



Тестирование



Содержание

1. Введение
2. Комплексный подход к обеспечению качества
3. Управление тестированием в рамках проекта
4. Из чего состоит тестирование

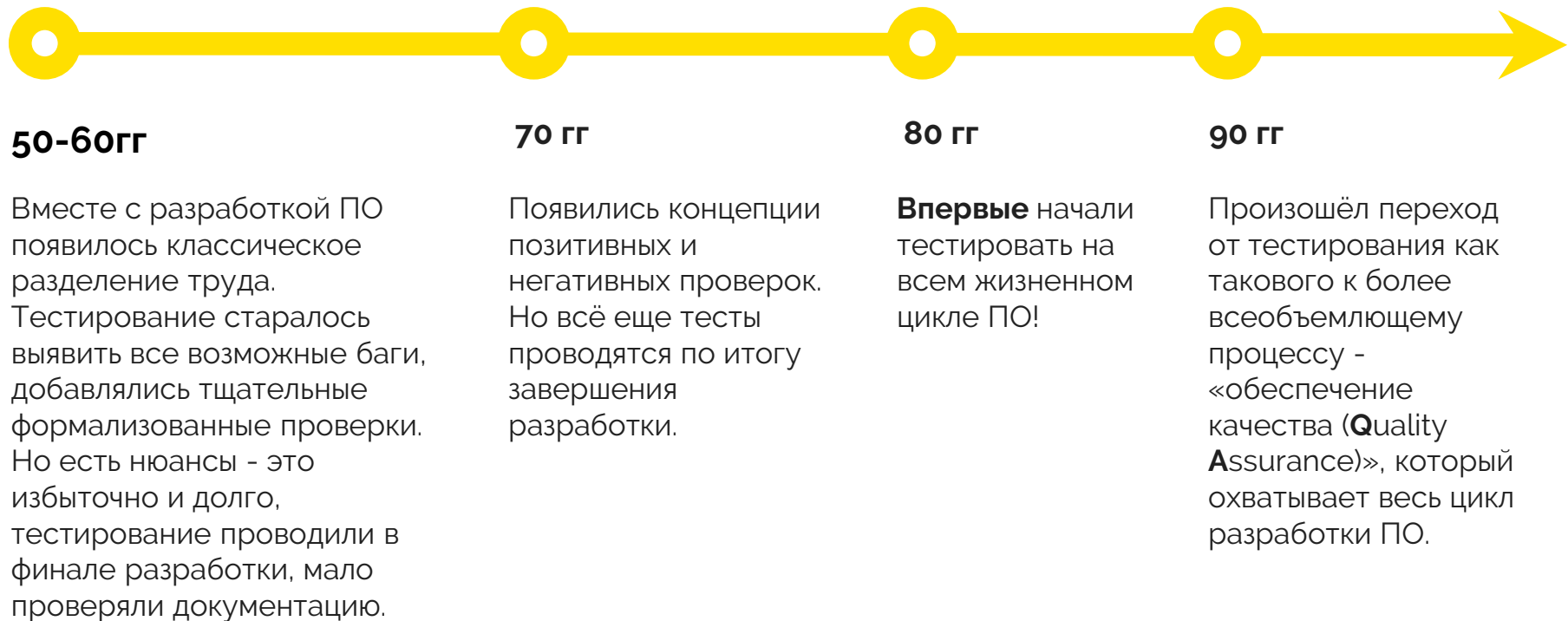
Когда появилось

В 1951 году Джозеф Джуран, которого сейчас считают отцом тестирования программного обеспечения, впервые отметил важность обеспечения качества программного обеспечения в своей книге «Руководство по контролю качества».

Он также определил 3 части управления качеством: планирование качества, контроль и улучшение.

А в 1957 году Чарльз Л. Бейкер разграничил **тестирование** программ от **отладки** в своём обзоре книги Дэна Маккракена «Программирование цифровых компьютеров».

Как развивалось



Как развивалось




00гг

Стало важно не соответствие программы требованиям, а **её способность предоставить конечному пользователю возможность эффективно решать свои задачи**. Появились методологии TDD (Test-Driven Development) и полноценные автотесты.

10гг-наше время

Гибкие методологии и гибкое тестирование, глубокая интеграция с процессом разработки, широкое использование автоматизации, колоссальный набор технологий и инструментальных средств, кроссфункциональность команды (когда тестировщик и программист во многом могут выполнять работу друг друга). Роль QA-инженера гораздо выше чем роль просто тестировщика.



