Основы тестирования ПО

ТЕСТИРОВАНИЕ ДОКУМЕНТАЦИИ И ТРЕБО

Что такое требование?

Требование (requirement) — описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной для пользователя задачи.

Небольшое «историческое отступление»: если поискать определения требований в литературе 10-20-30-летней давности, то можно заметить, что изначально о пользователях, их задачах и полезных для них свойствах приложения в определении требования не было сказано. Пользователь выступал некоей абстрактной фигурой, не имеющей отношения к приложению. В настоящее время такой подход недопустим, т.к. он не только приводит к коммерческому провалу продукта на рынке, но и многократно повышает затраты на разработку и тестирование.

Важность требований

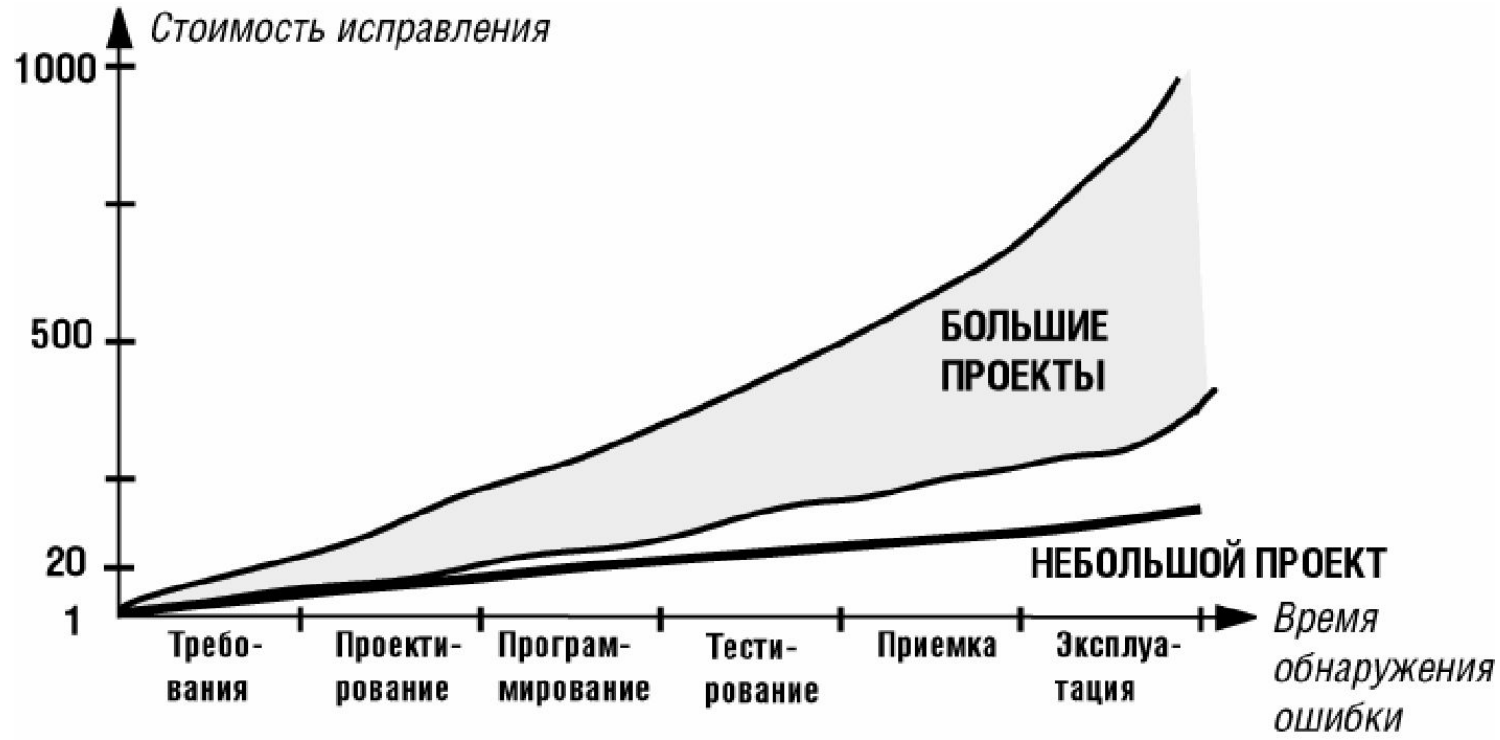
Требования являются отправной точкой для определения того, что проектная команда будет проектировать, реализовывать и тестировать.

Элементарная логика говорит нам, что если в требованиях что-то «не то», то и реализовано будет «не то», т.е. колоссальная работа множества людей будет выполнена впустую.

Брайан Хэнкс, описывая важность требований, подчёркивает, что они:

- Позволяют понять, что и с соблюдением каких условий система должна делать.
- Предоставляют возможность оценить масштаб изменений и управлять изменениями.
- Являются основой для формирования плана проекта (в том числе плана тестирования).
- Помогают предотвращать или разрешать конфликтные ситуации.
- Упрощают расстановку приоритетов в наборе задач.
- Позволяют объективно оценить степень прогресса в разработке проекта.

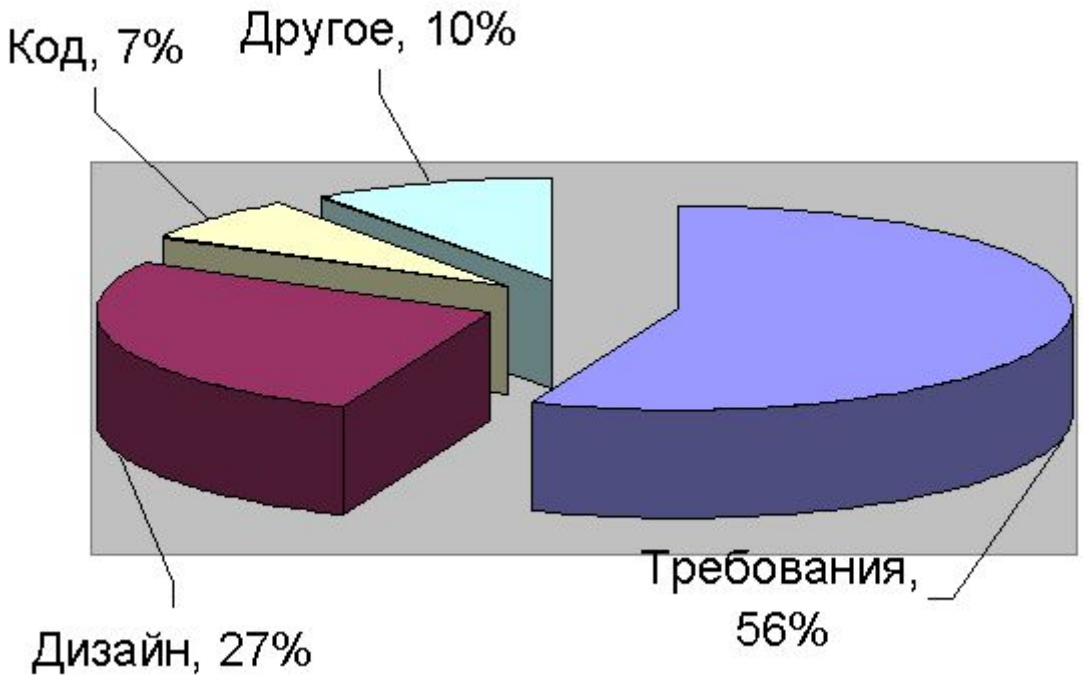
Стоимость исправления дефекта на разных стадиях развития проекта:



Именно в требованиях берёт начало большинство багов!

Получил от клиента письмо с правками в макет сайта: «То, чего здесь нет, оставляем так, как есть, но туда вносим правки».

