Виды и уровни тестирования ПО

Адукар

Требования

1. Свойство программного обеспечения, необходимое пользователю для решения проблемы при достижении поставленной цели.
2. Свойство программного обеспечения, которым должна обладать система или её компонент, чтобы соответствовать контракту, стандарту, спецификации, либо иной формальной документации.

Бизнес/ пользовательские
Функциональные/ нефункциональные
Прямые/косвенные

QA|QC|Тестирование

Качество программного обеспечения (Software Quality) - это степень, в которой программное обеспечение обладает требуемой комбинацией свойств.

Качество программного обеспечения (Software Quality) - это совокупность характеристик программного обеспечения, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

https://ru.wikipedia.org/wiki/ISO_9126 http://docs.cntd.ru/document/gost-r-iso-mek-9126-93

Обеспечение качества (Quality Assurance - QA) - это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации программного обеспечения (ПО) информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО для обеспечения требуемого уровня качества выпускаемого продукта.

Контроль качества (Quality Control - QC) - это совокупность действий, проводимых над продуктом в процессе разработки для получения информации о его актуальном состоянии в разрезах: "готовность продукта к выпуску", "соответствие зафиксированным требованиям", "соответствие заявленному уровню качества продукта".

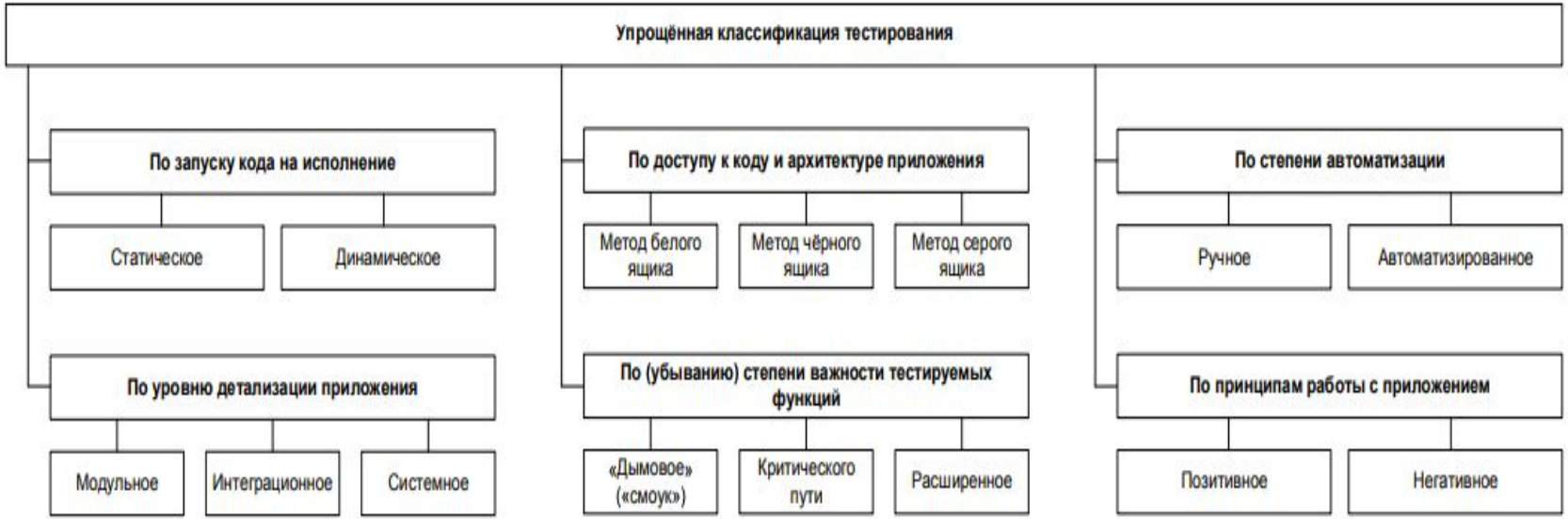
Тестирование программного обеспечения (Software Testing) - это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

Тестирование

Тестирование (software testing) – деятельность, выполняемая для оценки и улучшения качества программного обеспечения. Эта деятельность, в общем случае, базируется на обнаружении дефектов и проблем в программных системах.