Как тестирование превратилось в обеспечение качества

В 90-х, применение тестирования на протяжении всего цикла разработки, позволило не только быстро обнаруживать проблемы, но даже **предсказывать и предотвращать их появление.**

Развитие тестовой документации, позволило снизить зависимость от памяти и квалификации тестировщика, выработать общее для всей команды понимание процесса и получить **предсказуемый и повторяемый** результат.

Накопление тестовой документации и её анализ, подсказали выводы: посчитать, где дефектов больше, какие ошибки попадают чаще других. Это натолкнуло на мысль **«количественно» оценивать состояние качества.**

Оценив все результаты тестирования стало понятно, что качество формируется на каждом этапе производства и чаще на него влияет **работа с процессами.**



**Комплексный
подход к
обеспечению
качества продукта**

«Контроль качества» и «обеспечение качества»

Quality assurance –

качество не тождественно
тестированию.

Обеспечение качества это
постоянный процесс
направленный на
удовлетворение
требованиям.

Качество лежит в
области **работы с
потребителем.**

1

Оно основано на
**фактическом опыте
клиента** в отношении
продукта.

2

**Измеряется в соответствии с
требованиями клиента —**
формальными и не формальными,
осознанными или просто в
ощущениях (возможно, даже
полностью субъективных).

3

Качество всегда
**динамически
движется** с рынком и
потребителем.

4

«Контроль качества» и «обеспечение качества»

Quality control - контроль за качеством.
Основной способ проконтролировать - это провести тестирование конечного продукта.

Однако, дополнительно может контролироваться уровень инфраструктуры для проведения самих тестов, безопасность разработки на соответствие стандартам компании, документации на понятность для проведения проверок и формирования чек-листов, и т.п.

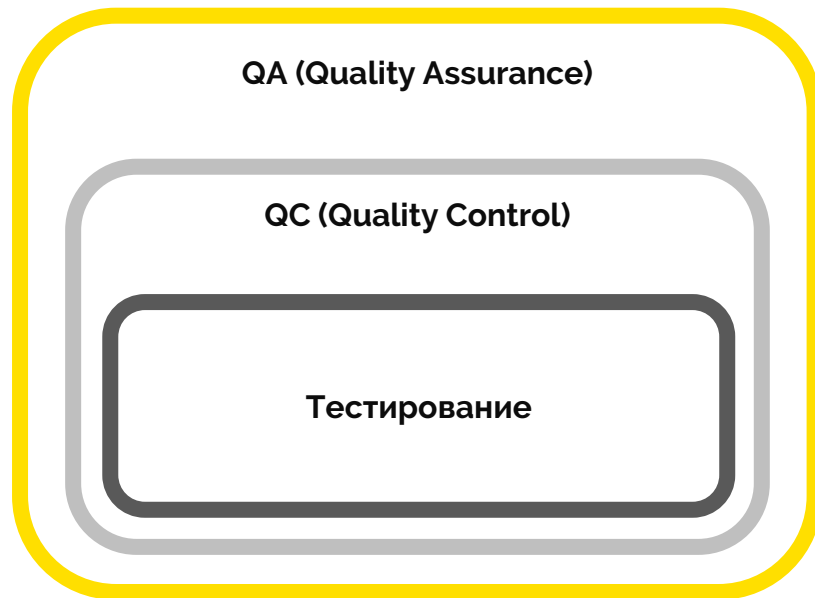
Тестирование – это **основная часть процесса QC**. Проводится по чек-листам и тест-кейсам силами QC-специалиста или автоматическими тестами.



Команда по качеству

1. QC-специалисты: контролируют соответствие требованиям, разрабатывают и выполняют тест-кейсы, заполняют тестовую документацию. Практически во всей отрасли «тестировщик» приравнивается к QC-специалисту.

1. QA-специалисты: контролируют качество на всех циклах производства, начиная с этапа планирования (могут повлиять на сами требования к продукту).





Команда по качеству. Кого набрать в проект?

QC – берем всегда. Чем легче и ближе к стартапу проект – тем меньше тестировщиков нужно.

Для фин.тех проектов соотношение разработчиков и тестировщиков может достигать 1:1.

QA – если есть несколько QC, то в команде должен быть и QA. Он контролирует процессы обеспечивающие качество и будет QA-lead для тестировщиков. На практике один QA может совмещать и функции управления качеством, и функции тестирования, особенно в небольших проектах.



**Влияние
тестирования
на управление
проектом**

Планирование

Что влияет на планирование сроков:

- Договоренность** с заказчиком о требуемом качестве
- Скорость тестирования** конкретных сотрудников
- Вовлеченность сотрудников** в другие процессы (сопровождение, аналитика и т.п.)
- Покрытие автотестами** (регресс, интеграции, API, нагрузочные и т.п.)