© bash.im



Типичный проект с плохими требованиями

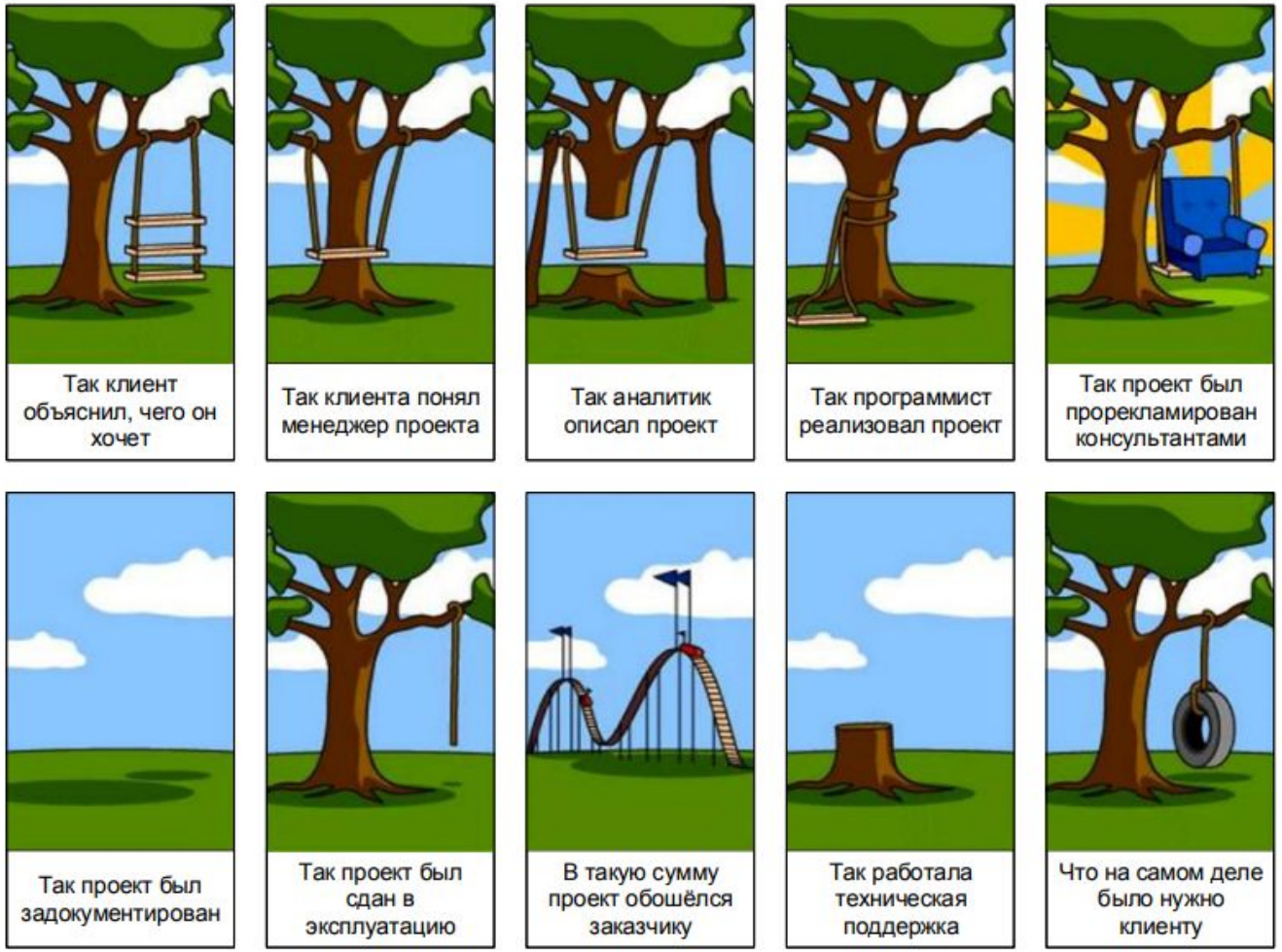


Рисунок 2.2.b — Типичный проект с плохими требованиями

Важность тестирования требований

Хорошие требования позволяют:

Выработать общее понимание между заказчиком и разработчиком

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»

Какую документацию надо тестировать

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).

Типы требований

- **Функциональные требования** (functional requirements):
  «**что** система должна делать».
- **Нефункциональные требования** (non-functional requirements):
  «**как** система должна это делать».

Требования, относящиеся не к функциональности, а к таким атрибутам как:

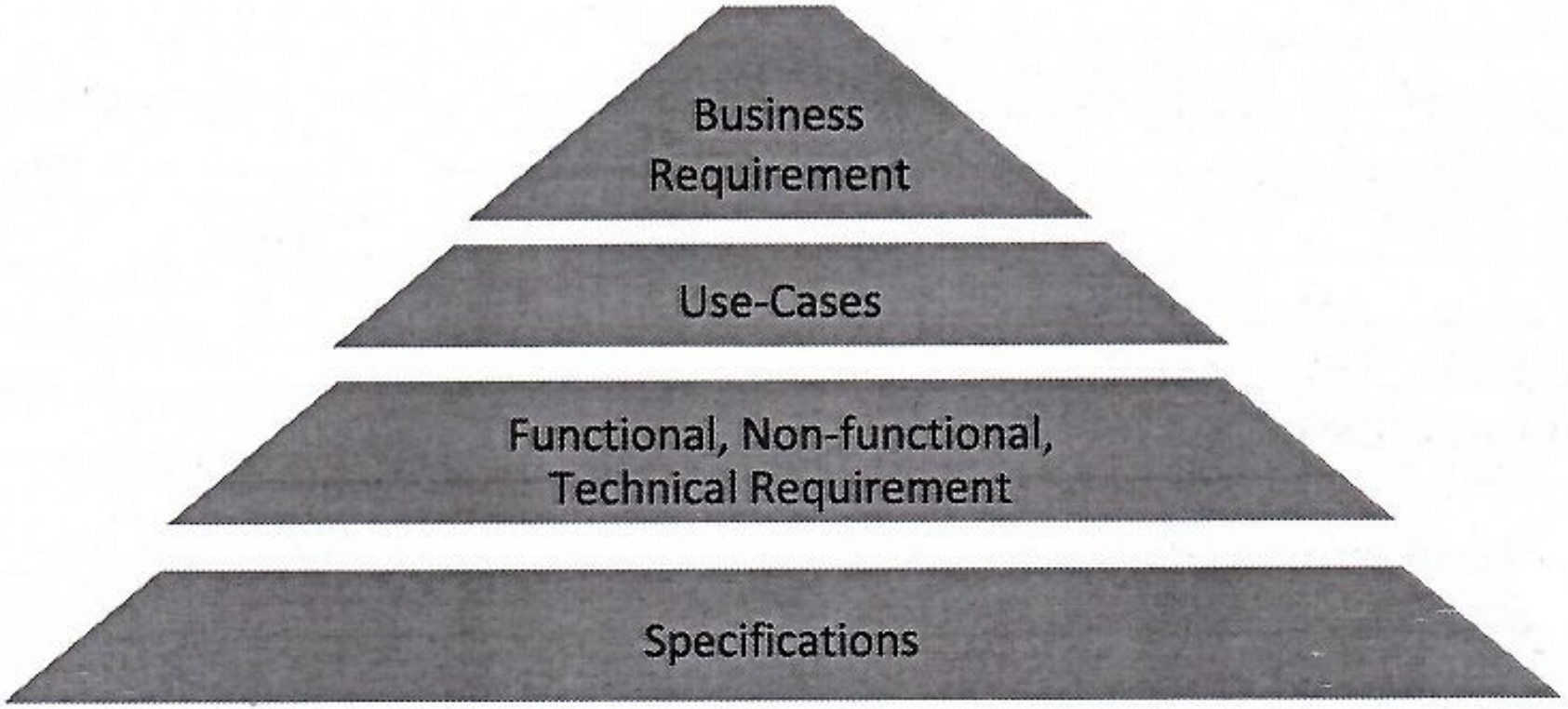
- надежность,
- эффективность,
- практичность,
- сопровождаемость,
- переносимость.

