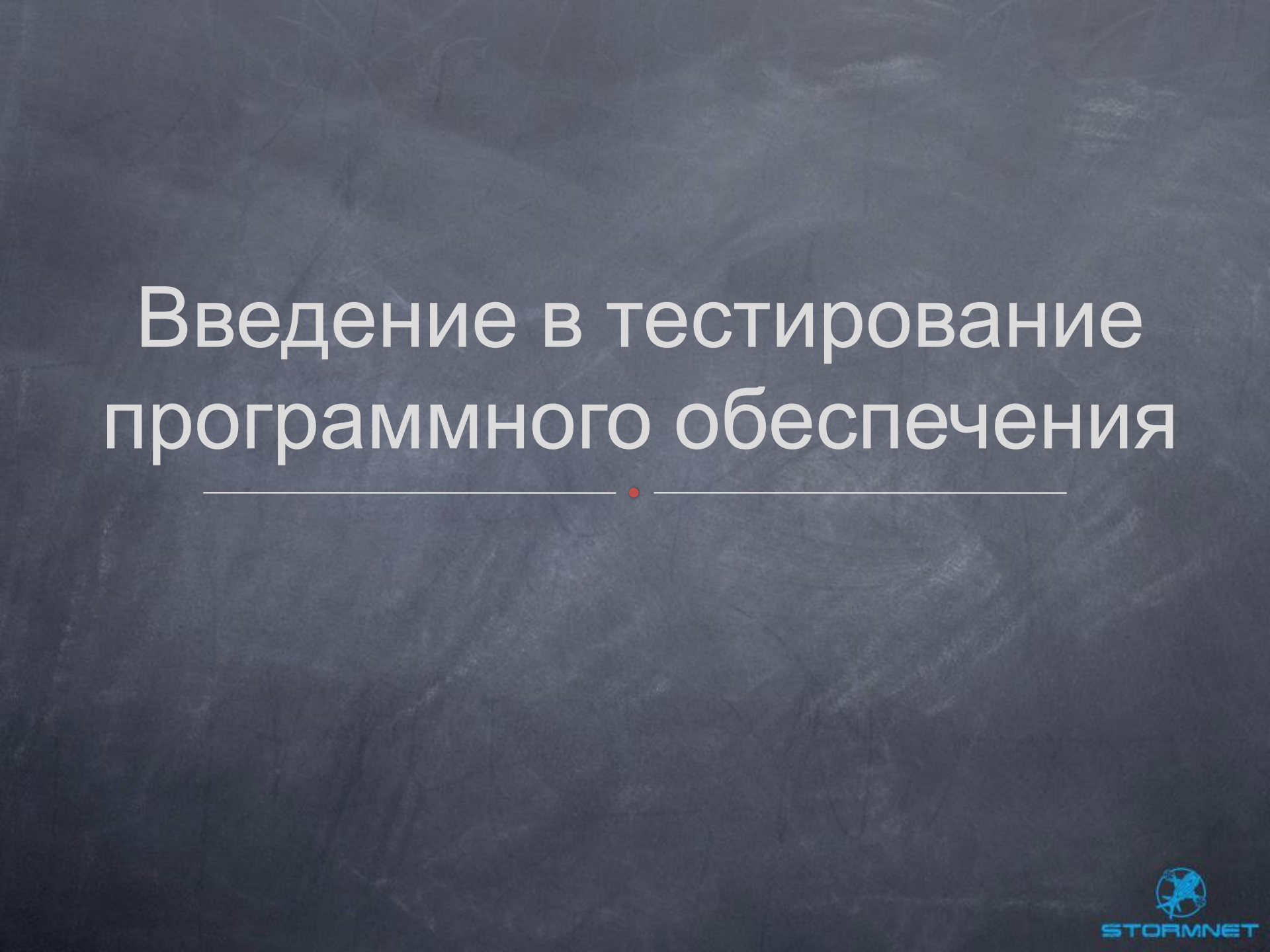
Введение в тестирование программного обеспечения



Введение. С чего всё начиналось?

- только крупные фирмы и институты
- всего единицы продуктов в год
- сверхпрофессиональность кадров
- низкоуровневые языки разработки ПО

Основные «эпохи тестирования»

- В **50-60-х годах** тестирование представляло собой скорее отладку программ (debugging). Существовала концепция т.н. «исчерпывающего тестирования (exhaustive testing)» — проверки всех возможных путей выполнения кода со всеми возможными входными данными. Однако очень скоро было выяснено, что исчерпывающее тестирование невозможно, т.к. количество возможных путей и входных данных очень велико, а также при таком подходе сложно найти проблемы в документации.