Тестирование программных систем состоит из динамической ***верификации*** поведения программ на конечном (ограниченном) наборе тестов (set of test cases), выбранных соответствующим образом из обычно выполняемых действий прикладной области и обеспечивающих проверку соответствия ожидаемому поведению системы.

Тестирование делится на следующие виды:



По запуску кода на исполнение

1. Статическое - без запуска
2. Динамическое - с запуском

Статическое тестирование - без запуска

В рамках этого подхода тестированию могут подвергаться:

- Документы (требования, тест-кейсы, описания архитектуры приложения, схемы баз данных и т.д.).
- Графические прототипы (например, эскизы пользовательского интерфейса).
- Код приложения.

Динамическое тестирование — тестирование с запуском кода

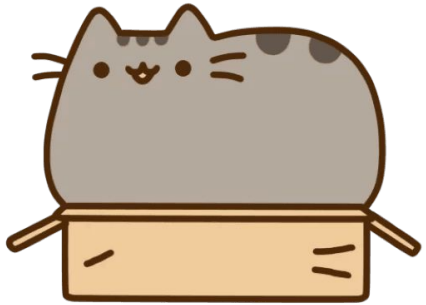
Динамическое тестирование — тестирование с запуском кода на исполнение.

Запускаться на исполнение может как код всего приложения целиком, так и код нескольких взаимосвязанных частей, отдельных частей и даже отдельные участки кода.

Основная идея этого вида тестирования состоит в том, что проверяется реальное поведение приложения.

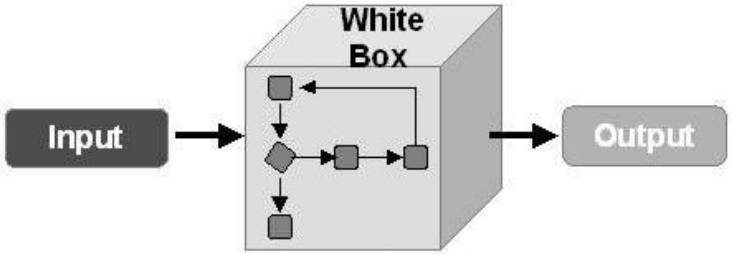
По доступу к коду

1. Метод белого ящика - с доступом к коду
2. Метод серого ящика - доступ к части кода есть, а к другой части нету
3. Метод черного ящика - без доступа к коду



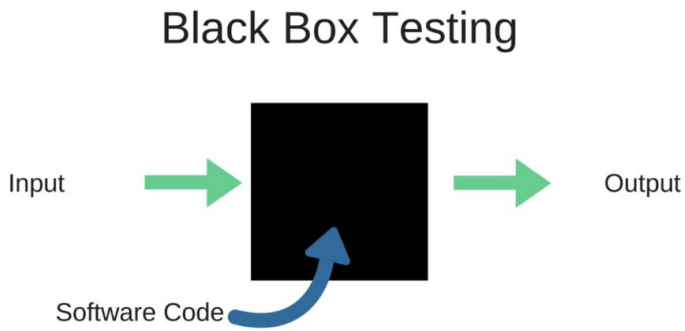
Метод белого ящика - с доступом к коду

Метод белого ящика (white box) — у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.



Метод черного ящика - без доступа к коду

Метод чёрного ящика — у тестировщика либо нет доступа к внутренней структуре и коду приложения, либо недостаточно знаний для их понимания, либо он сознательно не обращается к ним в процессе тестирования.



Метод серого ящика - доступ только к части кода

Метод серого ящика - комбинация методов белого ящика и черного ящика, состоящая в том, что к части кода и архитектуры у тестировщика доступ есть, а к части — нет. Обычно говорят о методах белого или чёрного ящика в применении к тем или иным частям приложения, при этом понимая, что «приложение целиком» тестируется по методу серого ящика.