Измерение:
Отмена
занимает менее
одной секунды

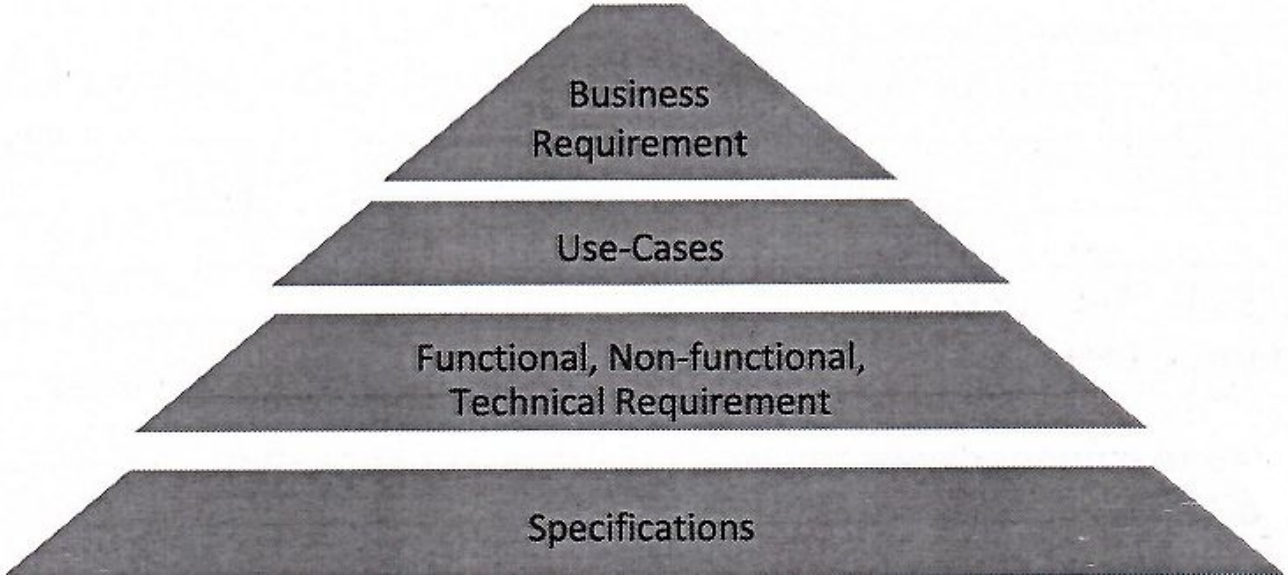
ПОНЯТИЕ О ТРЕБОВАНИЯХ

Требования – это документ, описывающий то, что должно быть реализовано в системе. В нем описано поведение системы, свойства системы и/или ее атрибуты.

Выделяют следующие уровни требований:

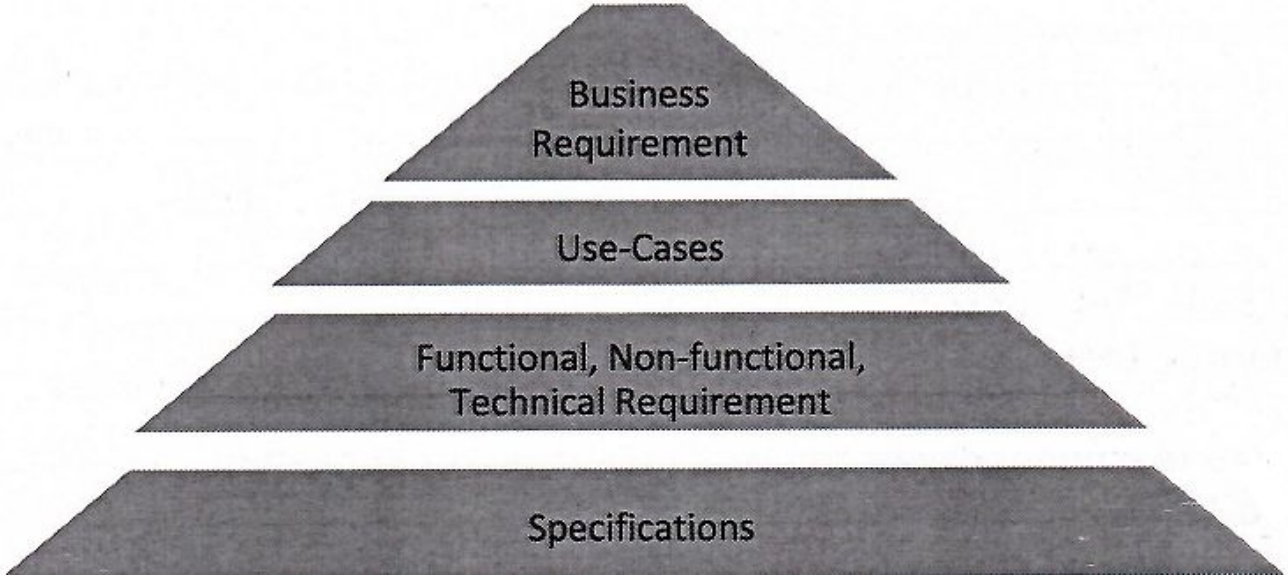


ПОНЯТИЕ О ТРЕБОВАНИЯХ



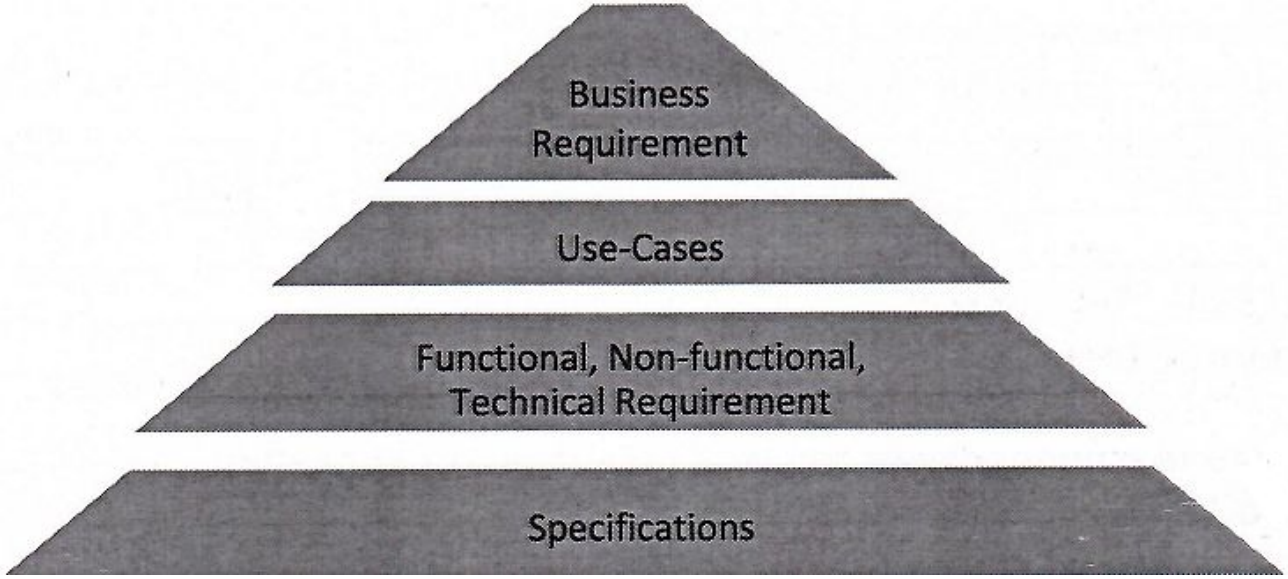
- **Уровень бизнес-требований** (Business Requirement) – «общее видение системы», то есть основная концепция приложения (Vision). Например, нужно разработать приложение, которое увеличит продажи благодаря возможности просматривать каталог продукции и заказывать товары онлайн.

ПОНЯТИЕ О ТРЕБОВАНИЯХ



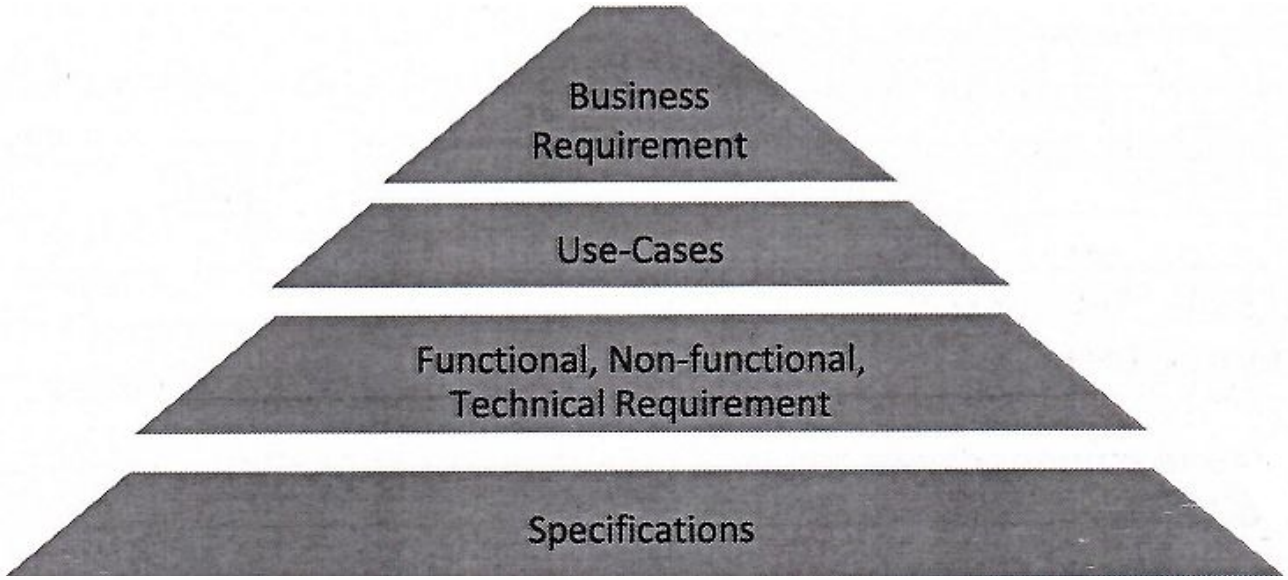
- **Уровень пользовательских требований** (User Requirements, Use-Cases) – «что можно будет делать», то есть варианты использования системы. Например, в приложении можно создать личный кабинет, в котором будут размещаться истории заказов, можно посмотреть каталог всей продукции по категориям, отмечать понравившиеся товары, добавлять в корзину, делать заказ, оплачивать его онлайн.