Основные «эпохи тестирования»

- **В 70-х годах** фактически родились две фундаментальные идеи тестирования: тестирование сначала рассматривалось как процесс доказательства работоспособности программы в некоторых заданных условиях (positive testing), а затем – строго наоборот: как процесс доказательства неработоспособности программы в некоторых заданных условиях (negative testing).
- **!!! Неверное утверждение:** Негативные тест-кейсы должны заканчиваться возникновением сбоев и отказов в приложении.
- !!! **Верное утверждение:** Негативные тест-кейсы

Основные «эпохи тестирования»

- В **80-х годах** произошло ключевое изменение места тестирования в разработке ПО: вместо одной из финальных стадий создания проекта тестирование стало применяться на протяжении всего цикла разработки (software lifecycle)
- А также появились первые элементарные попытки автоматизировать тестирование
- В **90-х годах** произошёл переход от тестирования как такового к более всеобъемлющему процессу, который называется «обеспечение качества (quality assurance)», охватывает весь цикл разработки ПО и затрагивает процессы планирования, проектирования, создания и

Основные «эпохи тестирования»

- В 2000-х годах развитие тестирования продолжалось в контексте поиска всё новых и новых путей, методологий, техник и подходов к обеспечению качества. Серьёзное влияние на понимание тестирования оказало появление гибких методологий разработки.
- Автоматизация тестирования уже воспринималась как обычная неотъемлемая часть большинства проектов, а также стали популярны идеи о том, что во главу процесса тестирования следует ставить не соответствие программы требованиям, а её способность предоставить конечному пользователю

Основные «эпохи тестирования»

- О **современном этапе** развития тестирования мы будем говорить на протяжении всего курса. Основные характеристики:
  - гибкие методологии и гибкое тестирование
  - глубокая интеграция с процессом разработки
  - широкое использование автоматизации
  - колоссальный набор технологий и инструментальных средств
  - кросс-функциональность команды.

Появление ПК – революция в области разработки ПО

- Проникновение компьютеризации во все сферы жизнедеятельности
- Увеличение количества фирм-разработчиков
- Постоянный рост создаваемых программ
- Пересмотр подхода к обеспечению качества и надежности программ

Рост числа фирм – рост конкуренции

- Потребитель выбирает разработчика, обещающего оптимальное сочетание цены, времени разработки, качества продукта.
- Разработчик ищет новые способы обхода конкурентов
- Тестирование как конкурентное преимущество
  Всё больше и больше программ в наше время должны быть безупречными – Современные программы управляют оборудованием больниц, аэропортов, атомных реакторов, космических кораблей

Тестирование и качество

- **Тестирование программного обеспечения (Software Testing)** – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.
- В более широком смысле, **тестирование** – это одна из техник контроля качества, включающая в себя активности по планированию работ (**Test Management**), проектированию тестов (**Test Design**), выполнению тестирования (**Test Execution**) и анализу полученных результатов (**Test Analysis**).
- **Качество программного обеспечения** – это совокупность характеристик ПО, относящихся к

Объекты тестирования

- Программы при их непосредственном запуске и исполнении (software).
- Код программ без запуска и исполнения (code).
- Прототип программного продукта (product prototype).
- Проектную документацию (project documentation):
  - Требования к программному продукту (product requirements).
  - Функциональные спецификации к программному продукту (functional specifications).
  - Документы, описывающие архитектуру (product architecture), дизайн (product design).
  - План проекта (project plan) и тестовый план (test plan).
  - Тестовые сценарии (test cases).
- Сопроводительную документацию (и документацию для пользователей):
  - Интерактивную помощь (on-line help).
  - Руководства по установке (Installation guide) и использованию программного продукта (user manual).
- Необходимость тестирования того или иного рабочего продукта (work product) определяется конкретным проектом и условиями его разработки.

Характеристики качества (ISO 9126-1)

- Функциональные возможности
- Функциональная пригодность
- Правильность (корректность)
- Способность к взаимодействию
- Надежность
- Эффективность
- Защищенность
- Портативность (мобильность)
- Практичность
- Удобство использования
- Удобство сопровождения

Функциональность

- **Функциональные возможности** – способность программного средства обеспечивать решение задач, удовлетворяющих сформулированным потребностям заказчиков и пользователей при применении комплекса программ в заданных условиях.
- **Функциональная пригодность** – набор атрибутов, определяющих назначение, номенклатуру, основные, необходимые и достаточные функции программного средства, соответствующие техническому заданию и спецификациям требований заказчика или потенциального пользователя.
- ПО решает прикладные задачи заказчика/пользователя, для того продукт и используется пользователями. Более того, ПО используется в какой-то конкретной области. Продукт не может быть всюду применимым.

