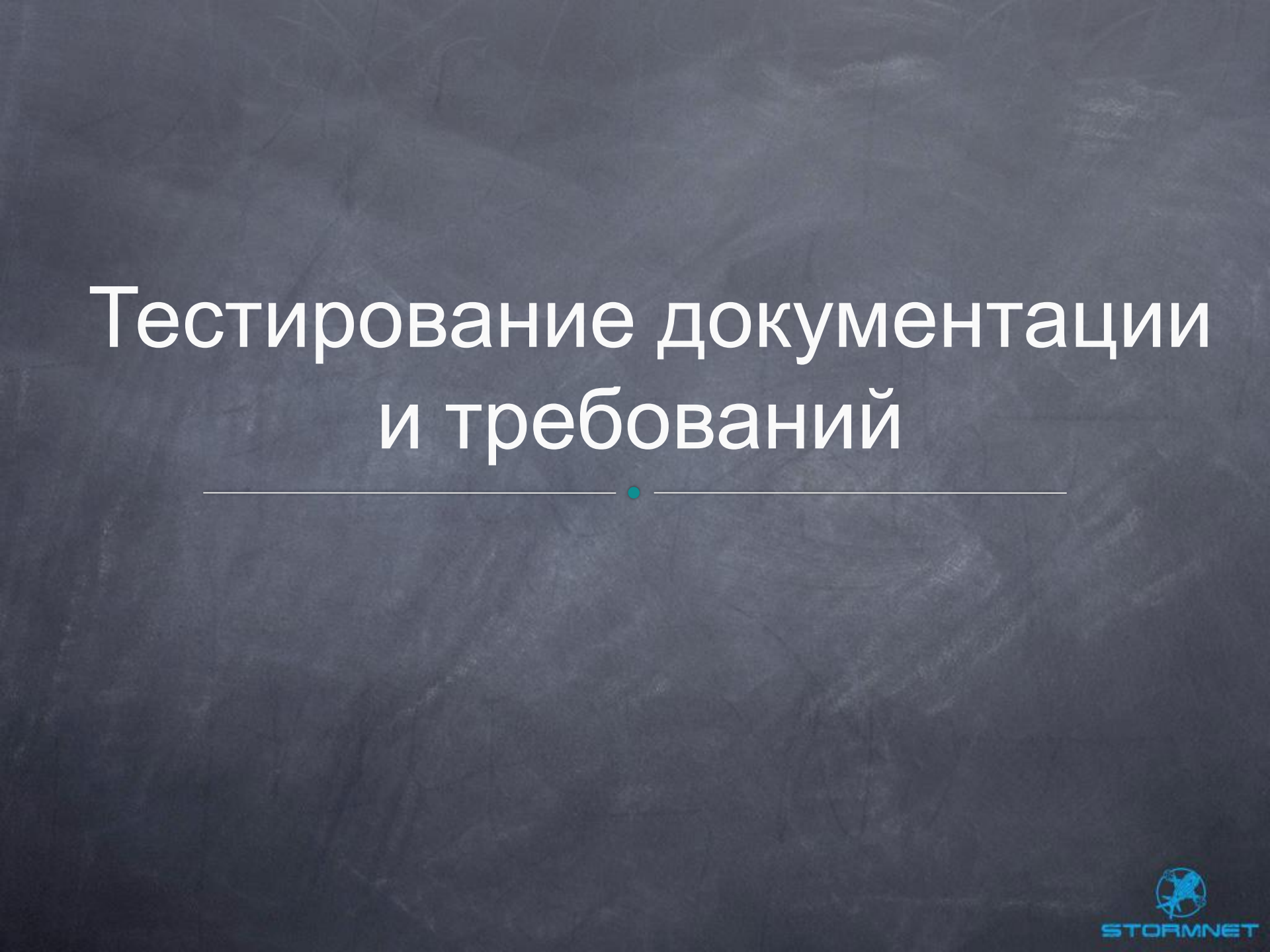
Тестирование документации и требований



Требования

- **Требования (requirements)** – это подробное изложение функционального наполнения системы. Требования должны быть независимы от внутренней архитектуры приложения, т.е. должны описывать то, что необходимо заказчику без деталей реализации (принцип «what, not how»). Правда, бывают исключения в виде ограничений, например, на операционную систему.

