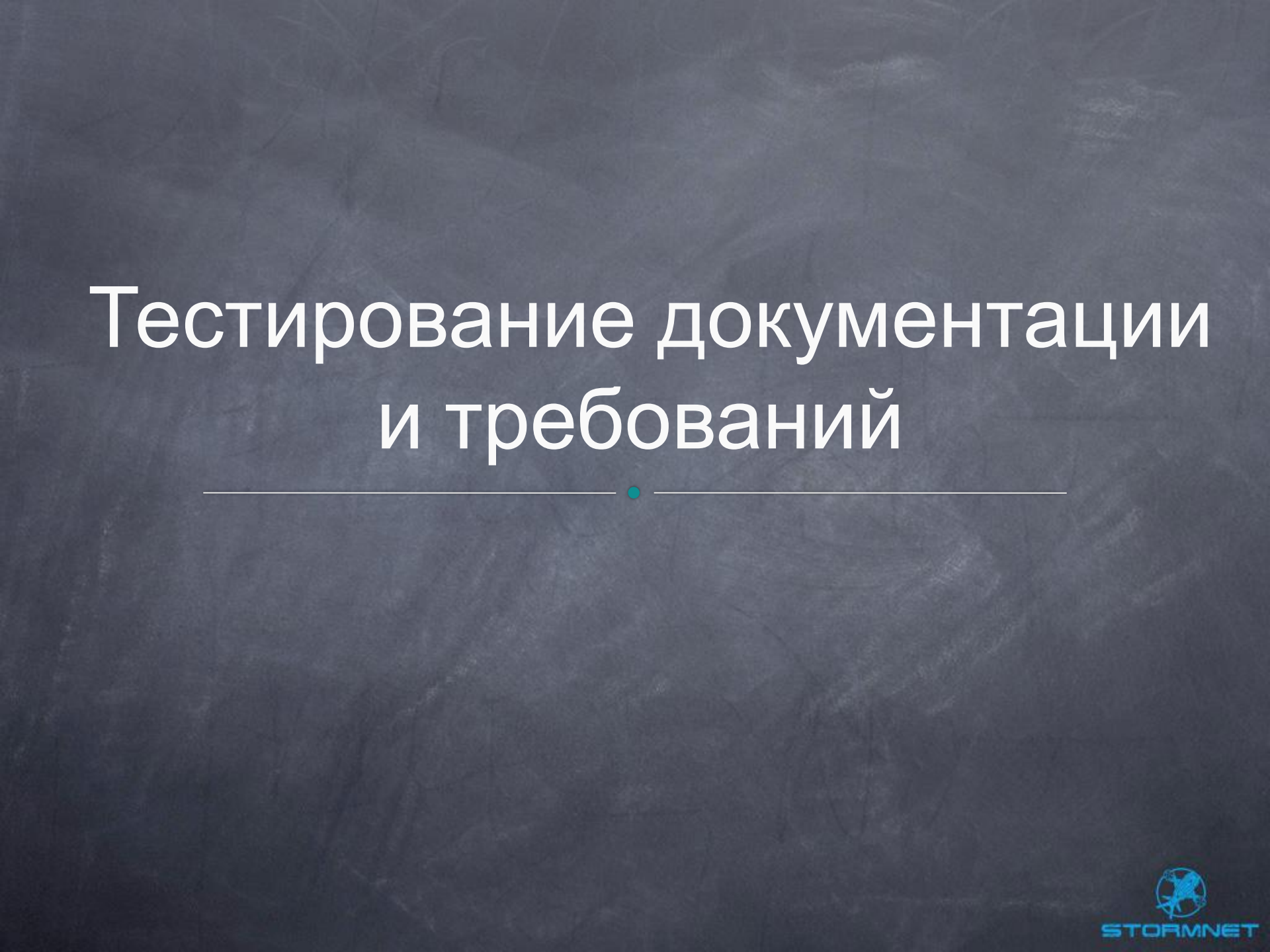
Тестирование документации и требований



Требования

- **Требования (requirements)** – описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной для пользователя задачи. 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.

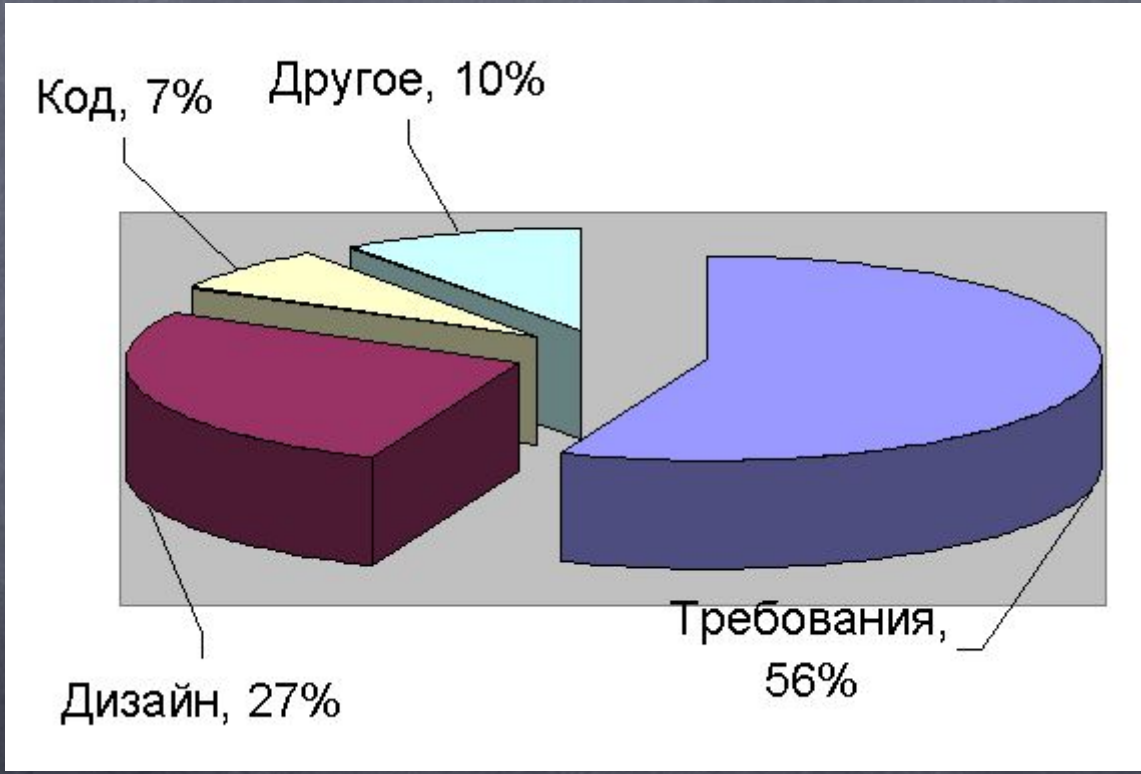
Важность требований

- 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.