Действия управляющего проектом:

- Закладываем больше времени **на тестирование**
- Закладываем время на **покрытие автотестами и обеспечение этого процесса ресурсами** (специалисты, компетенции, инструментарий)
- Квотируем время** участия тестировщика в других активностях (сопровождение, аналитика и др.)

Повышение ценности

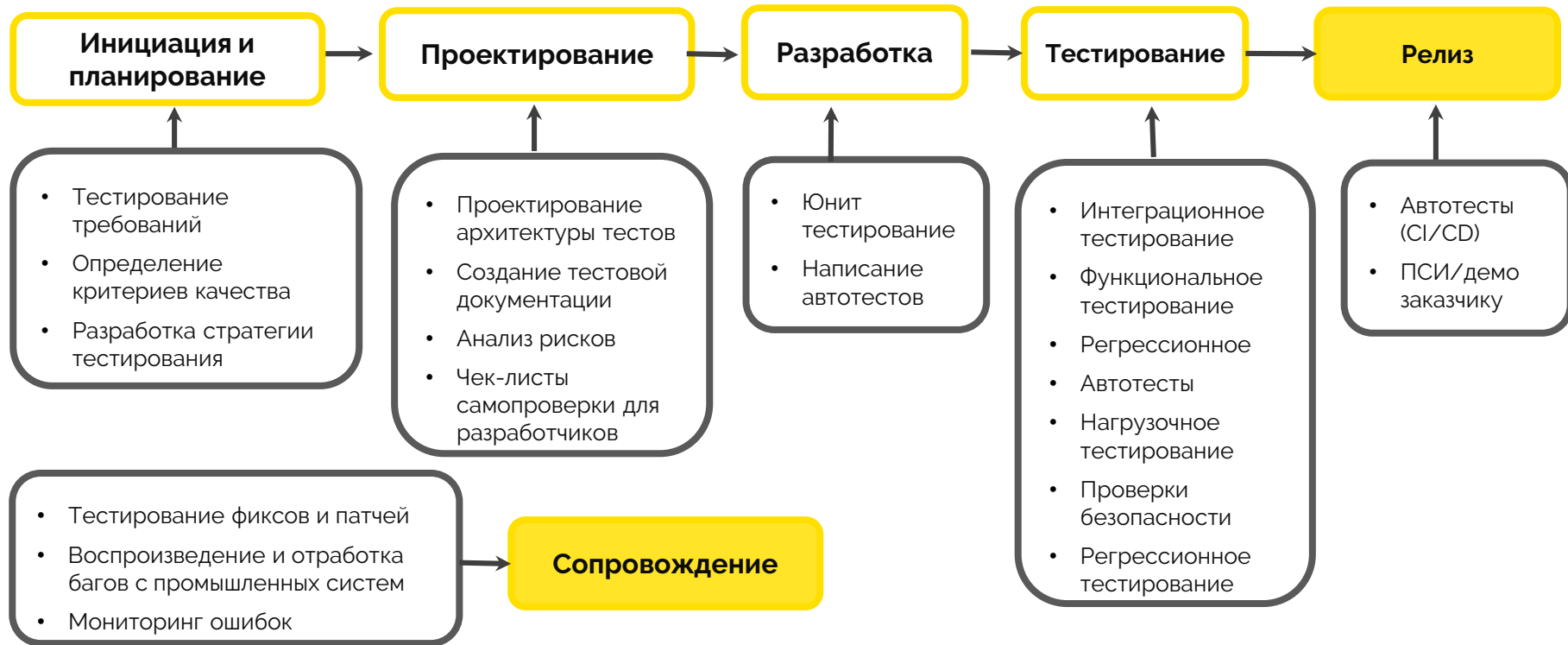
Повышение ценности для клиента

- Удовлетворение **ожиданий** клиента по качеству продукта
- Повторяемый **стабильный результат** релиза
- Поиск способов **превзойти ожидания** клиента по качеству и юзабилити продукта

Повышение ценности для производства

- Сохранение и улучшение качества при снижении стоимости** (shift-left практики – выявление ошибки на этом этапе ниже в 4-5 раз в сравнении с ошибкой на этапе тестирования. Внедрение автотестов и сокращение времени проверок)
- Анализ багов** с промышленных систем, **минимизация рисков повторения** (добавление проверочных кейсов, изменение сценариев совместно с разработчиком и т.п.)