По степени автоматизации

1. Автоматизированное — тест-кейсы частично или полностью выполняет специальное инструментальное средство.
2. Ручное - все проверки выполняет человек

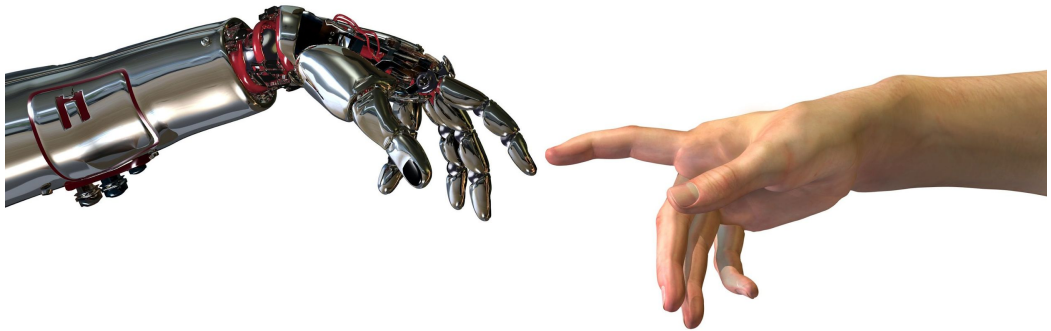
Автоматизированное — тест-кейсы выполняет инструментальное средство

Автоматизированное тестирование — набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования.

```
WebElement element = driver.findElement(By.tagName("a"));
element.click();
```

Ручное - все проверки выполняет человек

Ручное тестирование — тестирование, в котором тест-кейсы выполняются человеком вручную без использования средств автоматизации.

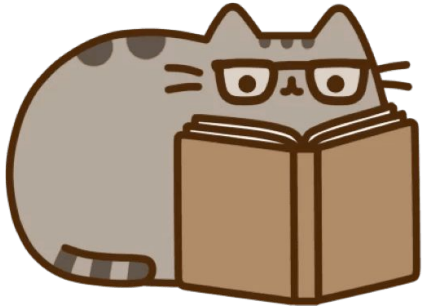


По уровню детализации приложения/уровни тестирования

1. Модульное - проверяются отдельные небольшие части приложения.
2. Интеграционное - проверяется взаимодействие между несколькими частями приложения
3. Системное - приложение проверяется как единое целое

Модульное - отдельные небольшие части приложения

Модульное (компонентное) тестирование направлено на проверку отдельных небольших частей приложения, которые можно исследовать изолированно от других подобных частей.



Интеграционное - взаимодействие частей приложения

Интеграционное тестирование (integration testing) направлено на проверку взаимодействия между несколькими частями приложения

Тестирование интеграции компонентов - тестирование взаимодействия между несколькими интегрированными компонентами одного приложения.

Системное интеграционное тестирование (system integration testing) - тестирование взаимодействия между различными приложениями.

Основная задача- поиск дефектов, в реализации и интерпретации взаимодействия между модулями.

Системное тестирование - приложение проверяется как единое целое

Системное тестирование направлено на проверку всего приложения как единого целого, собранного из частей, проверенных на двух предыдущих стадиях.

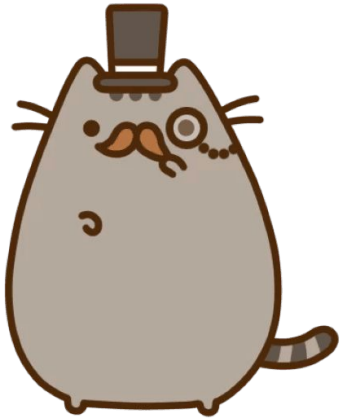
На уровне системного тестирования не только выявляются дефекты «на стыках» компонентов, но и появляется возможность полноценно взаимодействовать с приложением с точки зрения конечного пользователя.