Определение

- Необходимо также обратить внимание на следующие определения понятия “требование” (на основе работ Вигерса и стандарта IEEE Standard Glossary of Software Engineering Terminology, 1990):
  1. Условие или возможность, требуемая пользователем для решения задач или достижения целей.
  2. Условие или возможность, которые должны удовлетворяться системой/компонентом системы или которыми система/компонент системы должна обладать для обеспечения условий контракта, стандартов, спецификаций или др. регулирующими документами.
  3. Документальная репрезентация (зафиксированное определение, описание) условий или возможностей, перечисленных в предыдущих пунктах.

Требования к продукту и процессу

- Проводится разграничение соответствующих требований как *свойств продукта*, который необходимо получить, и *процесса*, с помощью которого продукт будет создаваться. Отметим, что ряд требований может быть заложен неявно и программные требования могут порождать требования к процессу, например: работа в режиме 24×7 (как бизнес-требование) наверняка приведёт к ограничению выбора тех или иных программных средств, платформ развёртывания и архитектурных решений.

Требования к продукту

- В своей основе требования — это то, что формулирует заказчик. Цель, которую он преследует — получить хороший конечный продукт: функциональный и удобный в использовании. Поэтому требования к продукту являются основополагающим классом требований.

Требования к процессу

- Вопросы формулирования требований к процессу, т.е. к тому, как разработчик будет выполнять работы по созданию целевой системы, казалось бы, не лежат в компетенции заказчика. Без регламентации процесса заказчиком легко можно было бы обойтись, если бы все проекты всегда выполнялись точно и в срок. Однако, к сожалению, статистика говорит об обратном. Заказчик, вступая в договорные отношения с разработчиком, несёт различные риски, главными из которых является риск получить продукт с опозданием, либо ненадлежащего качества. Основные мероприятия по контролю и снижению риска — регламентация процесса создания программного обеспечения и его аудит.

Требования к процессу

- Насколько подробно заказчику следует регламентировать требования к процессу – вопрос, на который сложно ответить однозначно. Ответ на него зависит от множества факторов, таких, как
  - ценность конечного продукта для заказчика,
  - степень доверия заказчика к разработчику,
  - сумма подписанного контракта,
  - увязка срока сдачи продукта в эксплуатацию с бизнес-планами заказчика и т.д.

  Однако, со всей определенностью можно сказать следующее:
  - Регламентация процесса заказчиком позволяет снизить его риски.
  - Мероприятия заказчика по регламентации процесса приводят к дополнительным накладным расходам. Требуется найти разумный компромисс между *степенью контроля рисков* и величиной расходов.
- В качестве требований к проекту может быть внесен регламент отчётов разработчика, совместных семинаров по оценке промежуточных результатов, определены характеристики компетенций участников рабочей группы, их количество, указана методология управления проектом.

Пример

- Рассмотрим пример формулировки требования к оффшорному проекту (заказчик и разработчик физически находятся в разных государствах). В этой ситуации заказчику требуется особый контроль над процессом выполнения проекта.
  - Разработчик представляет заказчику согласованный план работ с детализацией с точностью до конкретных исполнителей.
  - Разработчик осуществляет ежедневные сборки, регрессионное тестирование частей разрабатываемого продукта и тестирование продукта в целом (системное тестирование).
  - Все управленческие и проектные артефакты, исходные коды и тестовые примеры размещаются в режиме реального времени в интегрированной среде разработки с возможностью для заказчика осуществления мониторинга на базе web-технологий.