Управление качеством в цикле разработки



Командное взаимодействие на этапах

Инициация и планирование

Заказчик

Руководитель проекта

Аналитик

Проектирование

Руководитель проекта

Архитектор

Артефакт взаимодействия:

- Отчет для аналитика о тестировании требований
- Стратегия тестирования
- План тестирования
- Оценка времени тестирования

Артефакт взаимодействия:

- Чек-лист самопроверки для разработчика
- Документ с оценкой соответствия архитектуры требованиям

Командное взаимодействие на этапах

Разработка

Тимлид
Разработчик

Артефакт взаимодействия:

- Автотесты
- Отчет о покрытии юнит-тестами

Тестирование

Руководитель проекта
Аналитик
Разработчик

Артефакт взаимодействия:

- Заполненная тестовая документация
- Пройденные чек-листы
- Зафиксированные тикеты (баги) в ПО
- Отчеты о работе автотестов
- Отчеты о тестах производительности
- Отчеты о тестах безопасности

Командное взаимодействие на этапах

Релиз

Руководитель проекта
Заказчик
Разработчик

Артефакт взаимодействия:

- Подготовленный документ ПСИ
- Презентация Демо заказчику
- Отчет о прохождении автотестов в рамках релиза в CI/CD

Сопровождение

Инженер сопровождения
Руководитель проекта

Артефакт взаимодействия:

- Зафиксированные тикеты (баги) в ПО
- Метрики, графики мониторингов ошибок

Влияние тестирования на ТТМ проекта

Руководитель проекта учитывает влияние на человеческие ресурсы тестировщиков и результаты их деятельности

1. **Субъективная оценка.** Тестировщики делают оценку своей работы, она может быть слишком позитивной или слишком негативной.
2. При вовлеченности тестировщика на последнем этапе разработки **без тестирования требований возрастает вероятность получить «не то»**. Получаем возврат функционала в разработку.
3. Обнаруженные критичные баги требуют немедленного исправления, происходит **сдвиг по первоначальным оценкам на тестирование** (понадобится ретест багов после исправления).
4. Тестировщики **могут заблокировать поставку релиза**, если наделены такой способностью
5. **Нужно учитывать занятость тестирования в процессах поддержки ПО:**
 - воспроизведение багов из промышленной среды,
 - консультирование по работе продукта,
 - помощь поддержке в классификации багов
 - составление баг-репортов

(необходимо квотировать/закладывать время участия сотрудников в активностях, связанных с работой поддержки).



**Глубже в
тестирование**

Проблемы и методы их решения

Что такое тестирование – это процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.

Тестирование не может быть «полным», ошибки могут быть даже в тщательно проверенных продуктах.

Как быть?

- 1. Управлять ожиданиями заказчика** по предоставляемому качеству. Например, минимальное требование качества – нет критичных ошибок, блокирующих бизнес функционал после релиза. Это область ответственности руководителя проекта.
- 2. Повышать качество самого тестирования.** Это область ответственности QA, поговорим об этом дальше!



Как повысим качество через тестирование?

1. **Shift-left практики** (сдвиг тестирования «налево» по циклу производства)
2. **Тщательное планирование и разработка стратегии тестирования**
3. **Эффективное ручное тестирование**

4. **Внедрение автотестов** (в следующей лекции)
5. **Внедрение тестов производительности** (в следующей лекции)
6. **Работа с приоритетами и важностью багов** (в следующей лекции)

Shift-left практики

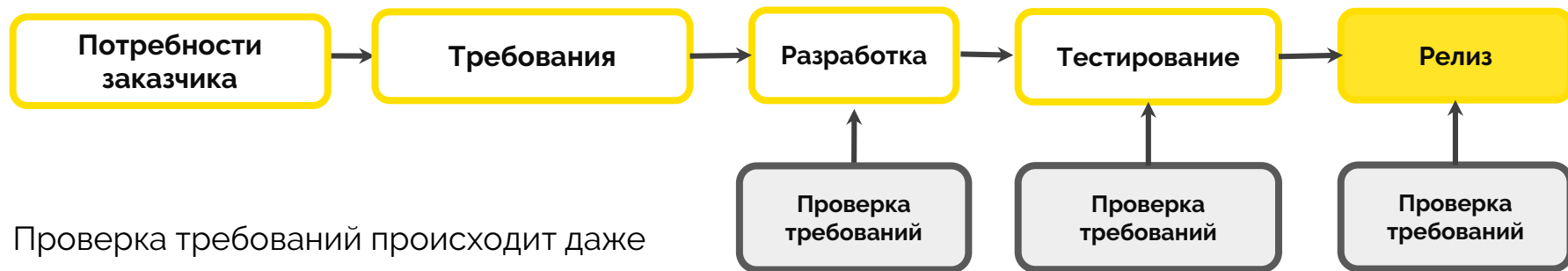
1. Тестирование требований до начала разработки – не равно прочтению требований. Это применение техник тестирования, но не к софту, а *к документации*.