По степени важности тестируемых функций

1. **Дымовое тестирование (smoke test)** - тестируем только самый важный функционал
2. **Тестирование критического пути (critical path test)** - тестируем функционал, которым типичный пользователь пользуется в типичных ситуациях
3. **Расширенное тестирование (extended test)** - проверка остальной (всей) функциональности

Smoke testing - только самый важный функционал

Дымовое тестирование направлено на проверку самой главной, самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной саму идею использования приложения (или иного объекта, подвергаемого дымовому тестированию)

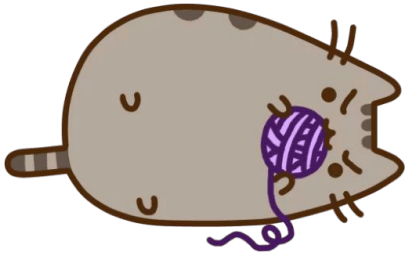


Critical path testing - типичный функционал

Тестирование критического пути направлено на исследование функциональности, используемой **типичными** пользователями в **типичной** повседневной деятельности.

Extended testing - остальная функциональности

Расширенное тестирование направлено на исследование всей заявленной в требованиях функциональности — даже той, которая низко проранжирована по степени важности.



По принципу работы с ПО

1. **Позитивное** - все действия с приложением выполняются строго по инструкции без никаких недопустимых действий, некорректных данных
2. **Негативное** - в работе с приложением выполняются (некорректные) операции и используются данные, потенциально приводящие к ошибкам

Позитивное - строго по инструкции без некорректных данных

Позитивное тестирование направлено на исследование приложения в ситуации, когда все действия выполняются строго по инструкции без каких бы то ни было ошибок, отклонений, ввода неверных данных и т.д.

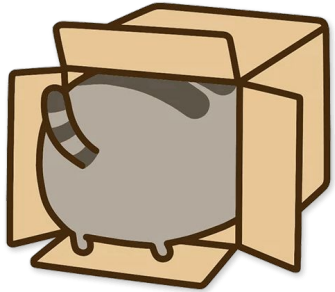
Негативное - выполняются “некорректные” операции

Негативное тестирование — направлено на исследование работы приложения в ситуациях, когда с ним выполняются (некорректные) операции и/или используются данные, потенциально приводящие к ошибкам (классика жанра — деление на ноль).

Внимание! Очень частая ошибка!

Негативные тесты НЕ предполагают возникновения в приложении ошибки. Напротив — они предполагают, что верно работающее приложение даже в критической ситуации поведёт себя правильным образом (в примере с делением на ноль, например, отобразит сообщение «Делить на ноль запрещено»).

Если есть пакет “негативных” и “позитивных” проверок с какого начинаем?



Регрессионное тестирование (regression testing)

Регрессионное тестирование (regression testing) — тестирование, направленное на проверку того факта, что в ранее работоспособной функциональности не появились ошибки, вызванные изменениями в приложении или среде.

“Исправление одной ошибки с большой вероятностью влечёт появление новой».

Потому регрессионное тестирование является неотъемлемым инструментом обеспечения качества и активно используется практически в любом проекте.

Чаще всего, Регрессионное тестирование проводится перед Релизом

Когда проводят регрессию

- Добавление нового (крупного) функционала
- Крупный баг-фиксинг
- Перед релизом

Приемочное тестирование (acceptance tests)

Приемочное (acceptance) – вид тестирования, проводимый на этапе сдачи готового продукта (или готовой части продукта) заказчику. Целью приемочного тестирования является определение готовности продукта, что достигается путем прохода тестовых сценариев и случаев, которые построены на основе спецификации требований к разрабатываемому ПО.

Результатом приемочного тестирования может стать:

- Отправка проекта на доработку.
- Принятие его заказчиком, в качестве выполненной задачи.

Тестирование на основе тест-кейсов

Подход, в котором тестирование производится на основе заранее подготовленных тест-кейсов, наборов тест-кейсов и иной документации.

Test Case – это тестовый артефакт, суть которого заключается в выполнении некоторого количества действий и/или условий, необходимых для проверки определенной функциональности разрабатываемой программной системы.

Исследовательское тестирование (Exploratory testing)

Подход, в рамках которого тестировщик выполняет работу с приложением по выбранному сценарию, который, в свою очередь, дорабатывается в процессе выполнения с целью более полного исследования приложения.

Свободное тестирование (ad hoc testing)

Подход, в котором не предполагается использование тест-кейсов, чек-листов, сценариев — тестировщик полностью опирается на свой профессионализм и интуицию для спонтанного выполнения с приложением действий, которые, как он считает, могут обнаружить ошибку.

Тестирование совместимости (compatibility testing)

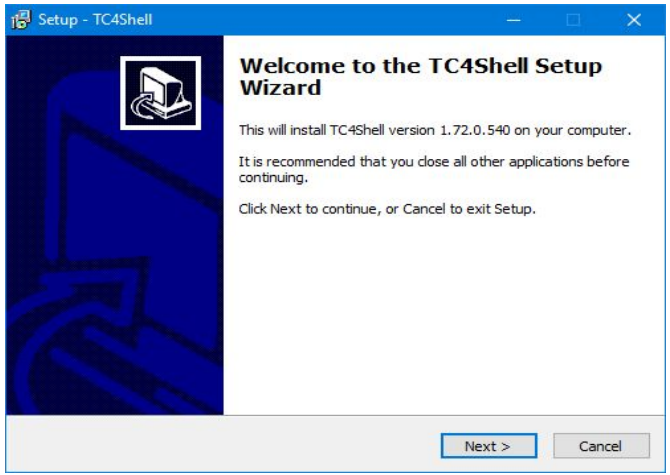
Тестирование, направленное на проверку способности приложения работать в указанном окружении.
Например:

- Совместимость с аппаратной платформой, операционной системой
- Совместимость с браузерами и их версиями
- Совместимость с мобильными устройствами

Инсталляционное тестирование (installation testing)

Инсталляционное тестирование (installation testing) — тестирование, направленное на выявление дефектов, влияющих на протекание стадии инсталляции (установки) приложения.
Такое тестирование проверяет множество сценариев и аспектов работы инсталлятора в таких ситуациях, как:

- Установка
- Удаление
- Апдейт
- Даунгрейт



Функциональное тестирование

Функциональность (Functionality) - определяется способностью ПО решать задачи, которые соответствуют зафиксированным и предполагаемым потребностям пользователя, при заданных условиях использования ПО. Т.е. эта характеристика отвечает то, что ПО работает исправно и точно, функционально совместимо соответствует стандартам отрасли и защищено от несанкционированного доступа

Функциональное тестирование (functional testing) - тестирование, основанное на анализе спецификации функциональности компонента или системы.

Проверка того, что актуальный результат соответствует ожидаемому и здравому смыслу. Отвечает на вопрос “Что делает?”

Нефункциональное тестирование

Нефункциональное тестирование (non-functional testing) - тестирование атрибутов компонента или системы, не относящихся к функциональности. Отвечает на вопрос “Как ПО это делает?”

К не функциональному тестированию относится

Надежность (Reliability) – способность ПО выполнять требуемые задачи в обозначенных условиях на протяжении заданного промежутка времени или указанное количество операций. Атрибуты данной характеристики – это завершенность и целостность всей системы, способность самостоятельно и корректно восстанавливаться после сбоев в работе, отказоустойчивость.

Удобство использования (Usability) – возможность легкого понимания, изучения, использования и привлекательности ПО для пользователя.

Эффективность (Efficiency) – способность ПО обеспечивать требуемый уровень производительности, в соответствии с выделенными ресурсами, временем и другими обозначенными условиями.

Удобство сопровождения (Maintainability) – легкость, с которой ПО может анализироваться, тестироваться, изменяться для исправления дефектов для реализации новых требований, для облегчения дальнейшего обслуживания и адаптирования к имеющемуся окружению.

Портативность (Portability) – характеризует ПО с точки зрения легкости его переноса из одного окружения (software/ hardware) в другое.

UI тестирование

UI тестирование, направлено на проверку интерфейсов приложения или его компонентов.

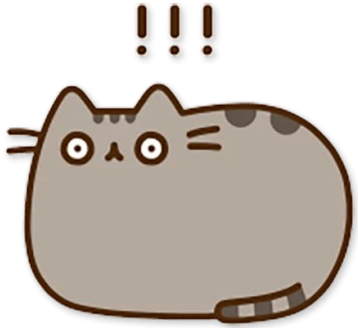
Например:

- орфографические и грамматические ошибки
- неровное расположение полей ввода в формах, самих форм
- неправильное отображение элементов при смене размера окна браузера и масштаба страницы
- изменение размера текста при смене языка
- неровное расположение форм
- разные шрифты
- выбранные элементы не отличаются от невыбранных

Тестирование удобства (usability testing)

Тестирование удобства использования (usability testing) — тестирование, направленное на исследование того, насколько конечному пользователю понятно, как работать с продуктом, а также на то, насколько ему нравится использовать продукт.

Важно! Тестирование удобства интерфейса и тестирование удобства использования — не одно и то же!



Тестирование интернационализации (internationalization testing)

Тестирование, направленное на проверку готовности продукта к работе с использованием различных языков и с учётом различных национальных и культурных особенностей.
Этот вид тестирования не подразумевает проверки качества соответствующей адаптации, оно сфокусировано именно на проверке возможности такой адаптации

Например:

- Что будет, если открыть файл с иероглифом в имени
- Как будет работать интерфейс, если всё перевести на японский
- Может ли приложение искать данные в тексте на корейском и т. д.

Тестирование локализации (localization testing)

Тестирование, направленное на проверку корректности и качества адаптации продукта к использованию на том или ином языке с учётом национальных и культурных особенностей. Это тестирование следует за тестированием интернационализации

Например:

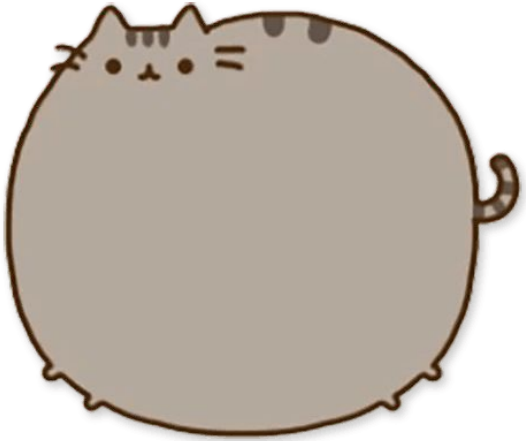
- Перевод
- Картинки
- Время/Дата
- Денежные единицы
- Единицы измерений
- Другие национальные и культурные особенности
- Разметку (если перевести текст на немецкий он будет занимать больше места и может растянуть все таблицы и поля)

Нагрузочное тестирование

Исследование способности приложения сохранять заданные показатели качества при нагрузке в допустимых пределах и некотором превышении этих пределов.

Например:

- Количество юзеров на сайте
- Количество запросов в минуту



Стрессовое тестирование

Исследование поведения приложения при нештатных изменениях нагрузки, значительно превышающих расчетный уровень, или в ситуациях недоступности значительной части необходимых приложению ресурсов.

Стрессовое тестирование представляет собой крайнюю форму негативного тестирования.

Например:

- Отключить интернет
- Выключить из розетки ПК в момент обработки задачи
- Отменить действие когда оно уже начало обрабатываться

Как итог

Типы: ручное/автоматизированное, ящики (белый/черный/серый), статическое и динамическое

Уровни: модульное, интеграционное, системное, приемочное
Виды: функциональное, нефункциональное, связанное с изменениями

Самым высоким уровнем в иерархии подходов к тестированию будет понятие **типа**, которое может охватывать сразу несколько смежных техник тестирования. То есть, **одному типу тестирования может соответствовать несколько его видов**, при этом они могут проводится на **разных уровнях** работы с ПО.

Как это работает на практике

Протестируем карандаш:)
Придумайте пару проверок по нескольким методам

- Автоматизированное
- Ручное
- Модульное
- Интеграционное
- Системное
- Дымовое тестирование
- Негативное
- UI
- UX

ДЗ

1. https://ru.wikipedia.org/wiki/ISO_9126

   http://docs.cntd.ru/document/gost-r-iso-mek-9126-93

1. **Найти два дефекта, где угодно. Скриншот и указание видов/типов тестирования до 29.03.2021**

https://www.rstqb.org/ru/istqb-downloads.html