ПОНЯТИЕ О ТРЕБОВАНИЯХ



- **Уровень функциональных и нефункциональных требований** (Functional, Non-functional, Technical Requirement) – «что конкретно должна выполнять система и каким образом она должна это выполнять». Например, какие страницы в приложении, какие вкладки, как выглядят вкладки, какие переходы между вкладками (в нашем примере – на странице оформления заказа после заполнения всех необходимых полей становится доступной кнопка оплаты, которая переводит на форму для ввода карты и т.д.).

ПОНЯТИЕ О ТРЕБОВАНИЯХ



- **Детализация требований в спецификации** (Specifications) – «система должна соответствовать таким-то параметрам», то есть требованиям, уточненным до числовых характеристик. Например, какие поля должны быть на странице заказа, в какой последовательности они должны располагаться на странице, какие допустимые значения для ввода в поля, какие поля являются обязательными, каковы их точные размеры.

АНАЛИЗ ТРЕБОВАНИЙ

Вне зависимости от того, какая модель разработки ПО используется на проекте, чем позже будет обнаружена проблема, тем сложнее и дороже будет её решение. А в самом начале («водопада», «спуска по букве v», «итерации», «витка спирали») идёт планирование и работа с требованиями.

Если проблема в требованиях будет выяснена на этой стадии, её решение может свестись к исправлению пары слов в тексте, в то время как недоработка, вызванная пропущенной проблемой в требованиях и обнаруженная на стадии эксплуатации, может даже полностью уничтожить проект.

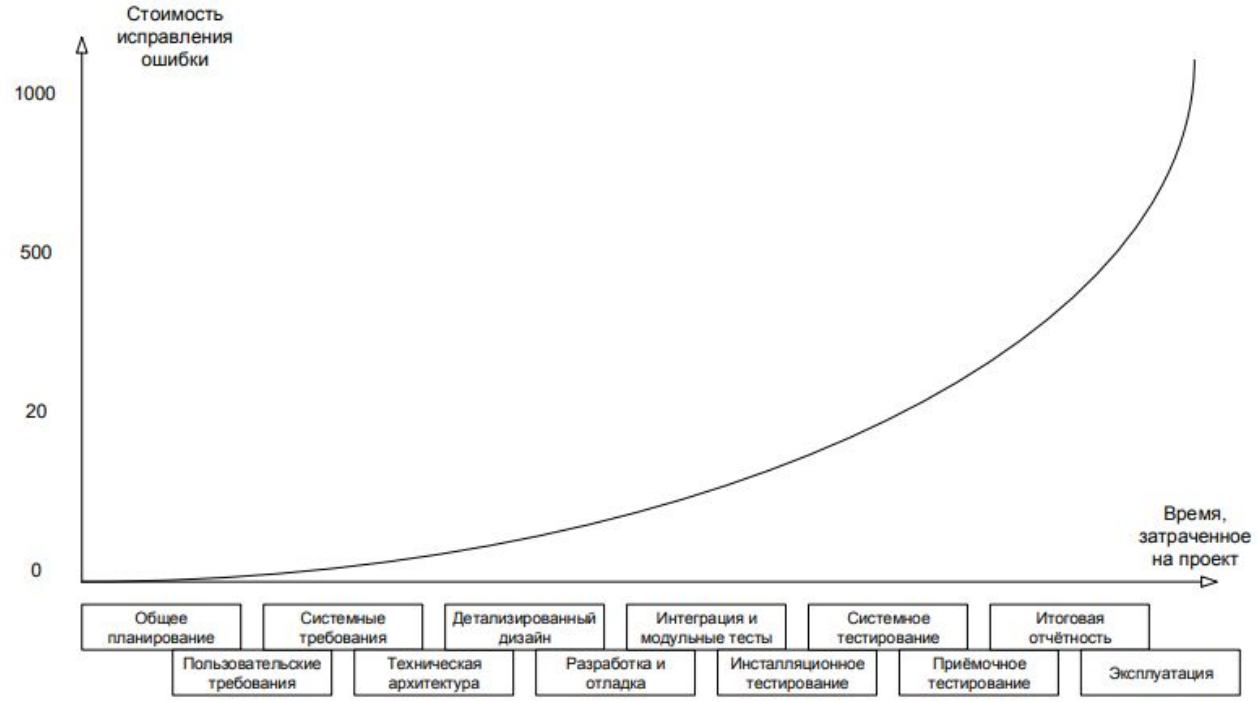


Рисунок 2.2.а — Стоимость исправления ошибки в зависимости от момента её обнаружения

Требования могут быть представлены в виде:

Общая концепция (Vision)

Диаграммы, блок-схемы

Варианты использования (Use cases)

Истории использования (User Stories)

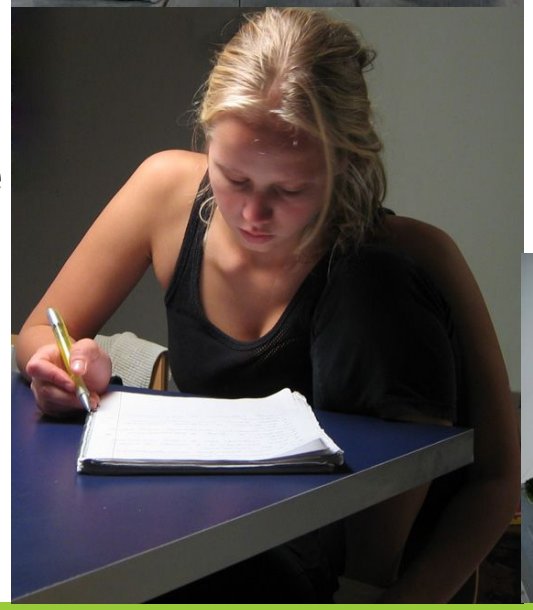
Mock-ups (отрисованные страницы приложения)

Функциональные спецификации (Functional specs)

Готовое приложение

Пути выявления требований

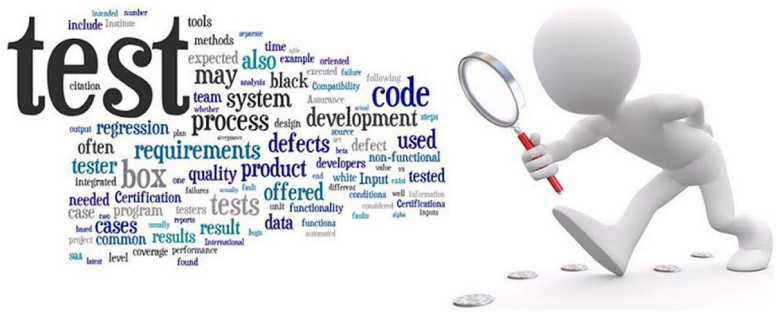
- Интервью
- Работа с фокус группами
- Наблюдение
- Анкетирование
- Самостоятельное описание
- Семинары
- Прототипирование

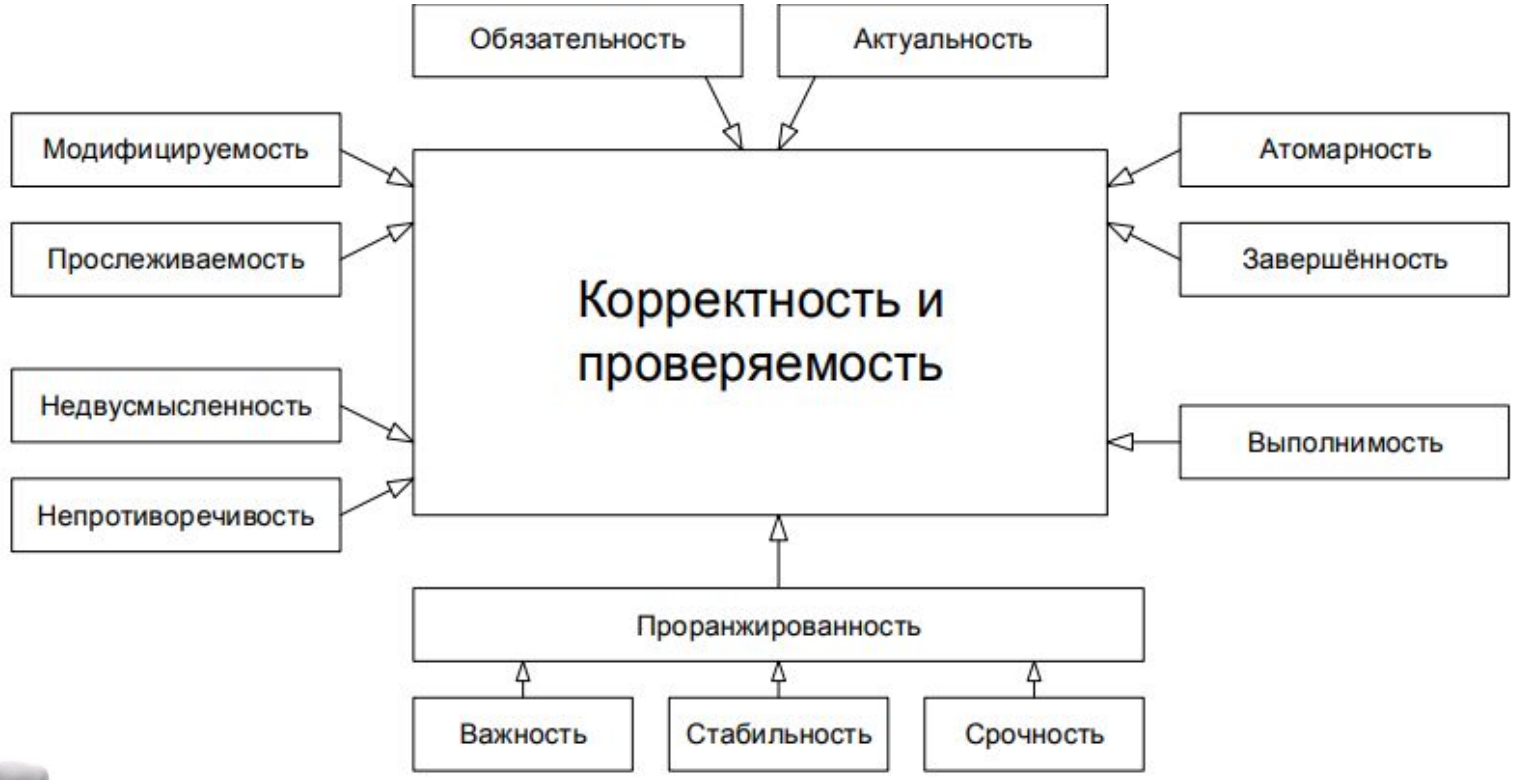


АНАЛИЗ ТРЕБОВАНИЙ

СВОЙСТВА ХОРОШИХ ТРЕБОВАНИЙ:

1) Завершенность
2) Непротиворечивость
3) Корректность
4) Недвусмысленность
5) Проверяемость
6) Модифицируемость
7) Прослеживаемость
8) Проранжированность





На переговорах все друг друга понимают

Визуализация же показывает иное

Слова-индикаторы.

В английском языке есть много слов-индикаторов, наличие которых в требовании должно насторожить тестировщика:

Adequate, be able to, easy, provide for, as a minimum, be capable of, effective, timely, as applicable, if possible, TBD, as appropriate, if practical, at a minimum, but not limited to, capability of, capability to, normal, minimize, maximize, optimize, rapid, user-friendly, simple, often, usual, large, flexible, robust, state-of-the-art, improved, efficient.

И ГЛАВНОЕ !

Требования нужны :

- **НЕ** для того, чтобы на проекте были требования,
- а для того, чтобы все участники процесса понимали **как** будущее приложение **должно работать**!

www.bash.im

skp: решил добавить на сайт при регистрации контрольный вопрос:
"Напишите СЛОВОМ первую ЦИФРУ года вашего рождения", и после этого резко упало кол-во регистраций на сайте

skp: посмотрев ответы причина стала ясна - около половины юзеров писала в ответе ТЫСЯЧА…

Проблемы спецификаций:

- Противоречия между картинкой и текстом;
- Противоречия между новой картинкой и старой;
- Противоречия внутри одной картинки;
- Логика работы функции непонятна (кнопка есть, но непонятно для чего);
- Длинные спецификации, невозможно редактировать, отслеживать изменения;
- Много страниц, трудно читать и работать с документом.

Техники тестирования требований

1. **Взаимный просмотр (peer review)**

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 выполняется группой специалистов

Формальная инспекция.

Техники тестирования требований

2. Вопросы

Самый простой и не требующий большого опыта способ – задавать как можно больше вопросов

Искусство задавать вопросы

«Умение ставить разумные вопросы есть уже важный и необходимый признак ума или проницательности» И.Кант

Умение задать правильный вопрос

требует соответствующего навыка

Огромный умственный труд стоит

между «Я ничего не понял» и

правильным вопросом

АНАЛИЗ ТРЕБОВАНИЙ

Таблица 2.2.а — Пример плохих и хороших вопросов к требованиям

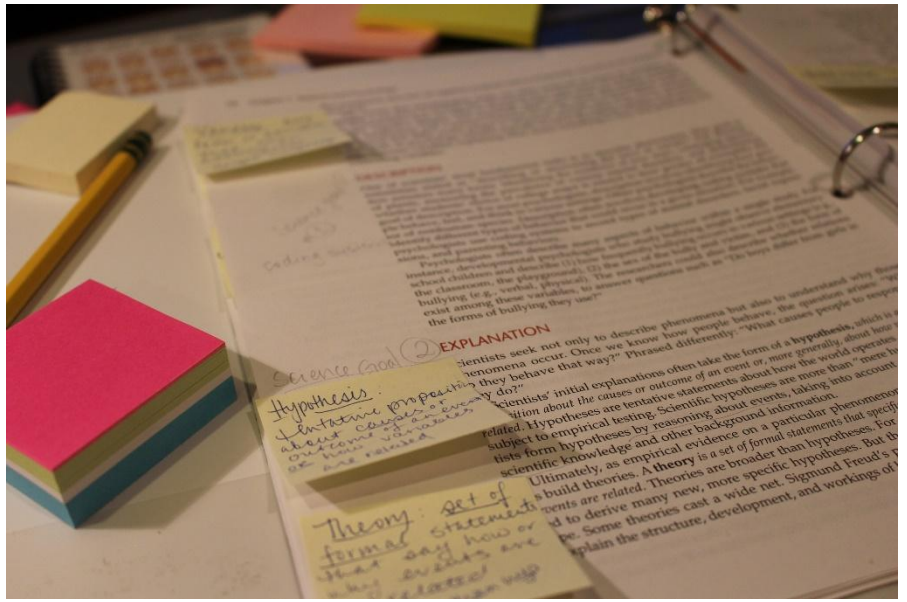
Плохое требование	Плохие вопросы	Хорошие вопросы
«Приложение должно быстро запускаться».	«Насколько быстро?» (На это вы рискуете получить ответы в стиле «очень быстро», «максимально быстро», «нууу… просто быстро»). «А если не получится быстро?» (Этим вы рискуете просто удивить или даже разозлить заказчика.) «Всегда?» («Да, всегда». Хм, а вы ожидали другого ответа?)	«Каково максимально допустимое время запуска приложения, на каком оборудовании и при какой загруженности этого оборудования операционной системой и другими приложениями? На достижение каких целей влияет скорость запуска приложения? Допускается ли фоновая загрузка отдельных компонентов приложения? Что является критерием того, что приложение закончило запуск?»
«Опционально должен поддерживаться экспорт документов в формат PDF».	«Любых документов?» (Ответы «да, любых» или «нет, только открытых» вам всё равно не помогут.) «В PDF какой версии должен производиться экспорт?» (Сам по себе вопрос хорош, но он не даёт понять, что имелось в виду под «опционально».) «Зачем?» («Нужно!» Именно так хочется ответить, если вопрос не раскрыт полностью.)	«Насколько возможность экспорта в PDF важна? Как часто, кем и с какой целью она будет использоваться? Является ли PDF единственным допустимым форматом для этих целей или есть альтернативы? Допускается ли использование внешних утилит (например, виртуальных PDF-принтеров) для экспорта документов в PDF?»
«Если дата события не указана, она выбирается автоматически».	«А если указана?» (То она указана. Логично, не так ли?) «А если дату невозможно выбрать автоматически?» (Сам вопрос интересен, но без пояснения причин невозможности звучит как издёвка.) «А если у события нет даты?» (Тут автор вопроса, скорее всего, хотел уточнить, обязательно ли это поле для заполнения. Но из самого требования видно, что обязательно: если оно не заполнено человеком, его должен заполнить компьютер.)	«Возможно, имелось в виду, что дата **генерируется** автоматически, а не **выбирается**? Если да, то по какому алгоритму она генерируется? Если нет, то из какого набора выбирается дата и как генерируется этот набор? P.S. Возможно, стоит использовать текущую дату?»