2. Чек-листы самопроверки для разработчика – это краткий чек лист, который соответствует требованиям и критериям приёмки. Его использует для проверки готового решения *сам разработчик*, ещё до передачи в тестирование. Это гарантирует, что код как минимум рабочий. Так мы избегаем отладки через тестирование.

Баги обнаруженные на этих этапах до 5 раз дешевле чем баги обнаруженные в промышленной эксплуатации.

Тестирование требований

БЕЗ этапа тестирования требований

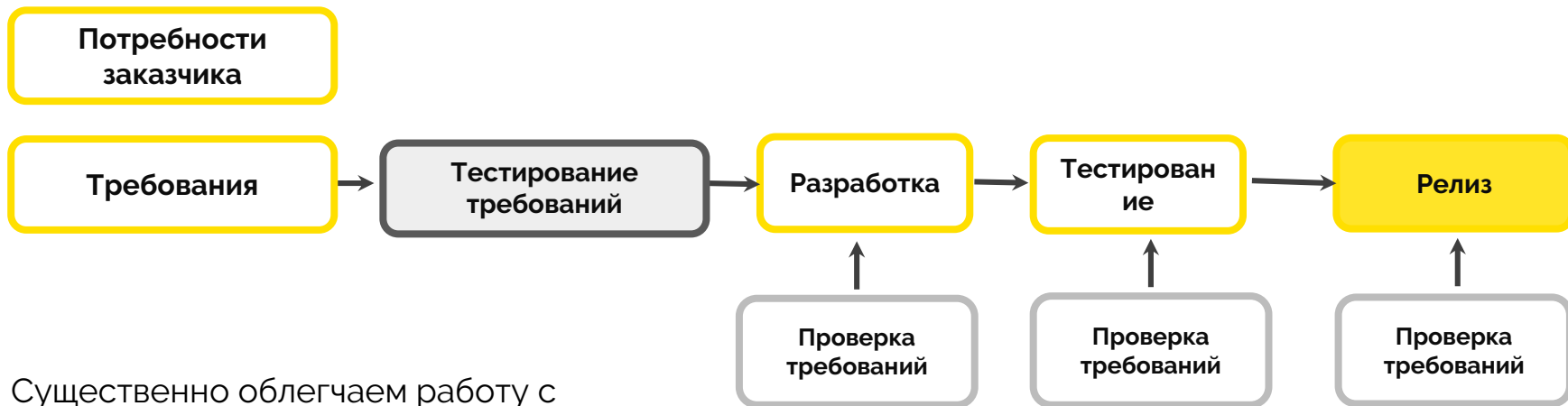


Проверка требований происходит даже тогда, когда нет полноценного тестирования.


Больше времени уходит на «вхождение в тему» для каждого этапа, сохраняется вероятность сделать не то и не так.

Тестирование требований

Добавили этап тестирования требований



Существенно облегчаем работу с требованиями на всех последующих этапах, снижаем риски



Требования к требованиям

1. Полнота (завершённость)
2. Непротиворечивость
3. Однозначность (ясность)
4. Атомарность
5. Корректность и согласованность
6. Выполнимость
7. Актуальность
8. Тестируемость
9. Прослеживаемость
10. Модифицируемость
11. Упорядоченность

Уровни требований





Техники тестирования требований

Рисунки

(графическое представление)

1

Вопросы

(повторное выявление требований)

2

Тест-кейсы и чек-листы

3

Исследование поведения системы

4

Прототипирование

5

Взаимный просмотр:

- Беглый просмотр
- Технический просмотр
- Формальная инспекция

6



Этап 0. Планирование

Прежде всего нужно определить, чего именно мы хотим достичь тестированием требований и сколько времени готовы на это выделить.

- Однозначность, понятность, поддерживаемость требований
- Исключение противоречий в рамках новых требований и существующих функциональностей.
- Начало формирования чек-листов и тест-кейсов, подготовка к задаче до её готовности к тестированию для экономии времени.
- Оценка необходимых для тестирования инструментов и тестовых данных.

Этап 2. Вычитка требований



Атомарность




Непротиворечивость внутри себя



Недвусмысленность



Прослеживаемость



Этап 2:1. Вычитка.

Структура текста

Атомарность. Требование нельзя разбить на отдельные требования без потери завершённости и оно описывает одну и только одну ситуацию.

Прослеживаемость. Требования структурированы, при наличии упоминаний других требований или документов – указаны ссылки на них.

Модифицируемость. При доработке требований искомую информацию легко найти, а её изменение не приводит к нарушению какого-либо из критериев требований



Этап 2.2. Вычитка. Корректность текста

Обязательность и актуальность. Требования не содержат в себе информации или описания кейсов, добавленных «на всякий случай». Требования соответствуют последним договорённостям, а неактуальные - удалены

Непротиворечивость внутри себя. В рамках требований нет противоречащих друг другу пунктов.

Однозначность. Требования содержат в себе только однозначно понимаемые утверждения. Не используется «общепринятость» и «очевидность»

Этап 3. Исследование системы



Полнота



Непротиворечивость



Обязательность



Реализуемость



Тестируемость



3.1. Исследование: полнота функциональных требований

Инструменты для проверки:

- Диаграмма состояний и переходов
- Таблица принятия решений
- Use case
- Макеты, прототипы
- Графические представления

3.3 Исследование: нефункциональные требования



Безопасность



Быстродействие



Факторы эксплуатации



Масштабируемость



Надежность

3.4. Исследование: тестируемость



Доступы к нужным системам



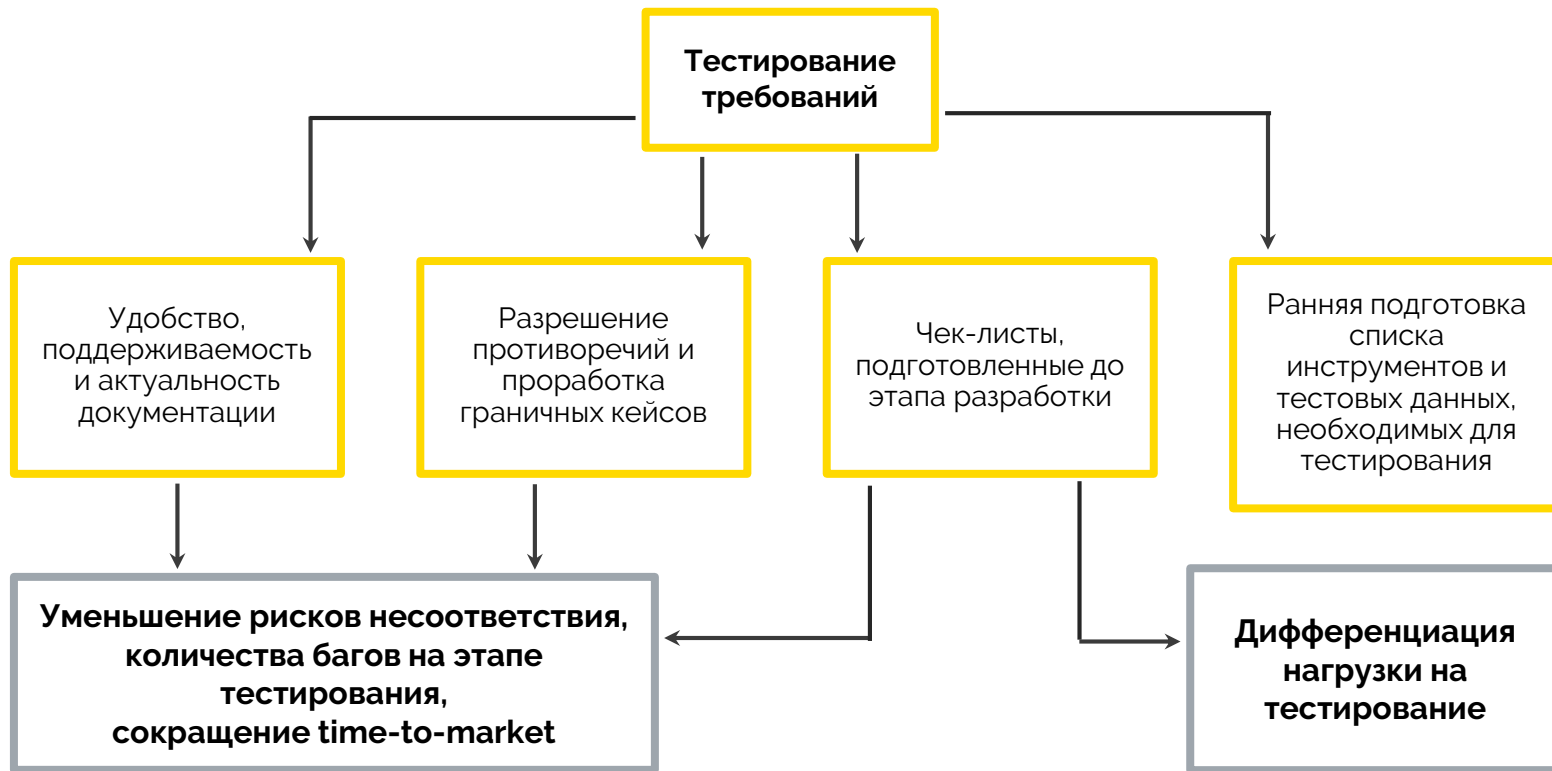
Наличие необходимых
инструментов



Необходимые тестовые
данные

Этап	Уровень требований	Инструменты тестирования	Ключевые критерии	Результат
1. Определение бизнес-требований	Бизнес-требования	Исследование системы	Полнота, реализуемость	Понимание сути задачи; Приёмка тз
2.1 Вычитка : структура		Текстовый редактор	Атомарность, недвусмысленность, отслеживаемость	Удобные в управлении и понимании требования.
2.2 Вычитка : логика	Пользовательские требования	Вопросы	Актуальность, однозначность	Исключение двоякого понимания и неточностей; обнаружение грубых ошибок в ТЗ
3.1 Исследование: полнота функциональных требований	функциональные требования	Майндкарты, таблицы перехода состояний, вариации use case	Полнота, тестируемость, непротиворечивость, обязательность	Тестовая документация; инструменты; закрытие граничных и редких кейсов
3.2. Исследование: полнота требований к интерфейсам	Нефункциональные требования	Исследование системы	Полнота, тестируемость	Тестовая документация, инструменты, необходимые для тестирования
3.3 Исследование: нефункциональные требования	Атрибуты качества, бизнес-ограничения	Вопросы	Полнота, тестируемость	Оценка инструментов, необходимых для тестирования.

Итого:





Стратегия тестирования



Стратегия тестирования

Теперь у нас есть понятные требования, пора решить как мы будем проверять качество продукта?

Стратегия тестирования –

описывает основные методы, инструменты и стандарты, которые будут использоваться для обеспечения качества продукта. Она устанавливает цели тестирования, определяет виды тестирования (функциональное, нефункциональное, регрессионное и т. д.), а также описывает роли и ответственности членов команды.

В составлении стратегии тестирования принимают участие:

- Ответственный за тестирование (QA-head/QA-lead/Тестировщик, в зависимости от проекта – чем сложнее тем важнее экспертность сотрудника)
- Руководитель проекта

Стратегия тестирования

Отвечаем на вопросы и фиксируем договоренности в общедоступном для команды месте – на выходе получаем «Стратегию тестирования». Ответить на вопросы придётся не только QA-специалисту, но и руководителю проекта, исходя из доступных ресурсов на проекте, и исходя из требуемого заказчиком качества.

1. Какие типы тестирования будем применять к проекту и почему

Типы	Комментарии
Проверка установки версии	Например для МП и десктоп – проверка поставки клиенту, обновление руками пользователя и т.п. Для web – сохранение поведения, целостность данных.
Тестирование производительности	Зависит от количества пользователей, софта и железа окружения, объёмы данных. Нужно ли тратить время на неё в принципе?
Регрессионное тестирование	Полная проверка всех сценариев и функционала, в том числе от предыдущих версий. Требуется для второй и последующих выпускаемых версий. Для регулярно выпускаемого продукта практически необходимость.

Стратегия тестирования

Какие типы тестирования будем применять к проекту и почему

Типы	Комментарии
Интеграционное тестирование	Определяем количество интеграций по архитектуре решения, какие данные получаем, какие передаём, будем ли проверять API тесты в этом виде тестирования
Кроссбраузерность Кроссплатформерность	Определяем типичный набор браузеров для web-проекта, учитываем специфику (например, возможен локальный для какого-либо предприятия браузер)
Тестирование безопасности	По итогу должно быть понимание кто отвечает за безопасность, насколько система готова к взлому. Проверки различных видов инъекций и хаков.
Функциональное тестирование	Проводится практически всегда, договариваемся кто именно проводит (например, проводят только тестировщики, QA-lead не проводит).
Приёмочное тестирование	Определяем порядок процесса, кто именно проводит такой вид тестирования (чаще всего заказчик).

Стратегия тестирования

Какие типы тестирования будем применять к проекту и почему

Типы	Комментарии
UX/UI тестирование	Предварительно определяем критерии хорошего интерфейса и кто отвечает за дизайн..
Модульное тестирование (UNIT)	Договариваемся, какой процент покрытия кода юнит тестами считается успешным на проекте. Какие разработчики будут их писать, какой процент времени закладывать на написание таких проверок.
Отказ и восстановление	Проверки того как система ведет себя после сбоев и в экстренных обстоятельствах.
Локализация	Какие языки поддерживаются. В каких странах будет использоваться продукт. Будет ли расширение локализаций, нужна ли работа с часовыми поясами.



