которое действующее лицо может выполнить для получения определённого значимого результата.

Главный принцип разработки требований — чёткая и эффективная коммуникация сначала с людьми, у которых есть потребности, затем с людьми, которые умеют создавать решения, и в итоге — с теми, кто может реализовать и проверить эти решения.

Границы проекта (project scope) показывают, к какой области конечного долгосрочного образа продукта будет направлен текущий проект.

Документ об образе и границе проекта (vision) определяет границу и связи системы с остальным миром.

Итог разработки требований — задокументированное соглашение между заинтересованными лицами о создаваемом продукте.

Ключевой принцип эффективной разработки требований — последовательное уточнение деталей.

Модель данных — это визуальное представление объектов и наборов данных, которые будет обрабатывать система, а также отношений между ними.

Образ продукта (product vision) — это определение стратегического образа системы, позволяющей выполнять бизнес-задачи.

Предположение (assumption) — это утверждение, которое предполагается верным в отсутствие знаний или доказательств иного.

Спецификация требований к ПО (software requirements specification, SRS) — структурированный набор требований к программному обеспечению и его внешним интерфейсам. Включает в себя функциональность, производительность, конструктивные ограничения и атрибуты.

Используемые источники

1. Определение образа и границ проекта.
2. Документирование требований.
3. Описание разделов шаблона спецификации требований.