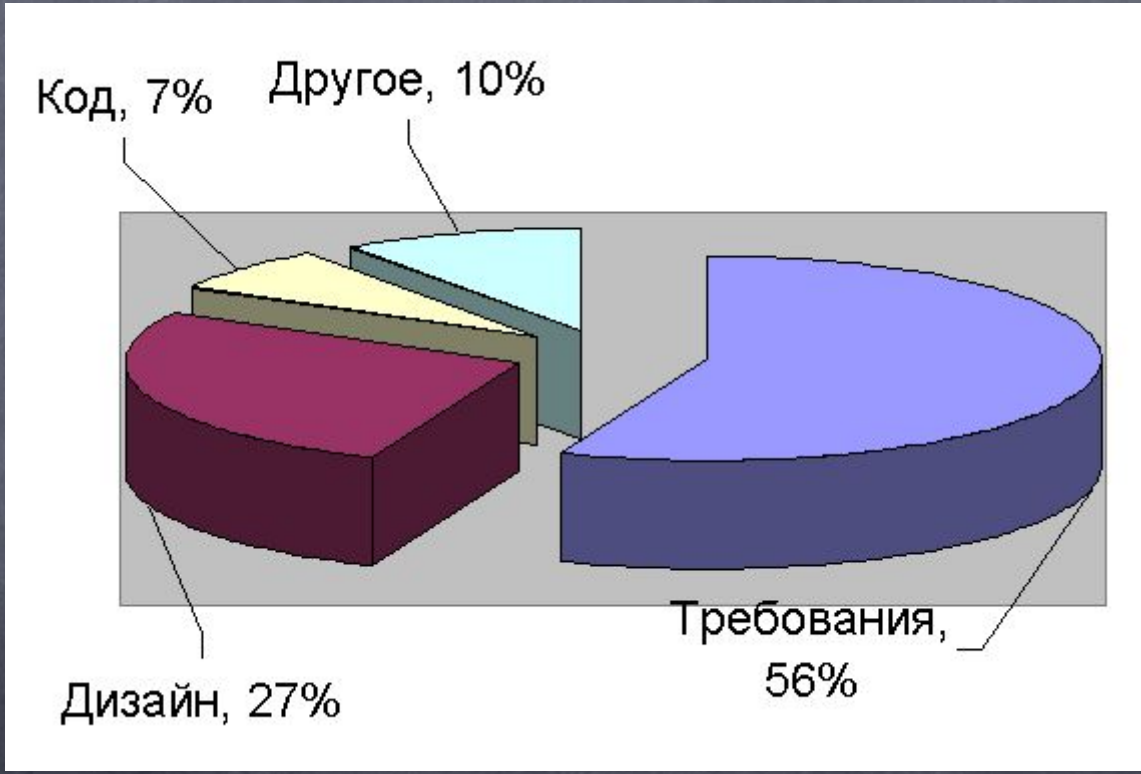
Важность требований

- Почему же так важны требования? Если взглянуть на рисунок, видно, что на каждом следующем шаге процесса разработки продукта стоимость обнаружения и устранения дефекта повышается в разы. Т.е. обнаружение максимального числа ошибок в требованиях поможет избежать лишней траты времени и средств в дальнейшем.

Важность требований

- Зачем же тестировщики нужны при анализе требований?
- А именно в требованиях закрадывается больше всего багов, а не в коде, как думают многие.

Важность требований



Виды документации, подвергаемой тестированию

Проектную документацию (project documentation):

- Требования к программному продукту (product requirements).
- Функциональные спецификации к программному продукту (functional specifications).
- Архитектуру (architecture) и дизайн (design).
- План проекта (project plan) и тестовый план (test plan).
- Тестовые случаи и сценарии (test cases, test scenarios).

Сопроводительную документацию (и документацию для пользователей):

- Интерактивную помощь (on-line help).
- Руководства по установке (Installation guide) и использованию программного продукта (user manual).

Уровни требований

- Обычно выделяют три уровня требований:
  1. На верхнем уровне представлены так называемые бизнес-требования (business requirements).
  Пример бизнес-требования: система должна сократить срок оборачиваемости обрабатываемых на предприятии заказов в три раза. Бизнес-требования обычно формулируются топ-менеджерами, либо акционерами предприятия.
  2. Следующий уровень – уровень требований пользователей (user requirements).
  Пример требования пользователя: система должна представлять диалоговые средства для ввода исчерпывающей информации о заказе, последующей фиксации информации в базе данных и маршрутизации информации о заказе к сотруднику, отвечающему за его планирование и исполнение.
- Требования пользователей часто бывают плохо структурированными, дублирующимися, противоречивыми. Поэтому для создания системы важен третий уровень, в котором осуществляется формализация требований.
  3. Третий уровень – функциональный (functional requirements).
  Пример функциональных требований (или просто функций) по работе с электронным заказом: заказ может быть создан, отредактирован, удалён и перемещён с участка на участок.
- Функциональные требования определяют «что система должна делать», нефункциональные – «как система это должна делать? С соблюдением каких условий» (например, скорость отклика при выполнении заданной операции).

Типы требований

- Различают следующие типы требований:
  - функциональные
  - нефункциональные
  - системные

Функциональные требования

- Для пользователя важно, чтобы система выполняла определённые действия, при этом пользователь некоторым образом взаимодействует с системой, использует её для своих целей. Таким образом, если определить все возможные варианты использования системы пользователями или другими внешними процессами, то мы получим функциональные требования к ней.
- Однако каждый конкретный пользователь работает с системой по-своему, поэтому для определения действительных вариантов использования системы необходимо досконально знать потребности всех заинтересованных пользователей, провести анализ полученной информации и систематизировать её в действительные варианты использования системы, т.е. абстрагироваться от конкретных пользователей и исходить от бизнес-задач. В дополнение к вариантам использования необходимо определить, как должен выглядеть пользовательский интерфейс для реализации того или иного варианта использования. Набросать эскизы, обсудить их с пользователями, создать прототипы и отдать их на тестирование пользователям.

Группа функциональных требований

- *Бизнес-требования* (business requirements) – определяют высокоуровневые цели организации или клиента (потребителя) – заказчика разрабатываемого программного обеспечения.
- *Пользовательские требования* (user requirements) – описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы.
- *Функциональные требования* (functional requirements) – определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требований и в контексте пользовательских требований.

Нефункциональные требования

- К нефункциональным требованиям относятся такие свойства системы, как:
  - производительность,
  - зависимость от платформы,
  - расширяемость,
  - надёжность и т.д.
- Под надёжностью понимаются такие характеристики, как пригодность, точность, средняя наработка на отказ и т.п.
- Требования по производительности – это скорость, пропускная способность, время отклика, используемая память. Многие требования, связанные с производительностью должны быть описаны в конкретных вариантах использования, а не в разделе относящейся ко всей системе.
- Также можно отметить, что часто нефункциональные требования *не могут быть привязаны* к конкретному варианту использования и должны быть вынесены в отдельный список дополнительных требований к системе.

Группа нефункциональных требований

- **Бизнес-правила** (business rules) – включают или связаны с корпоративными регламентами, политиками, стандартами, законодательными актами, внутрикорпоративными инициативами (например, стремление достичь зрелости процессов по CMMI 4-го уровня), учётными практиками, алгоритмами вычислений и т.д. В контексте управления проектами такие правила обладают высокой значимостью и, именно они, определяют ограничения использования бизнес-проектов, для автоматизации которых создается соответствующее программное обеспечение.
- ***Внешние интерфейсы*** (external interfaces) – часто подменяются «пользовательским интерфейсом». На самом деле вопросы организации пользовательского интерфейса безусловно важны в данной категории требований, однако, конкретизация аспектов взаимодействия с другими системами, операционной средой (например, запись в журнал событий операционной системы), возможностями мониторинга при эксплуатации – все это не столько функциональные требования, сколько вопросы интерфейсов, так как функциональные требования связаны непосредственно с функциональностью системы, направленной на решение бизнес-потребностей.
- ***Атрибуты качества*** (quality attributes) – описывают дополнительные характеристики продукта в различных измерениях, важных для пользователей и/или разработчиков. Атрибуты касаются вопросов портируемости, интер-операбельности (прозрачности взаимодействия с другими системами), целостности, устойчивости и т.п.
- ***Ограничения*** (constraints) – формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации. В частности, к ним могут относиться параметры производительности, влияющие на выбор платформы реализации и/или развёртывания (протоколы, серверы приложений, баз данных, …), которые, в свою очередь, могут относиться, например, к внешним интерфейсам.

Другие требования

- Помимо рассмотренного выше также отметим, что к требованиям относятся такие группы как:
- ***Потребности*** (needs) – отражают проблемы бизнеса, персоналии или процесса, которые должны быть соотнесены с использованием или приобретением системы.
- ***Системные требования*** (system requirements) – иногда классифицируются как составная часть группы функциональных требований (не путайте с как таковыми «функциональными требованиями»). Описывают высокоуровневые требования к программному обеспечению, содержащему несколько взаимосвязанных подсистем и приложений. При этом, система может быть как целиком программной, так и состоять из программной и аппаратной частей. В общем случае, частью системы может быть персонал, выполняющий определенные функции системы, например, авторизацию выполнения определенных операций с использованием программно-аппаратных подсистем.
- ***Независимые свойства*** (emergent properties) – требования, которые не могут быть адресованы тому или иному компоненту программной системы, но которые должны быть соблюдены, например, в контексте сетевой инфраструктуры или регламентов работы пользователей.
- ***Требования с количественной оценкой*** (quantifiable requirements) – требования, поддающиеся количественному определению/измерению, например, система должна обеспечить пропускную способность «столько-то запросов в секунду».

Уровни требований

- **Уровень бизнес-требований**: «общее видение» (обзорная документация)
- **Уровень пользовательских требований**: «что можно будет делать» (варианты использования)
- **Уровень функциональных и нефункциональных требований**: «что конкретно должна выполнять система и как она должна это выполнять» (набор требований)

Кто создаёт и использует требования

- Представитель заказчика – постановка задачи, определение рамок проекта, контроль работы исполнителя, приемка результатов работы;
- Архитектор системы – разработка архитектуры, проектирование подсистем
- Программист – разработка программного кода
- Тестировщик – составление тест-плана, тестовых сценариев
- Менеджер проекта – планирование и контроль исполнения работ

Источники требований

- Федеральное и муниципальное отраслевое законодательство (конституция, законы, распоряжения)
- Нормативное обеспечение организации (регламенты, положения, уставы, приказы)
- Текущая организация деятельности объекта автоматизации
- Модели деятельности (диаграммы бизнес-процессов)
- Представления и ожидания потребителей и пользователей системы
- Журналы использования существующих программно-аппаратных систем
- Конкурирующие программные продукты

Источники требований

- Более конкретно:
  - Соображения, высказанные представителем заказчика (сотрудники аналитической группы исполнителя, внешних экспертов и т.д.)
  - Артефакты, описывающие предметную область
  - «Лучшие практики» («best practices»)

Пути выявления требований

- интервью (заказчик)
- анкетирование (заказчик, пользователи)
- наблюдение (за целевыми пользователями)
- самостоятельное описание
- семинары (заказчик, пользователи)
- прототипирование

Видение продукта

- Когда мы говорим о «видении», «концепции», «образе» продукта мы имеем в виду «Какой должна быть система?»
  - Нужно «увидеть», как программный продукт впишется в организационные процессы предприятия
  - Какие ключевые выгоды он даст
  - Какие проблемы позволит разрешить
- Обсуждаются:
  - Высокоуровневые требования (возможности, свойства) продукта
  - Наиболее существенные ограничения

Границы проекта

- Когда речь идет о «рамках», «границах», «контексте» проекта выдвигаются следующие вопросы для обсуждения:
  - Границы системы и среды
  - Требуемые ресурсы на создание системы (бюджет, календарное планирование, подбор персонала)
  - Сроки (ключевые даты)

Документирование требований. Секции документа “Требования” на основе IEEE Standard 830-1998

1. Введение
   - 1.1 Назначение документа.
   - 1.2. Поддерживаемые соглашения.
   - 1.3. Предполагаемая аудитория и рекомендации по последовательности работы с документом для каждого класса читателей.
   - 1.4. Границы проекта. Здесь содержится ссылка на документ "Концепция", если таковой имеется, либо краткое резюме продукта.
   - 1.5. Ссылки.

Документирование требований. Секции документа “Требования” на основе IEEE Standard 830-1998

2. Общее описание.

2.1. Общий взгляд на продукт. Здесь необходимо определить – является ли описываемый продукт новым членом растущего семейства продуктов, новой версией существующей системы, заменой существующего приложения или совершенно новым продуктом. Если спецификация требований определяет компонент более крупной системы, нужно указать, как это ПО соотносится со всей системой и определить основные интерфейсы между ними.

2.2. Особенности продукта. Перечисляются ключевые особенности продукта или его главные свойства. Здесь уместно поместить контекстную диаграмму (в виде диаграммы вариантов использования, потоков данных или др. спецификаций).

2.3. Классы и характеристики пользователей. Документируется процесс поиска акторов, в котором выявляются все пользователи системы и осуществляется обобщение (выделение классов) пользователей. Найденные классы описываются (например – уровень квалификации, доступный функционал и т.д.).

2.4. Операционная среда. Рассматривается среда функционирования АИС, включая аппаратные средства, операционные системы, для распределенных систем – географическое расположение пользователей и серверов, топология сети.

Документирование требований. Секции документа “Требования” на основе IEEE Standard 830-1998

2.5. Ограничения проектирования и реализации. Рассмотрим классификацию ограничений:

- определенные технологии, средства, языки программирования и базы данных, которые следует использовать или избегать;
- ограничения, налагаемые операционной средой продукта;
- обязательные соглашения или стандарты разработки;
- обратная совместимость с продуктами, выпущенными ранее;
- ограничения, налагаемые бизнес-правилами;
- ограничения, связанные с оборудованием, например требования к быстродействию, ограничения памяти или процессора;
- соглашения, связанные с пользовательским интерфейсом существующего продукта, которые необходимо соблюдать при его улучшении
- форматы и протоколы обмена данными.

2.6 Документация для пользователей.

2.7 Предположения и зависимости

Документирование требований. Секции документа “Требования” на основе IEEE Standard 830-1998

3. Функции системы

Для каждой i-й функции составляется следующее описание.

3.i Наименование i-й функции системы.

3.i.1 Описание и приоритеты. Приводится краткое описание функции и указывается ее приоритет (степень важности/очередности реализации).

3.i.2 Последовательности "воздействие - реакция". Необходимо перечислить последовательность воздействий, оказываемых на систему (действия пользователей, сигналы внешних устройств и др.), и отклики системы, определяющие реакцию конкретной функции.

3.i.3 Функциональные требования. Необходимо дать детализацию i-й функции, перечислить детализированные функциональные требования, включая реакцию на ожидаемые ошибки и неверные действия. Каждому детальному функциональному требованию присваивается уникальный идентификатор.

Документирование требований. Секции документа “Требования” на основе IEEE Standard 830-1998

4. Требования к внешнему интерфейсу

Ниже рассмотрены конкретные рекомендации по написанию разделов этого параграфа, согласно:

4.1 Интерфейсы пользователя

Основные характеристики UI:

ссылки на стандарты графического интерфейса пользователей или стилевые рекомендации для семейства продукта, которые необходимо соблюдать;

стандарты шрифтов, значков, названий кнопок, изображений, цветовых схем, последовательностей полей вкладок, часто используемых элементов управления и т. п.;

конфигурация экрана или ограничения разрешения;

стандартные кнопки, функции или ссылки перемещения, одинаковые для всех экранов, например кнопка справки;

быстрые клавиши;

стандарты отображения сообщений;

стандарты конфигурации для упрощения локализации ПО;

специальные возможности для пользователей с проблемами со зрением.

Документирование требований. Секции документа “Требования” на основе IEEE Standard 830-1998

4.2 Интерфейсы оборудования

Описываются характеристики каждого интерфейса между компонентами ПО и оборудования системы. В описание могут входить типы поддерживаемых устройств, взаимодействия данных и элементов управлений между ПО и оборудованием, а также протоколы взаимодействия, которые будут использоваться.

4.3 Интерфейсы ПО

Описывается соединения продукта и других компонентов ПО (идентифицированные по имени и версии), в том числе базы данных, операционные системы, средства, библиотеки и интегрированные коммерческие компоненты. Указывается назначение элементов сообщений, данных и элементов управления, обмен которыми происходит между компонентами ПО. Описываются службы, необходимые внешним компонентам ПО, и природа взаимодействия между компонентами. Определяются данные, к которым будут иметь доступ компоненты ПО.

4.4 Интерфейсы передачи информации

Указываются требования для любых функций взаимодействия, которые будут использоваться продуктом, включая электронную почту, Web-браузер, протоколы сетевого соединения и электронные формы. Определяются соответствующие форматы сообщений. Описываются особенности безопасности взаимодействия или шифрования, частоты передачи данных и механизмов синхронизации.

Документирование требований. Секции документа “Требования” на основе IEEE Standard 830-1998

5. Другие нефункциональные требования

В этом разделе описываются остальные нефункциональные требования, не относящиеся к требованиям к интерфейсу, которые представлены в разделе 4, и к ограничениям, описываемым в разделе 2.5.

5.1 Требования к производительности

Указываются специальные требования к производительности для различных системных операций. Обосновывается их необходимость для того, чтобы помочь разработчикам принять правильные решения, касающиеся дизайна.

Приложение А. Словарь терминов (глоссарий).

Приложение Б. Модели анализа. В этот раздел помещаются все модели, построенные в процессе анализа требований.

Приложение В. Список вопросов.

Это динамический список еще не разрешенных проблем, связанных с требованиями. Это могут быть элементы, помеченные как "TBD" (To Be Determined – необходимо определить), отложенные решения, необходимая информация, неразрешенные конфликты и т.п.

Тестирование требований

- Важность тестирования требований состоит в том, что хорошие требования позволяют:
  - Достичь общего понимания между заказчиком и разработчиком
  - Определить рамки проекта
  - Более точно определить финансовые и временные характеристики проекта.
  - Обезопасить заказчика от риска получить продукт, в котором он не сможет работать
  - Обезопасить разработчика от риска попасть в ситуацию «неконтролируемого размытия границ», которое может привести к непредвиденным затратам ресурсов сверх начальных ожиданий.

Характеристики хорошего требования

- **Каждое требование должно быть:**
- **Завершённым** (complete). Все важные аспекты должны быть включены. Ничто не должно быть оставлено «для будущего определения» (TBD – to be defined).
- **Непротиворечивым** (consistent). Требование не должно содержать противоречий как внутри себя, так и с другими требованиями.
- **Корректным** (correct). Требование должно чётко указывать на то, что должно выполнять приложение. Недопустимо при написании требования предполагать, что что-то окажется очевидным. Каждый человек понимает это «очевидное» по-своему, и в итоге система получится не такой, как задумывалось.
- **Недвусмысленным** (unambiguous). Требование не должно допускать разночтений.
- **Проверяемым** (verifiable). Требование должно быть сформулировано так, чтобы существовали способы однозначной проверки – выполнено

Характеристики хорошего набора требований

- **Наборы требований должны быть:**
- **Модифицируемыми (modifiable).** Структура и стиль набора требований должны быть такими, чтобы набор требований можно было легко модифицировать. Должна отсутствовать избыточность. Должно быть построено корректное содержание всего документа.
- **Прослеживаемыми (traceable).** У каждого требования должен быть уникальный идентификатор, по которому на это требование можно сослаться.
- **Проранжированными по важности, стабильности и срочности (ranked for importance, stability and priority).** Для каждого требования должен быть указан уровень его важности (насколько оно важно для заказчика), стабильности (насколько высока вероятность, что это требование ещё будет изменено в процессе обсуждения деталей проекта) и срочности (как быстро требование должно быть реализовано).

Проблемы с требованиями

- Проблема незавершенности (неполноты)
- Проблема «To Be Defined»
- Проблема противоречивости
- Проблема некорректности
- Проблема двусмысленности
- Проблема непроверяемости
- Проблема немодифицируемости
- Проблема непрослеживаемости
- Проблема непроранжированности

Проблема незавершённости (неполноты)

- Хорошо, когда вся важная информация присутствует. Только вот вопрос – а как можно найти то, чего нет, но должно быть? Как догадаться, что что-то отсутствует?
- Следует обратить внимание на такие типичные случаи.
- **Отсутствуют нефункциональные требования или нефункциональные составляющие требования.** Мы знаем, что система должна делать. А про то, как (как быстро, как безопасно) в лучшем случае узнаём только в самом конце.
- Итак, мы уточнили какие-то требования, и заказчик высказал свои предпочтения. Например, ему нужно, чтобы приложение работало «надёжно и быстро».
- Думаем, как мы сможем проверить эти требования? Проверить, что приложение работает быстро. А как проверить? И что проверять?
- **Вывод:** кроме самого нефункционального требования, нам обязательно нужны критерии – как это требование проверить (и измерить). Чем важнее требование для заказчика, тем точнее должны быть эти критерии.

Проблема незавершённости (неполноты)

Рекомендуется задавать общие вопросы.

Их преимущества:

- Они универсальные.
- Они не навязывают решение.
- Они не загоняют заказчика в ситуацию, когда ему приходится выбирать из имеющихся вариантов.

Хорошая аналогия – вопросы репортера (или маленького ребёнка): «А что?», «А зачем?», «А почему?»