Стратегия тестирования

2. Определение приоритетов в тестировании

Мы поговорим об определении приоритетов в отдельной главе. На всех проектах случаются форс-мажоры, поэтому полезно иметь шпаргалку с заранее продуманным планом – например, какие блоки тестирования можно опускать и в каком порядке. Допустимо ли оставить только Smoke-тесты или критичный функционал регрессим в любом случае и т.д.

3. Окружения для работы

Необходимо продумать и согласовать с командой, какие окружения требуются для работы, и кто ими будет пользоваться. Как правило, достаточно 2-3 окружений (например, DEV для разработки, QA для черного тестирования, UAT – доступный для заказчика, на нем могут проводиться приёмки и работать бета-тестировщики

Стратегия тестирования

4. Роль тестировщика на проекте

Широкое поле для обсуждения. Руководителю проекта и всей команде нужно понимать ожидания от тестирования. Какая степень компетенции требуется от тестировщика, и какая нагрузка по времени ожидается.

Пример функций роли тестировщика:

- оценка задач спринта на полноту и непротиворечивость,
- оценка трудозатрат на тестирование,
- подготовка чек-листов для тестирования (или тест-кейсов),
- ревью сценариев автотестов,
- написание функциональных автотестов,
- непосредственно ручное тестирование нового функционала
- деплой изменений на окружения,
- актуализации стратегии тестирования и т.д.

Стратегия тестирования

5. Тестовая документация на проекте

Нужно договориться и с ростом проекта пересматривать формат ведения тестовой документации:

- какую тестовую документацию будем вести на проекте
- в какой системе хранить
- будем ли писать тест-кейсы,
- будем ли писать чек-листы
- как часто обновлять эту документацию

Для руководителя проекта **важно понимать время, которое планируют тестировщики на ведение тестовой документации.** В случае нехватки ресурсов допустимо ограничиться написанием только чек-листов. Но для написания автотестов в будущем лучше иметь полноценные тест-кейсы.

Стратегия тестирования

6. Генерация тестовых сценариев и техники тест-дизайна

Условия тестирования зависят от конкретного решения , обязательно нужно обсудить:

какими техниками тест-дизайна имеет смысл пользоваться.

В рамках этого пункта полезно составить список объектов, с которыми работает приложение, и их классы эквивалентности.

какие предположения и знания о сценариях использования у нас есть, в том числе банальный жизненный опыт помогает придумать сценарии тестирования для конкретного проекта.

Практический опыт: например, для разработки ПО продажи билетов в кассе полезно вместе с операционистом пройти все сценарии на текущем оборудовании вместе: посмотреть работу с разными клиентами и товарами, проверить негативные варианты – медленный интернет или его отсутствие; кончилась печатная лента; выяснить другие нетиповые случаи из практики.

Стратегия тестирования

7. Flow работы в трекере

Договариваемся о том, как работать в трекере, как отличать приоритеты, как движется задача по статусам и т.п. со всей командой. Об этом будет отдельная глава в лекциях.

8. Инструменты для работы

На этапе планирования тестирования можно составить перечень задач для инфраструктурного обеспечения выбранных инструментов (спланировать время серверных и DevOps администраторов. Так мы повышаем вероятность начать тестирование в полной боевой готовности.

Стоит обсудить такие вопросы:

1. Какие инструменты нам нужны для ведения тестовой документации,
2. Какими инструментами можно подготовить тестовые данные,
3. Какие инструменты нужны для автоматизации (E2E, API, Интеграционного)

Стратегия тестирования

9. Критерии начала и окончания тестирования

Договариваемся в какой момент времени можно брать следующую задачу и при каких условиях. Задаём формальные критерии, которые могут выглядеть так:

Задача считается полностью протестированной, если выполнены все условия:

**Чек-лист составлен
и пройдены** все его
пункты

**Ссылка на результаты
прохождения** чек-
листа прикреплена в
исходную задачу

Исходная задача на
разработку нового
функционала
находится в статусе
Resolved

Таски на тестирование
исходной задачи
находятся **в статусе**
Closed

Стратегия тестирования

Общие моменты:

Универсальной
стратегии нет, это
нормально, она
формируется под
каждый проект

1

Стратегия — это
только инструмент,
который позволяет
достичь наших
целей, а не
самоцель.

2

Вполне нормально, что **не
сразу удастся ответить на
все поставленные вопросы.**

Это повод улучшить
коммуникацию с заказчиком,
командой.

3

Проект изменяется со временем,
масштабируется, меняется
профиль пользователей,
требования к безопасности – с
учетом этого **нужно будет
пересматривать стратегию
тестирования.**

4

**После написания
стратегии** обычно
становится **понятно,**
**где есть провалы в
тестировании** и где
нужны улучшения.

5

Вопросы



N*

**Спасибо
за внимание**

ФИТ Лекция №8. 25'