Правильность

- **Правильность (корректность)** – способность программного средства обеспечи вать правильные или приемлемые для пользователя результаты и внешние эффекты.

Взаимодействие

- **Способность к взаимодействию** — свойство программных средств и их компонентов взаимодействовать с одной или большим числом компонентов внутренней и внешней среды.

Надежность

- **Надёжность** – обеспечение комплексом программ достаточно низкой вероятности отказа в процессе функционирования программного средства в реальном времени.
- Надежность – способность быть отказоустойчивым
- Как тестировщики мы ожидаем, после всех наших проверочных работ отказ оборудования является маловероятным событием.

Эффективность

- **Эффективность** – свойства программного средства, обеспечивающие требуемый уровень производительности решения функциональных задач, с учётом количества используемых вычислительных ресурсов в установленных условиях.
- Выдает оптимальное сочетание производительности приложения и затрачиваемых ресурсов системы.

Защищенность

- **Защищённость** – способность компонентов программного средства защищать программы и информацию от любых негативных воздействий.

Портативность (мобильность)

- **Мобильность** – подготовленность программного средства к переносу из одной аппаратно-операционной среды в другую, т.е. способность менять среды эксплуатации.
- По-другому, переносимость между операционными системами, оборудованием в целом

Практичность

- **Практичность (применимость)** – свойства программного средства, обусловливающие сложность его понимания, изучения и использования, а также привлекательность для квалифицированных пользователей при применении в указанных условиях.

Удобство использования

- Легкость понимания, изучения, использования (логичность построения, интуитивность интерфейса)
- Привлекательность продукта (соответствие новым веяниям в графическом представлении)

Удобство сопровождения

- **Сопровождаемость** – приспособленность программного средства к модификации и изменению конфигурации и функций.
- Добавление новой функциональности – обычное дело для приложений в промышленной эксплуатации.
- Важно, чтобы приложение не имело внутренних барьеров против развития.

Модель качества программного обеспечения (ISO/IEC 9126)

STORMNET

Направления тестирования

- Статическое (без запуска программного кода продукта)
- Динамическое (при запущенной системе)

Статическое тестирование

– раннее выявление дефектов (дешевле)

Проверяем:

- код
- требования
- спецификации
- архитектуру
- дизайн

Динамическое тестирование

- Более позднее выявление/более дорогое устранение дефектов (спецификации написаны, архитектура разработана, интерфейс программы тоже готов)

Проверяем:

- все характеристики качества запущенного приложения

Этапы тестирования

- Анализ требований
- Планирование испытаний
- Проектирование тестов
- Запуск тестов
- Редактирование тестов
- Системное тестирование
- Приемочные испытания
- Эксплуатация и сопровождение

Анализ требований

- Изучаем спецификации требований
- Изучаем функциональные требования к системе
- Отвечаем на вопросы:
  - что нам предстоит тестировать;
  - как много будет работы;
  - какие есть сложности;
  - всё ли необходимое у нас есть и т.п.
- Получаем данные, по которым далее составляем план проведения испытаний

Планирование испытаний

- Определяем объемы испытаний и ресурсы
- Пишем расписание того, когда будем выполнять намеченые действия

Проектирование тестов

Определяем:

- Цель тестирования
- Спецификацию входных данных
- Архитектуру тестов (для упорядочения по группам)