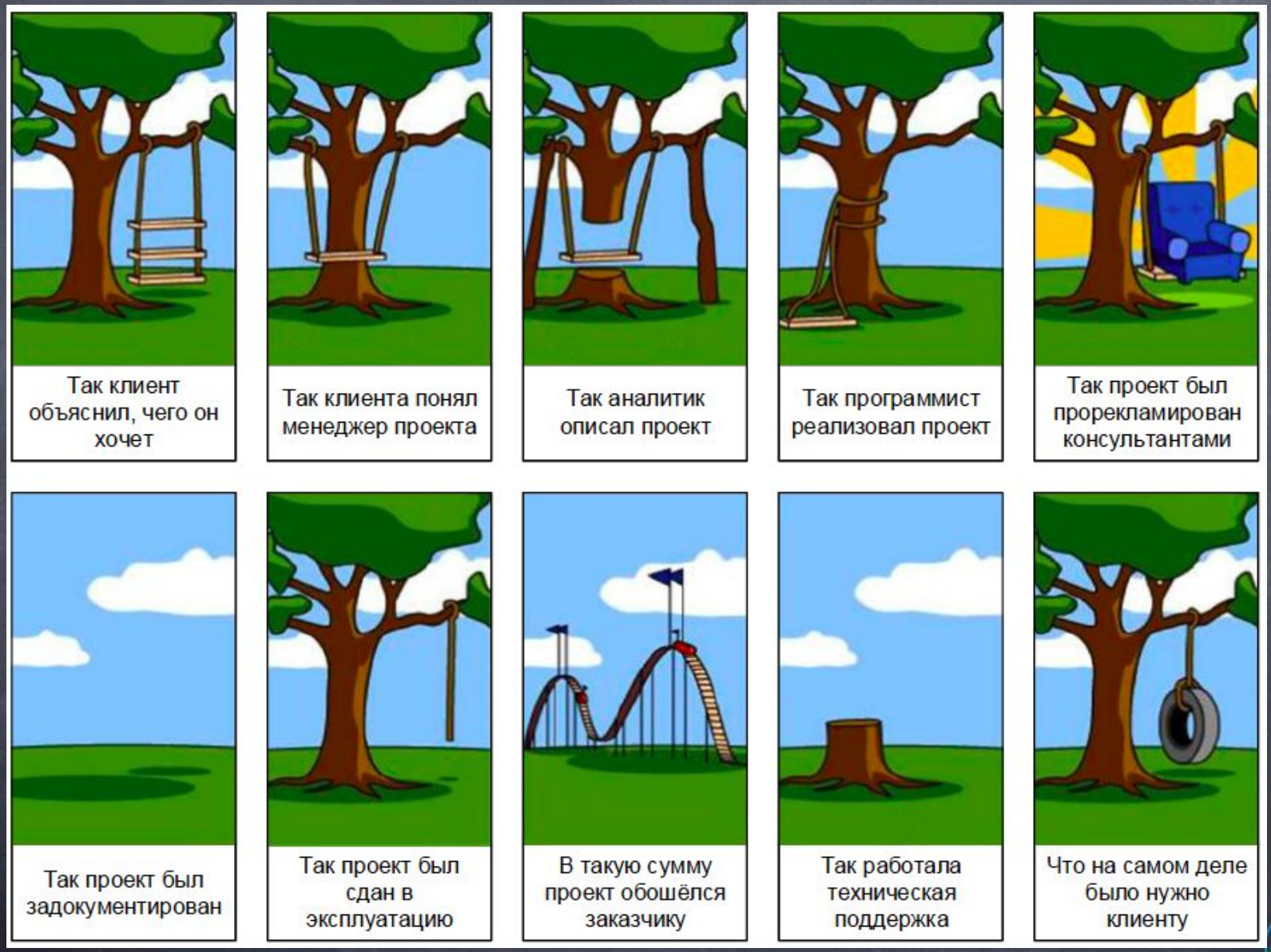
Важность требований



Важность требований

- Ещё одним аргументом в пользу тестирования требований является то, что, по разным оценкам, в них зарождается от ½ до ¾ всех проблем с программным обеспечением. В итоге есть риск, что получится так, как показано на рисунке на следующем слайде.

Типичный проект с плохими требованиями



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).
- Тест-кейсы (test cases).

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:
  - Бизнес-требования
  - Требования пользователей
  - Функциональные требования

Бизнес-требования

- 1. На верхнем уровне представлены так называемые бизнес-требования (business requirements), которые выражают цель, ради которой разрабатывается продукт (зачем вообще он нужен, какая от него ожидается польза).
  Пример бизнес-требования:
  - Система должна сократить срок оборачиваемости обрабатываемых на предприятии заказов в три раза. Бизнес-требования обычно формулируются топ-менеджерами, либо акционерами предприятия.
  - Нужен инструмент, в реальном времени отображающий наиболее выгодный курс покупки и продажи валюты.
  - Необходимо в два-три раза повысить количество заявок, обрабатываемых одним оператором за смену.
  - Нужно автоматизировать процесс выписки товарно-транспортных накладных на основе договоров.

Требования пользователей

- 2. Следующий уровень — уровень требований пользователей (user requirements). Эти требования описывают задачи, которые пользователь может выполнять с помощью разрабатываемой системы (реакцию системы на действия пользователя, сценарии работы пользователя).
- **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.
  - При первом входе пользователя в систему должно отображаться лицензионное соглашение.
  - Администратор должен иметь возможность просматривать список всех пользователей, работающих в данный момент в системе.
  - При первом сохранении новой статьи система должна выдавать запрос на сохранение в виде черновика или публикацию.
- Требования пользователей часто бывают плохо структурированными, дублирующимися, противоречивыми. Поэтому для создания системы важен третий уровень, в котором осуществляется формализация требований.

Функциональные требования

- 3. Третий уровень — функциональные требования (functional requirements) описывают поведение системы, т.е. её действия (вычисления, преобразования, проверки, обработку и т.д.).
- **Пример функциональных требований** (или просто функций)
  - По работе с электронным заказом: Заказ может быть создан, отредактирован, удалён и перемещён с участка на участок.
  - В процессе инсталляции приложение должно проверять остаток свободного места на целевом носителе.
  - Система должна автоматически выполнять резервное копирование данных ежедневно в указанный момент времени.
  - Электронный адрес пользователя, вводимый при регистрации, должен быть проверен на соответствие требованиям
- 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
  - другие

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)** –описывают особенности принятых в предметной области (и/или непосредственно у заказчика) процессов, ограничений и иных правил. Эти правила могут относиться к бизнес-процессам, корпоративным регламентам, политикам, стандартам, законодательным актам, правилам работы сотрудников, нюансам работы ПО и т.д.

Несколько простых, изолированных от контекста и друг от друга примеров бизнес-правил:

- Никакой документ, просмотренный посетителями сайта хотя бы один раз, не может быть отредактирован или удалён.
- Публикация статьи возможна только после утверждения главным редактором.
- Подключение к системе извне офиса запрещено в нерабочее время

Группа нефункциональных требований

- **Атрибуты качества** расширяют собой нефункциональные требования и на уровне пользовательских требований могут быть представлены в виде описания ключевых для проекта показателей качества (свойств продукта, не связанных с функциональностью, но являющихся важными для достижения целей создания продукта — производительность, масштабируемость, восстанавливаемость). Атрибутов качества очень много, но для любого проекта реально важными является лишь некоторое их подмножество.
- Несколько простых, изолированных от контекста и друг от друга примеров атрибутов качества:
  - Максимальное время готовности системы к выполнению новой команды после отмены предыдущей не может превышать одну секунду.
  - Внесённые в текст статьи изменения не должны быть утеряны при нарушении соединения между клиентом и сервером.

Группа нефункциональных требований

- **Требования к интерфейсам** описывают особенности взаимодействия разрабатываемой системы с другими системами и операционной средой.
- Несколько простых, изолированных от контекста и друг от друга примеров требований к интерфейсам:
  - Обмен данными между клиентской и серверной частями приложения должен быть реализован в формате JSON.
  - Протоколирование событий должно вестись в журнале событий операционной системы.

Группа нефункциональных требований

- **Ограничения** представляют собой факторы, ограничивающие выбор способов и средств (в том числе инструментов) реализации продукта.
- Несколько простых, изолированных от контекста и друг от друга примеров ограничений:
  - Все элементы интерфейса должны отображаться без прокрутки при разрешениях экрана от 800x600 до 1920x1080.
  - Не допускается использование Flash при реализации клиентской части приложения.

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.
- ***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)
- 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
- Работа с фокусными группами
- Анкетирование
- Семинары и мозговой штурм
- Наблюдение (за целевыми пользователями)
- Прототипирование
- Анализ документов
- Самостоятельное описание

Интервью

- Самый универсальный путь выявления требований, заключающийся в общении проектного специалиста (как правило, специалиста по бизнес-анализу) и представителя заказчика (или эксперта, пользователя и т.д.) Интервью может проходить в классическом понимании этого слова (беседа в виде «вопрос-ответ»), в виде переписки и т.п. Главным здесь является то, что ключевыми фигурами выступают двое — интервьюируемый и интервьюер (хотя это и не исключает наличия «аудитории слушателей», например, в виде лиц, поставленных в копию переписки).

Работа с фокусными группами

- Может выступать как вариант «расширенного интервью», где источником информации является не одно лицо, а группа лиц (как правило, представляющих собой целевую аудиторию, и/или обладающих важной для проекта информацией, и/или уполномоченных принимать важные для проекта решения).

Анкетирование

- Этот вариант выявления требований вызывает много споров, т.к. при неверной реализации может привести к нулевому результату при объёмных затратах. В то же время при правильной организации анкетирование позволяет автоматически собрать и обработать огромное количество ответов от огромного количества респондентов. Ключевым фактором успеха является правильное составление анкеты, правильный выбор аудитории и правильное преподнесение анкеты.

Семинары и мозговой штурм

- Семинары позволяют группе людей очень быстро обменяться информацией (и наглядно продемонстрировать те или иные идеи), а также хорошо сочетаются с интервью, анкетированием, прототипированием и моделированием — в том числе для обсуждения результатов и формирования выводов и решений. Мозговой штурм может проводиться и как часть семинара, и как отдельный вид деятельности. Он позволяет за минимальное время сгенерировать большое количество идей, которые в дальнейшем можно не спеша рассмотреть с точки зрения их использования для развития проекта.

Наблюдение

- Может выражаться как в буквальном наблюдении за некими процессами, так и во включении проектного специалиста в эти процессы в качестве участника. С одной стороны, наблюдение позволяет увидеть то, о чём (по совершенно различным соображениям) могут умолчать интервьюируемые, анкетируемые и представители фокусных групп, но с другой — отнимает очень много времени и чаще всего позволяет увидеть лишь часть процессов.

Прототипирование

- Состоит в демонстрации и обсуждении промежуточных версий продукта (например, дизайн страниц сайта может быть сначала представлен в виде картинок, и лишь затем свёрстан). Это один из лучших путей поиска единого понимания и уточнения требований, однако он может привести к серьёзным дополнительным затратам при отсутствии специальных инструментов (позволяющих быстро создавать прототипы) и слишком раннем применении (когда требования ещё не стабильны, и высока вероятность создания прототипа, имеющего мало общего с тем, что хотел заказчик).

Анализ документов

- Хорошо работает тогда, когда эксперты в предметной области (временно) недоступны, а также в предметных областях, имеющих общепринятую устоявшуюся регламентирующую документацию. Также к этой технике относится и просто изучение документов, регламентирующих бизнес-процессы в предметной области заказчика или в конкретной организации, что позволяет приобрести необходимые для лучшего понимания сути проекта знания.

Самостоятельное описание

- Является не столько техникой выявления требований, сколько техникой их фиксации и формализации. Очень сложно (и даже нельзя!) пытаться самому «придумать требования за заказчика», но в спокойной обстановке можно самостоятельно обработать собранную информацию и аккуратно оформить её для дальнейшего обсуждения и уточнения.

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)

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** 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 сложность – это тревожный звоночек: скорее всего, в требовании есть проблемы

Практическое задание

- Разработка требований и тестирование кружки