Перед тем как задать вопросы:

- вычитать требования с карандашом, сделать пометки

- найти ответ на часть вопросов в Google

- вычитать требования второй раз (сложится общая картина, найдутся ответы еще на часть вопросов)



Техники тестирования требований

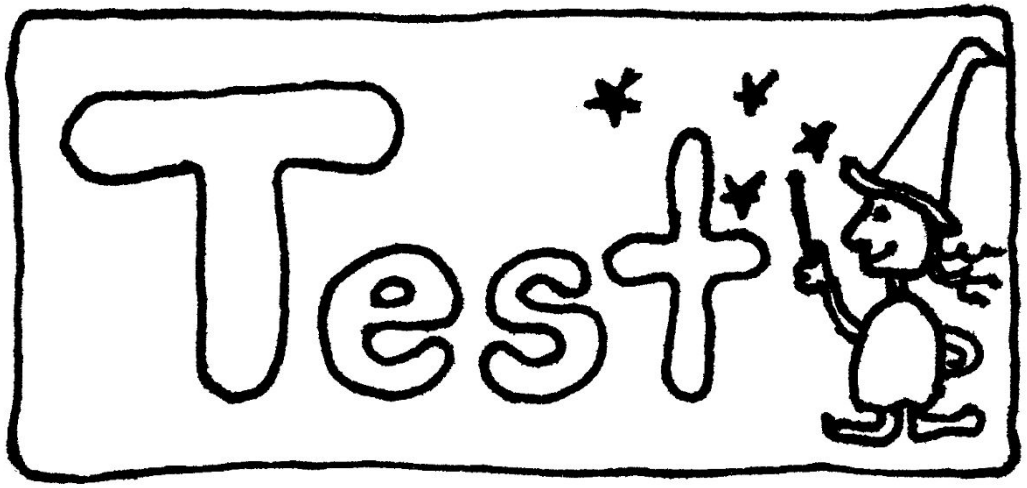
3. Тест кейсы

Когда вы видите требование, спросите себя: «Как я буду его тестировать?

Какие тесты очевидно покажут, что это требование реализовано правильно?»

Во время написания тест кейсов

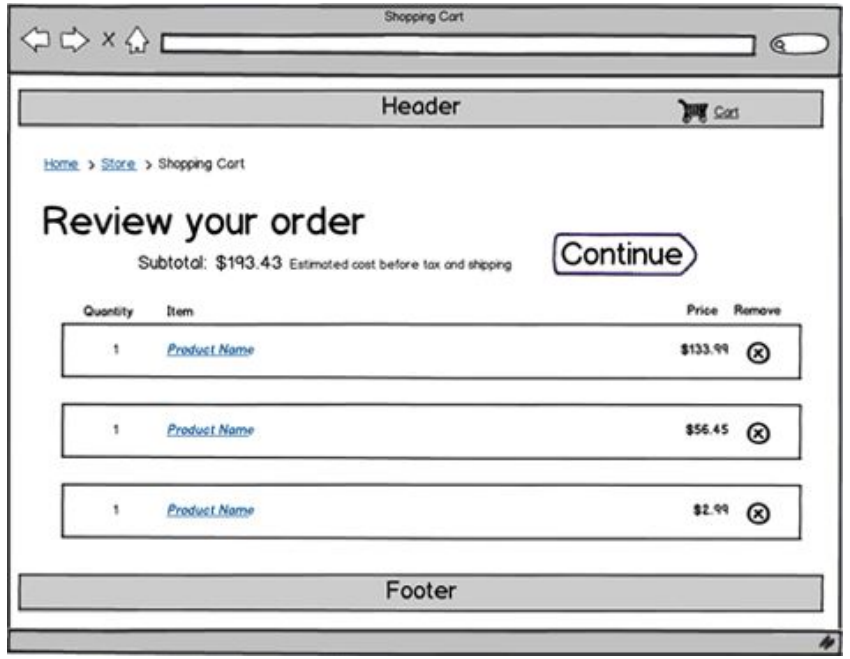
возникнут новые вопросы!



Техники тестирования требований

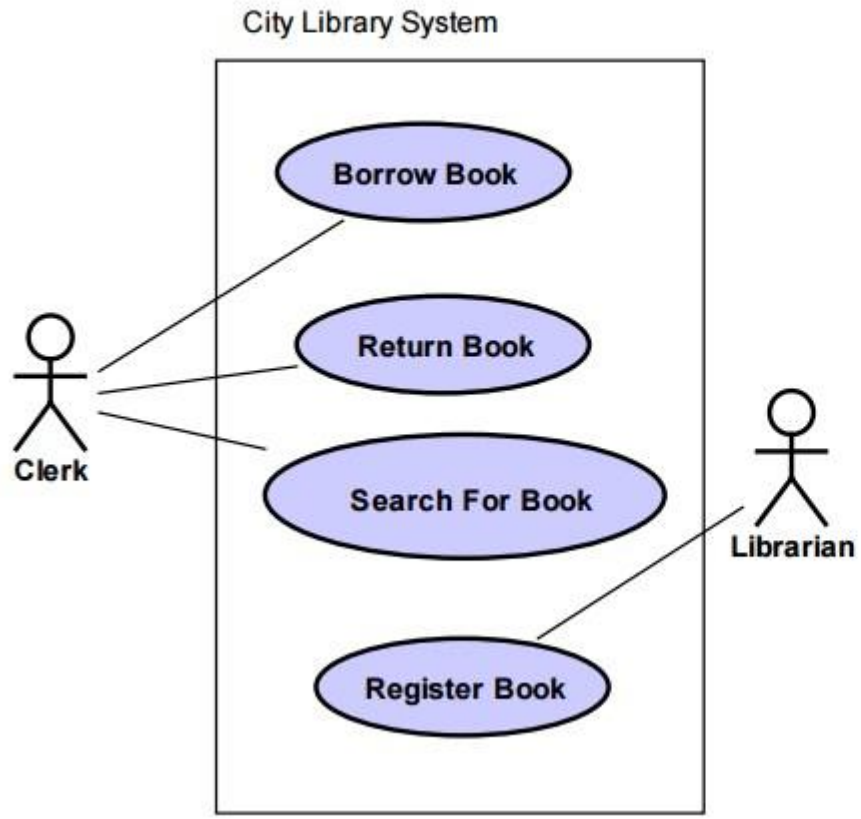
4. Рисунки, схемы и т.д.

Чтобы увидеть общую картину требований целиком, очень удобно использовать рисунки, схемы, диаграммы и т.п.

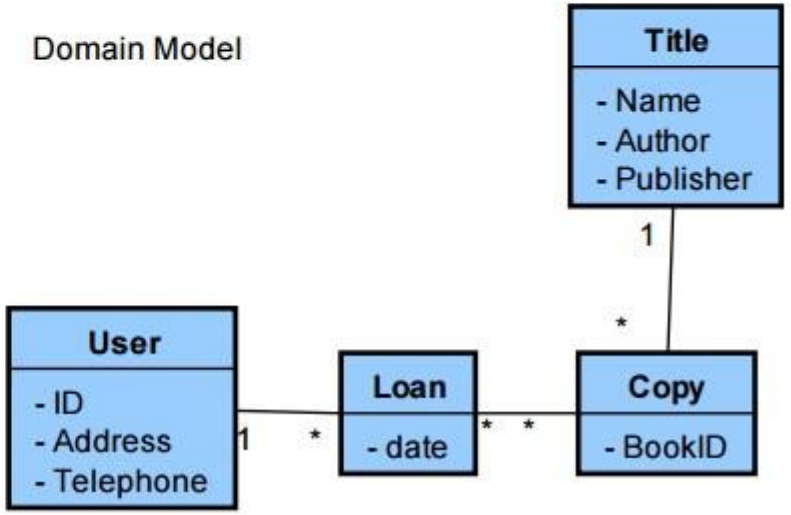


Интеллект-карты (диаграммы связей)

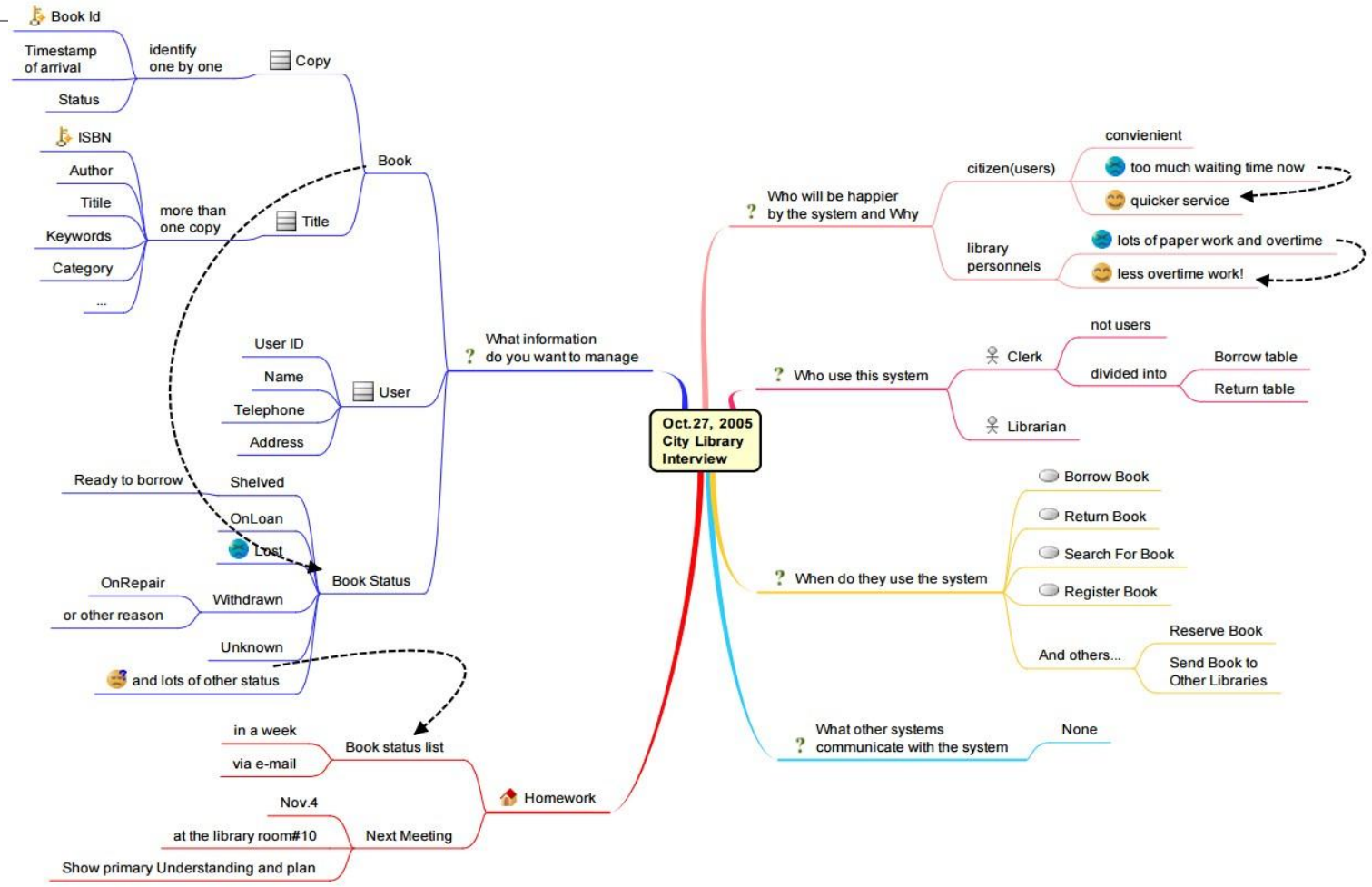
Use Case Diagram



Class Diagram



Интеллект-карты (диаграммы связей)



АНАЛИЗ ТРЕБОВАНИЙ

Рекомендации и техники по работе с требованиями:

- Начинаем анализировать (тестировать) требования как можно раньше, так как ошибка, найденная в требованиях на ранней стадии проекта дешевле в исправлении, чем такая же ошибка, найденная на конечной стадии проекта. А также хорошо протестированные требования позволяют сократить количество ошибок, допущенных разработчиками.
- Несколько раз вдумчиво читаем. Помечаем непонятные места. Не задаем вопросы, пока не дочитали до конца и второй раз, так как дальше в требованиях может быть ответ на вопрос.
- Пытаемся самостоятельно найти ответы на вопросы в Google (например, некоторые незнакомые аббревиатуры), уточняем у других членов команды (возможно, они разобрались в этом требовании).
- Рисуем диаграммы, схемы, интеллект-карты, которые помогут структурировать информацию, составить связи между сущностями приложения, найти пропущенные детали требований или ответы на возникшие вопросы.
- Составляем чек-лист, тест-кейсы (во время составления тест-кейсов, как правило, появляются дополнительные вопросы, ответов на которые не хватает в требованиях).

АНАЛИЗ ТРЕБОВАНИЙ

- Задаем четкие, последовательные, структурированные вопросы человеку, который компетентен в этом и ответственен за требования на проекте (бизнес-аналитику, заказчику и др.).
- Следует уточнять даже такие мелочи, как обязательность полей для заполнения, чувствительность к регистру текста, допустимая длина, разрешенные/запрещенные символы, поддерживаемые языки, значения по умолчанию, точные тексты сообщений.
- Расценивайте полученный ответ как новое требование, как можно быстрее актуализируйте свои тесты, согласно новым данным, если вы не получили нужное представление о интересующем вас вопросе, то задайте дополнительные вопросы (иногда нужно 5+ циклов вопрос/ответ, чтобы решить проблему).

Что нужно знать..

1. Где хранятся требования?
2. Какие источники требований у нас есть?
3. Кто помещает требования в официальное хранилище?
4. Как мы узнаем об изменениях в требованиях?
5. Что более правильно – изначальные спецификации, последующие письма, прототип?
6. Кто утверждает окончательные требования?
7. Как найти новые/измененные требования?
8. Как мы можем предложить внести изменения в требования?
9. Кому мы можем задавать вопросы?
10. Кому и как мы должны сообщить о проблемах с требованиями?
11. Если нам не отвечают, кого спрашивать следующим, кого в конечном итоге?

Типичные ошибки при анализе и тестировании требований

1. Изменение формата файла и документа.

По какой-то непонятной причине очень многие начинающие тестировщики стремятся полностью уничтожить исходный документ, заменив текст таблицами (или наоборот), перенеся данные из Word в Excel и т.д. Это можно сделать только в одном случае: если вы предварительно договорились о подобных изменениях с автором документа. В противном случае вы полностью уничтожаете чью-то работу, делая дальнейшее развитие документа крайне затруднительным.

Самое худшее, что можно сделать с документом, — это сохранить его в итоге в некоем формате, предназначенном скорее для чтения, чем для редактирования (PDF, набор картинок и тому подобное).

Типичные ошибки при анализе и тестировании требований

2. Отметка того факта, что с требованием всё в порядке.

Если у вас не возникло вопросов и/или замечаний к требованию — не надо об этом писать. Любые пометки в документе подсознательно воспринимаются как признак проблемы, и такое «одобрение требований» только раздражает и затрудняет работу с документом — сложнее становится заметить пометки, относящиеся к пробле

Типичные ошибки при анализе и тестировании требований

3. Описание одной и той же проблемы в нескольких местах.

Помните, что ваши пометки, комментарии, замечания и вопросы тоже должны обладать свойствами хороших требований (настолько, насколько эти свойства к ним применимы). Если вы много раз в разных местах пишете одно и то же об одном и том же, вы нарушаете как минимум свойство модифицируемости. Постарайтесь в таком случае вынести ваш текст в конец документа, укажите в нём же (в начале) перечень пунктов требований, к которым он относится, а в самих требованиях в комментариях просто ссылайтесь на этот текст.

Типичные ошибки при анализе и тестировании требований

4. Написание вопросов и комментариев без указания места требования, к которым они относятся.

Если ваше инструментальное средство позволяет указать часть требования, к которому вы пишете вопрос или комментарий, сделайте это (например, Word позволяет выделить для комментирования любую часть текста — хоть один символ). Если это невозможно, цитируйте соответствующую часть текста. В противном случае вы порождаете неоднозначность или вовсе делаете вашу пометку бессмысленной, т.к. становится невозможно понять, о чём вообще идёт речь.