- Пишем тесты

Запуск тестов

- Проверяем наши тесты в действии
- Анализируем тестовые случаи

Редактирование тестов

- Пересматриваем и корректируем тесты

Системное тестирование

- Проверяем всю систему
- Получаем сведения о качестве характеристик ПО

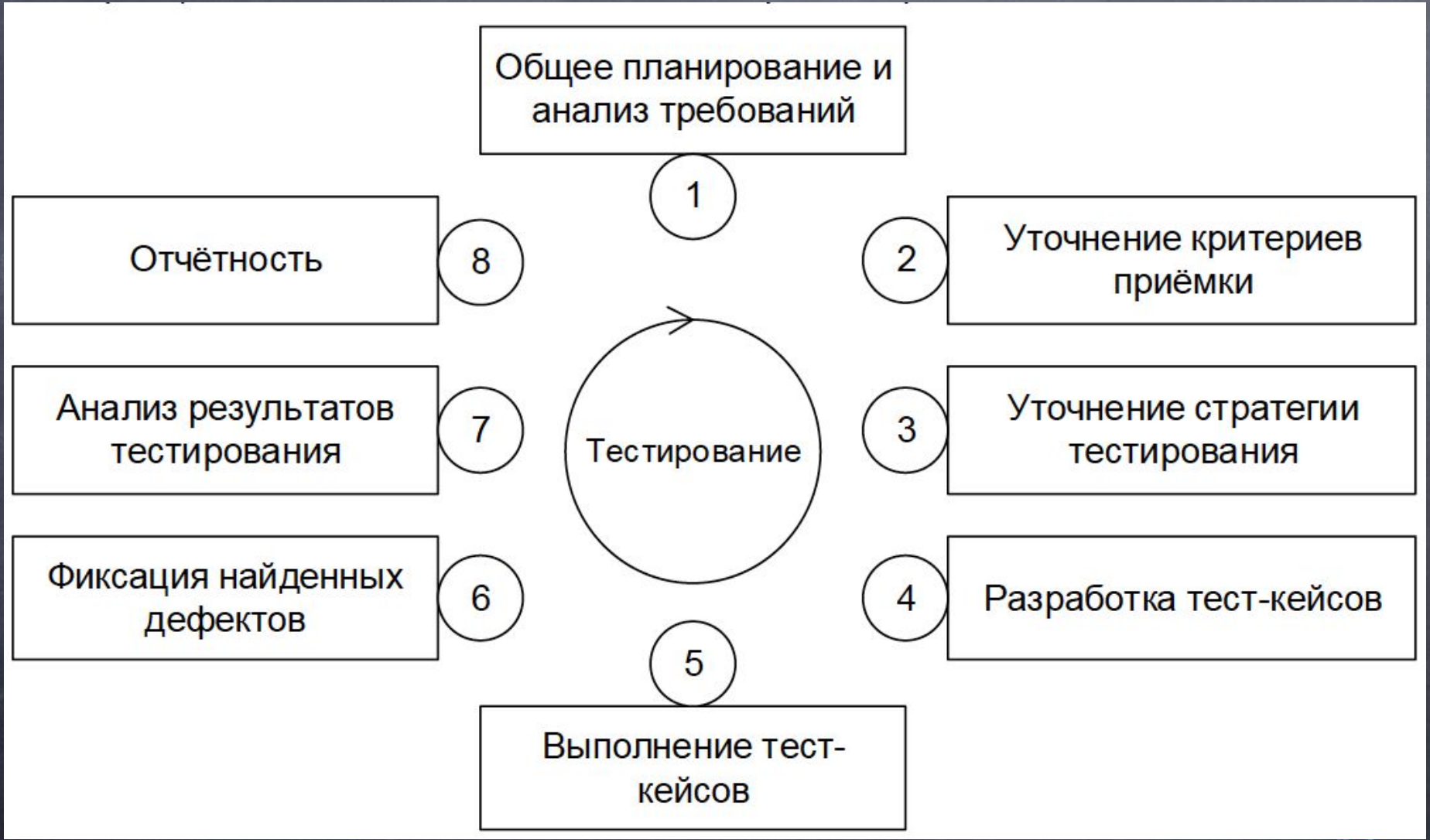
Приемочные испытания

- Альфа-тестирование
- Бета-тестирование

Поддержка и сопровождение

- Проверяем качество исправлений дефектов
- Проводим регрессионные тесты

Жизненный цикл тестирования



Методы тестирования

Мы работаем с кодом системы?

- НЕТ = метод «черного» ящика (*black-box*)
- Частично = метод «серого» ящика (*grey-box*)
- ДА = метод «белого» ящика (*white-box*)

Уровни тестирования

Над чем мы проводим тесты (модуль, группа модулей, система) «вширь»

- Компонентное/модульное (*component/unit testing*)
- Интеграционное (*integration testing*)
- Системное (*system testing*)

А также вторая классификация «вглубь»:

- Приемочное (smoke test, *acceptance testing*)
- Тест критического пути (critical path test)
- Расширенный тест (extended test)

Виды тестирования

Виды тестирования по преследуемым целям разбиваются на группы:

- Функциональные виды
- Нефункциональные виды
- Виды, связанные с изменениями

Функциональные виды

- Тестируем функции и взаимодействия с другими системами
- Пишем тесты на всех уровнях тестирования (компонентное / интеграционное / системное; приемочное/ тестирование критического пути/расширенное тестирование)
- Изучаем, как внешне ведет себя система

Функциональные виды:

- Функциональное тестирование (functional testing)
- Тестирование новой функциональности (new feature testing)
- Тестирование безопасности (security testing)

Нефункциональные виды

Тестируем то, «как» система работает

Виды:

1. Тестирование производительности (performance testing):

A) нагрузочное (*performance & load testing*)

B) стрессовое (*stress testing*)

C) тестирование стабильности / надежности (*stability / reliability testing*)

D) тестирование объемами (*volume testing*)

2. Установочное тестирование (*installation testing*);
3. Тестирование удобства пользования (*usability testing*);
4. Тестирование на отказ и восстановление (*failover & recovery testing*);

производительности (performance testing)

- **Тестирование производительности** ***(performance testing)*** – тестирование, которое проводится с целью определения, как быстро работает система или её часть под определённой нагрузкой.