Проблема «To Be Defined»

- Ещё одной проблемой чрезмерной общности утверждений является т.н. TBD («to be defined», «будет определено»).
- Мы можем успокоиться (на время) , только если знаем, кто и когда должен определить эти TBD. И почему они не определены сейчас.
- Также бывает, что стороны, ответственные за закрытие TBD, просто забывают об этом. Вывод? Нужно напоминать.

Проблемы противоречивости

- Противоречия *внутри одного требования*.
- Противоречия *между двумя и более требованиями*.
- Противоречия *между таблицами и текстом*.
- Противоречия *между картинкой и текстом*.
- Противоречия *между требованием и прототипом*.

Проблемы некорректности

Ошибки могут быть вызваны:

- опечатками, последствиями «copy-paste»;
- остатками устаревших требований;
- наличием «озолочения» («gold plating»);
- наличием технически невыполнимых требований;
- наличием неаргументированных требований к дизайну и архитектуре.

Проблемы двусмысленности

- Если что-то можно понять несколькими способами, можно быть уверенным, что разные люди поймут это по-разному.
- Например. В требовании сказано, что «функциональность X» является опциональной.
- Что думает отдел разработки? «Чудесно. Опционально – значит необязательно. Можно не реализовывать.»
- Что думает отдел маркетинга? «Ага. Мы выпустим две версии продукта. В более дорогой будет эта функциональность.»
- Что думает заказчик? «Я за те же деньги получу ещё и вот эту функциональность».

- Вывод? Следует писать требования так, чтобы исключить возможность их двоякого понимания.

Проблемы непроверяемости

- Для начала следует отметить, что все вышеперечисленные проблемы ведут в том числе к тому, что мы не можем проверить, удовлетворяет ли продукт требованию.

- **Как понять, что требование непроверяемо? Попробовать придумать несколько тестов для его проверки. Если тесты не придумываются – вот она, проблема.**

- А бывает ли «просто непроверямое требование»? Да.
- Например: «В приложении должно быть ноль ошибок». «Приложение должно поддерживать все версии всех операционных систем».

Проблемы немодифицируемости

- После каждого обновления требований понадобится тратить недели чтобы выловить все появившиеся противоречия

Проблемы непрослеживаемости

- Если требования не пронумерованы, не имеют чёткого оглавления, не имеют работающих перекрёстными ссылок – это хаос. А в хаосе ошибки плодятся с удивительной скоростью.

Проблемы непроранжированности

- Если требования не проранжированы по важности, стабильности и срочности, мы рискуем уделить основное внимание не тому, что на самом деле важно для заказчика.

Вывод

Если мы не можем проверить требование – это проблема.

В конечном итоге именно тестировщики отвечают за то, чтобы требование было проверено.

Если мы не можем проверить требование по объективным причинам, мы уточняем его до тех пор, пока оно не станет проверяемым. В зависимости от ситуации, мы расспрашиваем заказчика, разработчиков, своих более опытных коллег и т.д.

Работа с требованиями (техники и способы)

- Одна из наиболее распространённых техник работы с требованиями – взаимный перепросмотр.
- Суть взаимного перепросмотра требований проста: **после того, как один человек создал требование, другой человек это требование проверяет.**
- Обычно, выделяют три уровня перепросмотра:
- **Неформальный перепросмотр.** Двое коллег просто обмениваются листиками (файликами) и правят найденные ошибки, которые потом обсуждаются за чашкой чая или в любое другое относительно свободное время.
- **Технический перепросмотр.** Это немного более формализованный процесс, требующий подготовки, выделенного времени, участия некоторой группы специалистов (желательно, из различных областей).
- **Формальная инспекция.** Проводится редко и в случае очень больших проектов и крайней необходимости. Описывается

Работа с требованиями (техники и способы)

- Существует три простые техники работы с требованиями: задавать вопросы, писать тест кейсы и рисовать рисунки.
- Самый простой и не требующий большого опыта способ – **задавать как можно больше вопросов**. Получая разнообразные ответы от разных участников проекта (как наших коллег так и представителей заказчика), мы расширяем, углубляем и уточняем своё представление о том, что и как должно работать.
- **Второй способ – создавать тест-кейсы** (о которых мы поговорим позже). Когда вы видите требование, спросите себя: «Как я буду его тестировать? Какие тесты очевидно покажут, что это требование реализовано в ПС правильно?» Если с придумыванием таких тестов вы испытываете

Практическое задание

- Разработка требований и тестирование кружки