Типичные ошибки при анализе и тестировании требований

5. Задавание плохо сформулированных вопросов.

Эта ошибка была подробно рассмотрена выше. Однако добавим, что есть ещё три вида плохих вопросов:

• Первый вид возникает из-за того, что автор вопроса *не знает общепринятой терминологии или типичного поведения стандартных элементов интерфейса* (например, «что такое чек-бокс?», «как в списке можно выбрать несколько пунктов?», «как подсказка может всплывать?»).

• Второй вид плохих вопросов похож на первый из-за формулировок: вместо того, чтобы написать «что вы имеете в виду под {чем-то}?», автор вопроса пишет «что такое {что-то}?» То есть вместо вполне логичного уточнения получается ситуация, очень похожая на рассмотренную в предыдущем пункте.

• Третий вид сложно привязать к причине возникновения, но его суть в том, что *к некорректному и/или невыполнимому требованию задаётся вопрос* наподобие «что будет, если мы это сделаем?». Ничего не будет, т.к. мы это точно не сделаем. И вопрос должен быть совершенно иным (каким именно — зависит от конкретной ситуации, но точно не таким)

Типичные ошибки при анализе и тестировании требований

5. Задавание плохо сформулированных вопросов.

И ещё раз напомним о точности формулировок: иногда одно-два слова могут на корню уничтожить отличную идею, превратив хороший вопрос в плохой.
Сравните: «Что такое формат даты по умолчанию?» и «Каков формат даты по умолчанию?».
Первый вариант просто показывает некомпетентность автора вопроса, тогда как второй — позволяет получить полезную информацию.

К этой же проблеме относится непонимание контекста. Часто можно увидеть вопросы в стиле «о каком приложении идёт речь?», «что такое система?» и им подобные. Чаще всего автор таких вопросов просто вырвал требование из контекста, по которому было совершенно ясно, о чём идёт речь

Типичные ошибки при анализе и тестировании требований

6. Написание очень длинных комментариев и/или вопросов.

История знает случаи, когда одна страница исходных требований превращалась в 20–30 страниц текста анализа и вопросов. Это плохой подход. Все те же мысли можно выразить значительно более кратко, чем сэкономить как своё время, так и время автора исходного документа. Тем более стоит учитывать, что на начальных стадиях работы с требованиями они весьма нестабильны, и может получиться так, что ваши 5–10 страниц комментариев относятся к требованию, которое просто удалят или изменят до неузнаваемости

Типичные ошибки при анализе и тестировании требований

7. Критика текста или даже его автора.

Помните, что ваша задача — сделать требования лучше, а не показать их недостатки (или недостатки автора). Потому комментарии вида «плохое требование», «неужели вы не понимаете, как глупо это звучит», «надо переформулировать» неуместны и недопустимы.

Типичные ошибки при анализе и тестировании требований

8. Категоричные заявления без обоснования.

Как продолжение ошибки «критика текста или даже его автора» можно отметить и просто категоричные заявления наподобие «это невозможно», «мы не будем этого делать», «это не нужно». Даже если вы понимаете, что требование бессмысленно или невыполнимо, эту мысль стоит сформулировать в корректной форме и дополнить вопросами, позволяющими автору документа самому принять окончательное решение. Например, «это не нужно» можно переформулировать так: «Мы сомневаемся в том, что данная функция будет востребована пользователями. Какова важность этого требования? Уверены ли вы в его необходимости?»

Типичные ошибки при анализе и тестировании требований

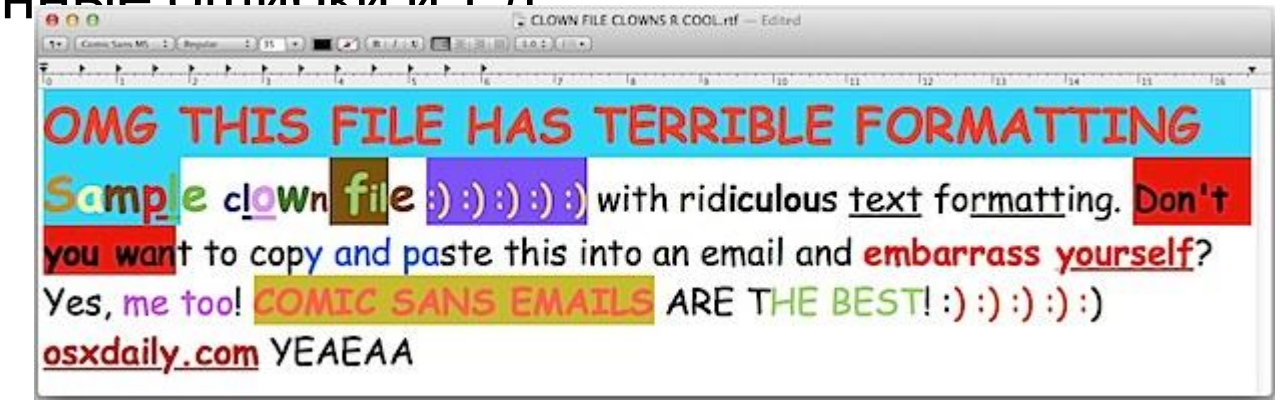
9. Указание проблемы с требованиями без пояснения её сути.

Помните, что автор исходного документа может не быть специалистом по тестированию или бизнес-анализу. Потому просто пометка в стиле «неполнота», «двусмысленность» и т.д. могут ничего ему не сказать. Поясняйте свою мысль. Сюда же можно отнести небольшую, но досадную недоработку, относящуюся к противоречивости: если вы обнаружили некие противоречия, сделайте соответствующие пометки во всех противоречащих друг другу местах, а не только в одном из них. Например, вы обнаружили, что требование 20 противоречит требованию 30. Тогда в требовании 20 отметьте, что оно противоречит требованию 30, и наоборот. И поясните суть противоречия.

Типичные ошибки при анализе и тестировании требований

10. Плохое оформление вопросов и комментариев.

Старайтесь сделать ваши вопросы и комментарии максимально простыми для восприятия. Помните не только о краткости формулировок, но и об оформлении текста (например, если вопросы структурированы в виде списка — такая структура воспринимается намного лучше, чем сплошной текст). Перечитайте свой текст, исправьте опечатки, грамматические и пунктуационные ошибки и т.д.



Типичные ошибки при анализе и тестировании требований

11. Описание проблемы не в том месте, к которому она относится.

Классическим примером может быть неточность в сноске, приложении или рисунке, которая почему-то описана не там, где она находится, а в тексте, ссылающемся на соответствующий элемент. Исключением может считаться противоречивость, при которой описать проблему нужн

Типичные ошибки при анализе и тестировании требований

12. Ошибочное восприятие требования как «требования к пользователю»

Требования в стиле «пользователь должен быть в состоянии отправить сообщение» являются некорректными. Но бывают ситуации, когда проблема намного менее опасна и состоит только в формулировке.

Например, фразы в стиле «пользователь может нажать на любую из кнопок», «пользователю должно быть видно главное меню» на самом деле означают «все отображаемые кнопки должны быть доступны для нажатия» и «главное меню должно отображаться».
Да, эту недоработку тоже стоит исправить, но не следует отмечать её как критическую проблему.

Типичные ошибки при анализе и тестировании требований

13. Скрытое редактирование требований

Эту ошибку можно смело отнести к разряду крайне опасных. Её суть состоит в том, что тестировщик произвольно вносит правки в требования, никак не отмечая этот факт. Соответственно, автор документа, скорее всего, не заметит такой правки, а потом будет очень удивлён, когда в продукте что-то будет реализовано совсем не так, как когда-то было описано в требованиях. Потому простая рекомендация: если вы что-то правите, обязательно отмечайте это (средствами вашего инструмента или просто явно в тексте). И ещё лучше отмечать правку как предложение по изменению, а не как свершившийся факт, т.к. автор исходного документа может иметь совершенно иной взгляд на ситуацию.

Типичные ошибки при анализе и тестировании требований

14. Анализ, не соответствующий уровню требований.

При тестировании требований следует постоянно помнить, к какому уровню они относятся, т.к. в противном случае появляются следующие типичные ошибки:

- Добавление в бизнес-требования мелких технических подробностей.
- Дублирование на уровне пользовательских требований части бизнес-требований (если вы хотите увеличить прослеживаемость набора требований, имеет смысл просто использовать ссылки).
- Недостаточная детализация требований уровня продукта (общие фразы, допустимые, например, на уровне бизнес-требований, здесь уже должны быть предельно детализированы, структурированы и дополнены подробной технической информацией).