Нагрузочное (performance & load testing) тестирование

- **Нагрузочное тестирование (performance & load testing)** – это автоматизированное тестирование, которое имитирует одновременную работу множества пользователей над тестируемым приложением.

Стрессовое тестирование (stress testing)

- **Стрессовое тестирование** (stress testing) позволяет проверить насколько приложение и система в целом работоспособны в условиях стресса и также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию после прекращения воздействия стресса.

  ! Целью стрессового тестирования (stress testing) является оценка работоспособности системы в условиях повышенных и предельных нагрузок.

Тестирование стабильности / надежности (*stability / reliability testing*)

- **Тестирование стабильности / надежности** (***stability / reliability testing***) – проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.

Тестирование объемами (*volume testing*)

- Задачей объемного тестирования является получение оценки производительности при увеличении объемов данных в базе данных приложения, при этом происходит:
  - измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций
  - может производиться определение количества пользователей, одновременно работающих с приложением

Тестирование документации (document testing)

- **Тестирование документации** (document testing) – вид тестирования, с которого начинается почти любой проект. Призвано обнаружить ошибки в документации. Эти ошибки опасны тем, что они как маленький комок снега могут вызвать лавину проблем, вырастая на более поздних стадиях работы с проектом в очень сложноустранимые и дорогостоящие последствия.

Установочное тестирование (*installation testing*)

- **Тестирование установки** направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

Тестирование удобства пользования (usability testing)

- **Тестирование удобства пользования (usability testing)** – это вид тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Тестирование на отказ и восстановление (*failover & recovery testing*)

- **Тестирование на отказ и восстановление** (**Failover and Recovery Testing**) проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказ сети).

тестирование(*configuration testing*)

- **Конфигурационное тестирование** (**Configuration Testing**) — специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.)

Тестирование совместимости (compatibility testing)

- **Тестирование совместимости** (compatibility testing) – вид нефункционального тестирования, основной целью которого является проверка корректной работы продукта в определенном окружении. Окружение может включать в себя следующие элементы:
  - Аппаратная платформа;
  - Сетевые устройства;
  - Периферия (принтеры, CD/DVD-приводы, веб-камеры и пр.);
  - Операционная система (Unix, Windows, MacOS, ...)
  - Базы данных (Oracle, MS SQL, MySQL, ...)
  - Системное программное обеспечение (веб-сервер, антивирус, ...)

интернационализации (internationalization testing)

- **Тестирование интернационализации (internationalization testing)** – проверяет готовность приложения к работе с различными языковыми интерфейсами. В частности, проверяется способность корректно отображать шрифты, пункты меню, производить поиск, сортировку, способность приложения обрабатывать файлы, поименованные на различных языках. Следующей стадией, как правило, является локализационное тестирование.

Локализационное тестирование (localization testing)

- **Локализационное тестирование** (localisation testing) – проверяет, насколько корректно продукт адаптирован к работе на том или ином языке: всё ли переведено и переведено правильно, не нарушилась ли логика построения интерфейса и обработки данных и т.д. Для локализационного тестирования рекомендуется обязательно приглашать в команду носителя того языка, перевод на который тестируется. В противном случае мы рискуем увидеть нечто подобное:
  Пункт меню: «Сделать под себя» («Customise»)
  Огромная красная кнопка «ВСЁ!» («Finish»)
  Пункт меню «Ясные печенья» («Clear cookies»)

Связанные с изменениями виды

- Починен баг – необходимо проверить воспроизводится он или нет
- Виды:
  - Дымовое тестирование (*smoke testing*)
  - Регрессионное тестирование (*regression testing*)
  - Тестирование сборки (*build verification testing*)
  - Проверка согласованности/исправности (*sanity testing*)

Smoke Test

- **Smoke Test** – короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции
- **Smoke Test** – выполнение минимального набора тестов для выявления явных ошибок критичного функционала

Регрессионное тестирование

Регрессионное тестирование – вид тестирования, направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде.

Тестирование сборки

- **Тестирование сборки** – тестирование направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования