

Классификация тестирования

Виды тестирования

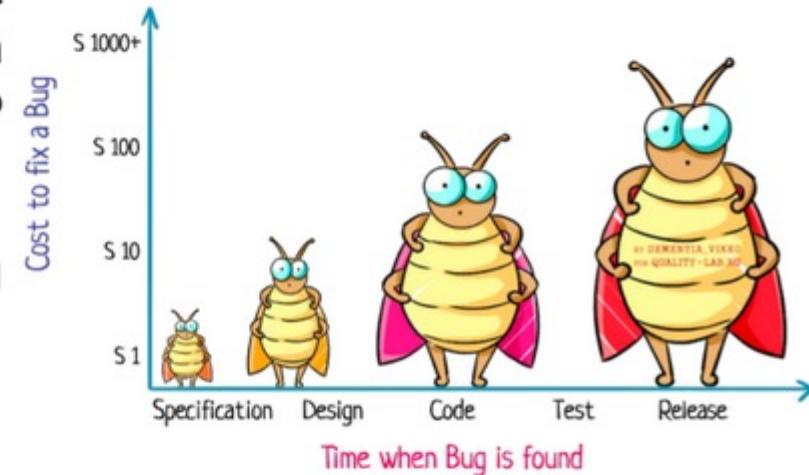


По запуску кода на исполнение

Статическое тестирование выполняется без запуска ПО.

Тестирование осуществляется путем анализа программного кода (code review) или скомпилированного кода.

Цель анализа – раннее выявление ошибок и потенциальных проблем в продукте.



По запуску кода на исполнение ↗

Статическое

Тестирование требований

Code review

Динамическое

Модульное тестирование

Интеграционное тестирование

Приёмочное тестирование

По запуску кода на исполнение

Примеры ошибок, которые можно выявить с помощью автоматического статического тестирования:

- Утечка ресурсов (утечка памяти, не освобождаемые файловые дескрипторы и т.д.);
- Возможность переполнения буфера (buffer overflows);
- Ситуации частичной (неполной) обработки ошибок.

Примеры статического тестирования:

- Тестирование требований;
- Статическое тестирование кода (Code review).



По запуску кода на исполнение

В отличие от статического, **динамическое тестирование** производится **путем запуска продукта** и проверки его функционала.

Она осуществляется с помощью ручного или автоматического выполнения заранее подготовленного набора тестов.

Примеры динамического тестирования:

- Модульное тестирование (unit testing);
- Интеграционное тестирование (integrated testing);
- Приемочное тестирование (acceptance testing).



По доступу к коду и архитектуре приложения



По доступу к коду и архитектуре приложения

Тестирование «черного ящика» – это стратегия или метод тестирования, который базируется только на тестировании по функциональной спецификации и требованиям, без учета внутренней структуры кода и без доступа к базе данных.

Мы знаем, какой должен быть результат при определенном наборе данных, которые подаются на вход.

Результат проверяем с интерфейса на уровне простого пользователя. На данный момент такая стратегия наиболее популярна в ИТ-компаниях.



По доступу к коду и архитектуре приложения

Проверка «серого ящика» – это метод тестирования программного продукта или приложения с частичным знанием его внутреннего устройства.

Для выполнения тестирования «серого ящика» **нет необходимости в доступе тестировщика к исходному коду.**

Тесты пишутся на основе знания алгоритма, архитектуры, внутренних состояний или других высокоуровневых описаний поведения программы.



По доступу к коду и архитектуре приложения

Наименьшая часть тестировщиков способна анализировать чужой код и заниматься написанием тестов, даже не запуская программу или приложение, а используя только программный код. **Это стратегия «белого ящика».**

Может использоваться в дополнение к черному и серому. Таких специалистов очень мало. Чаще всего это бывшие разработчики, ушедшие в тестирование, или тестировщики, занимающиеся автоматизацией и увлекающиеся программированием.

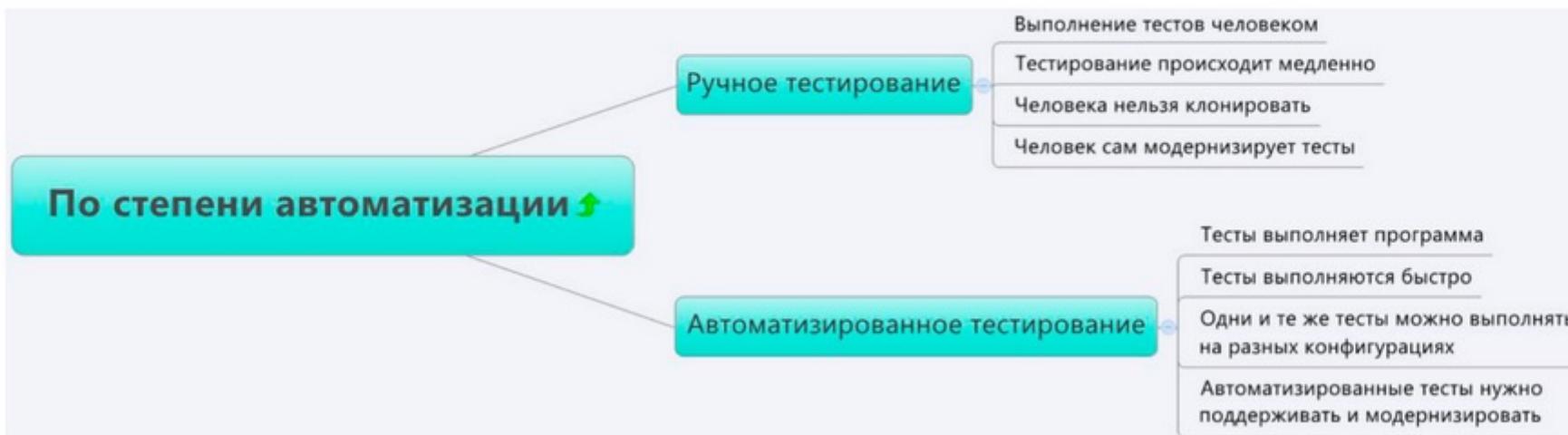
Юнит-тестирование, которое проводят, как правило, разработчики продукта, является **примером «white box» тестирования**.



По степени автоматизации

Ручное тестирование – это процесс поиска дефектов в работе программы, при котором тестировщик проверяет работоспособность всех компонентов, входя в роль пользователя.

Специалисты используют заранее заготовленные планы тестирования и тесты на основе требований к ПО.



По степени автоматизации

При автоматизированном тестировании используются программные средства для выполнения тестов и проверки результатов.

Применение автоматизированных тестов **позволяет сократить время** тестирования и **упростить сам процесс..**

Не стоит бросаться автоматизировать любой проект, но и исключать пользу автоматизированного



По степени автоматизации

И при ручном, и при автоматизированном тестировании нужно **составлять тесты и планы тестирования**.

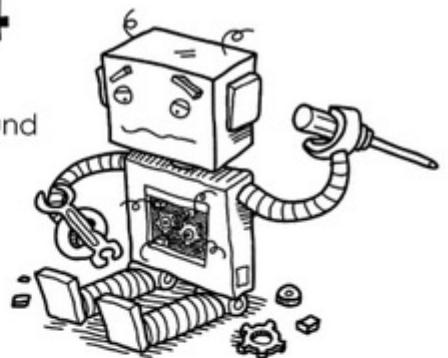
Тесты со временем устаревают, и их нужно актуализировать и поддерживать. **Тесты для ручного тестирования оформляются простым текстом, а автоматизированные – в виде программного кода.**

Если разработчики немного изменят интерфейс пользователя, то при ручном тестировании это изменение не будет критичным и тест, выполняемый человеком, пройдет успешно.

А автоматизированный тест сломается, т.к. он очень чувствителен к изменениям.

404

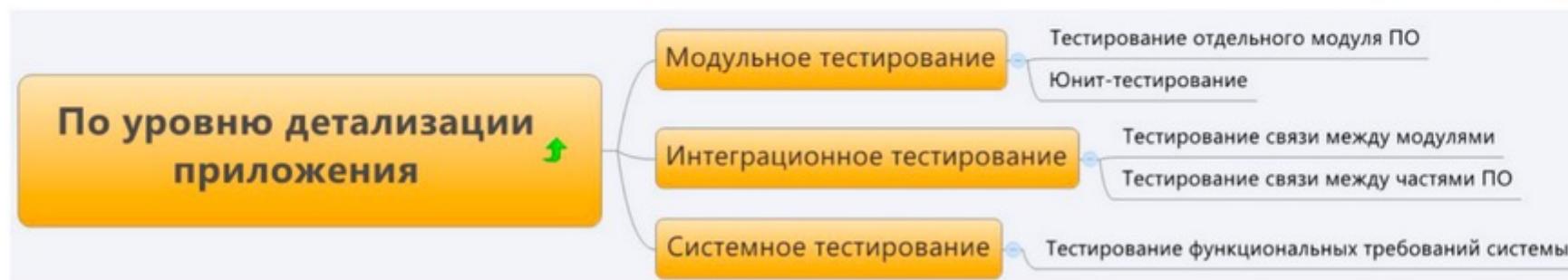
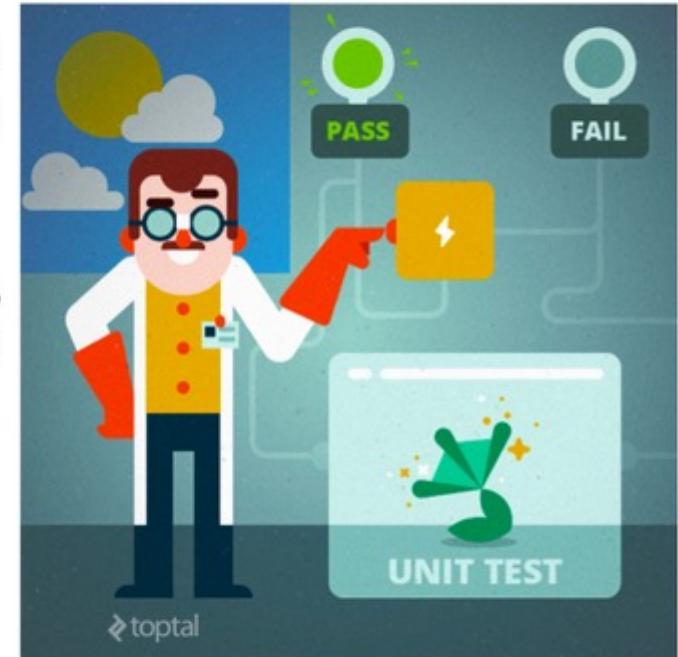
oops...
page not found



По уровню детализации

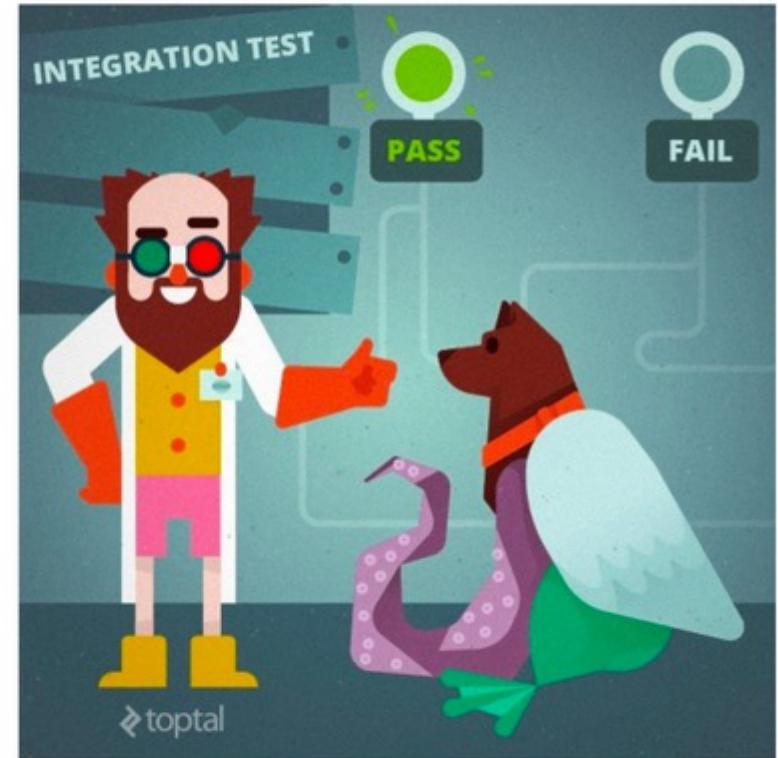
Модульное тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по отдельности.

Обычно модульное тестирование выполняют разработчики, создавая и программируя специальные модульные тесты (юнит-тесты) при разработке модуля программы.



По уровню детализации

Интеграционное тестирование предназначено для проверки связи между модулями ПО, а также взаимодействия с различными частями системы (ОС, оборудованием, взаимосвязями между различными системами).

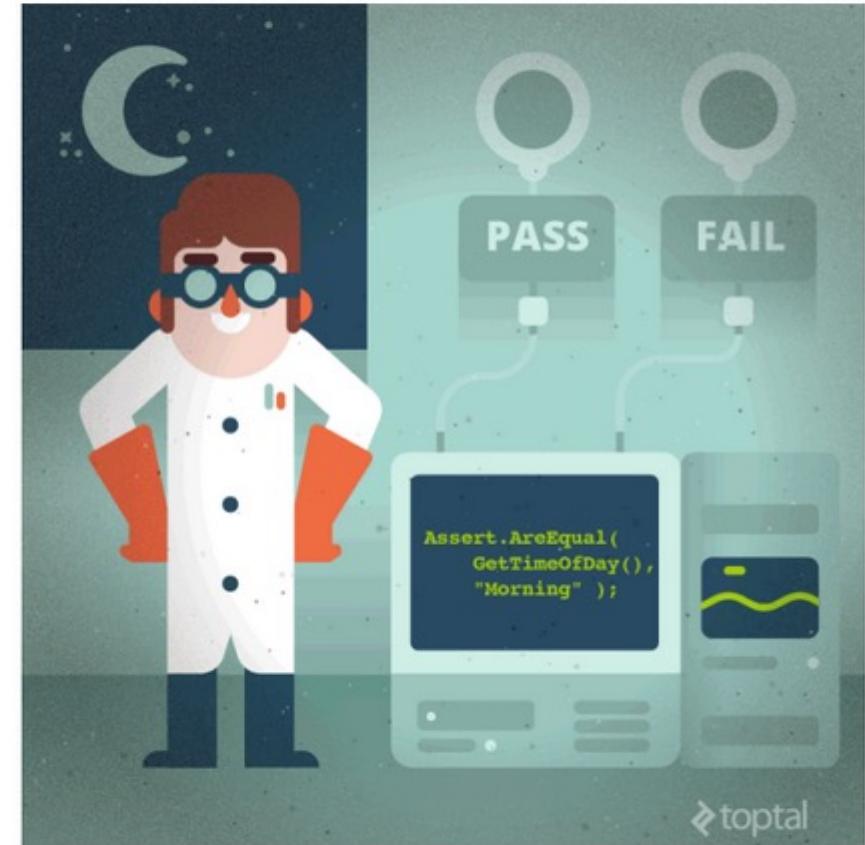


По уровню детализации

Основная задача системного тестирования – проверка как функциональных, так и нефункциональных требований в системе в целом.

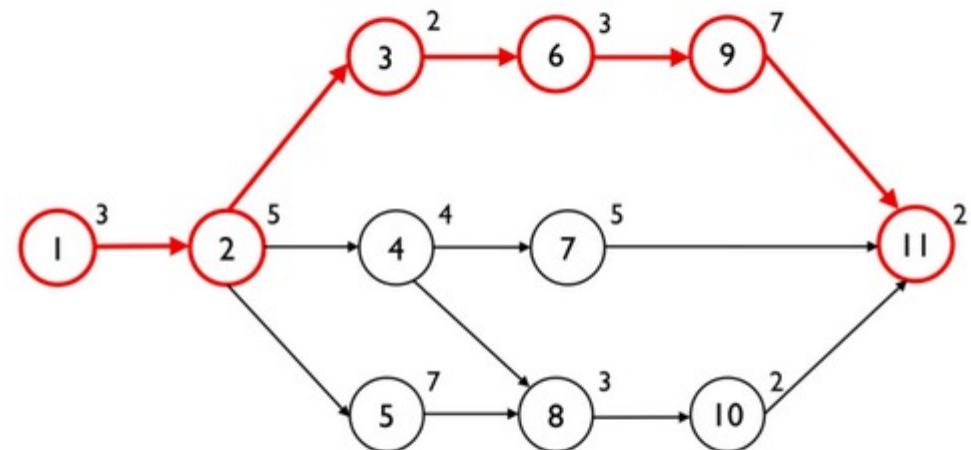
Выявляются такие дефекты, как неверное использование ресурсов системы, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д.

При проведении системного тестирования стараются создать специальные условия, приближенные к реальным.



По степени тестируемых функций

Тестирование критического пути – это проверка функциональности, используемой пользователями повседневно, наиболее часто используемых функций ПО.



По степени важности
тестируемых функций

Дымовое тестирование

Минимальный набор тестов на явные ошибки

Критерий продолжения более глубокого тестирования

Тестирование критического пути

Тестирование часто используемого функционала

Расширенное тестирование

Проверка всей функциональности по требованиям

По степени тестируемых функций

Дымовое (smoke) тестирование – это минимальный набор тестов на явные ошибки. Его успешное прохождение говорит о том, что ПО можно тестировать более глубоко и основательно.

Если ПО не прошло smoke-тестирование – то проверять дальше нет смысла.

Ранее термин применялся в тестировании радиоэлектроники.

Первое включение электронного устройства, пришедшего из производства, совершается на очень короткое время (меньше секунды). Затем инженер проверяет компоненты на предмет перегрева.

Если первое включение не выявило перегрева, то прибор включается снова на большее время. Проверка повторяется. И так далее несколько раз.



По степени тестируемых функций

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

При этом здесь также учитывается, какая функциональность является более важной, а какая — менее важной.

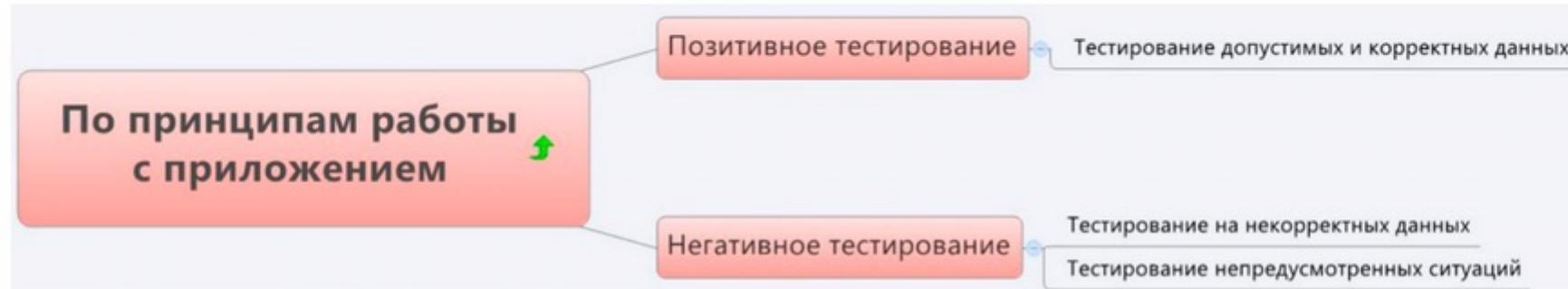
Но при наличии достаточного количества времени и иных ресурсов тест-кейсы этого уровня могут затронуть даже самые низкоприоритетные требования.



По принципам работы приложения

При позитивном тестировании проверяются функции ПО строго по требованиям и инструкциям, только с допустимыми действиями и корректными данными.

Например, позитивным тестом для проверки функции квадратного корня является ввод числа «4» и получение результата «+2» или «-2», или в поле «Возраст» ввести значение «18».



По принципам работы приложения

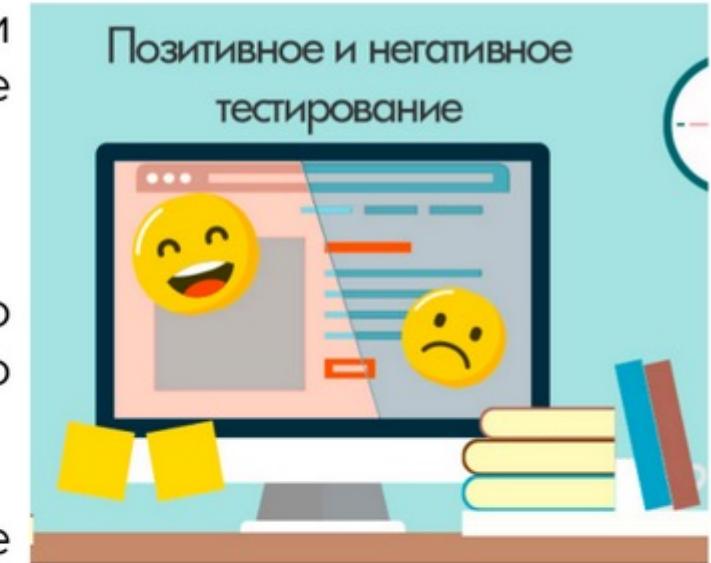
Негативное тестирование – это проверка поведения ПО при вводе некорректных данных.

Например, при вычислении квадратного корня ввести значение «FFA04» – число в шестнадцатеричной системе счисления. Как на него отреагирует калькулятор?

Или запросить корень из символов «!№@».

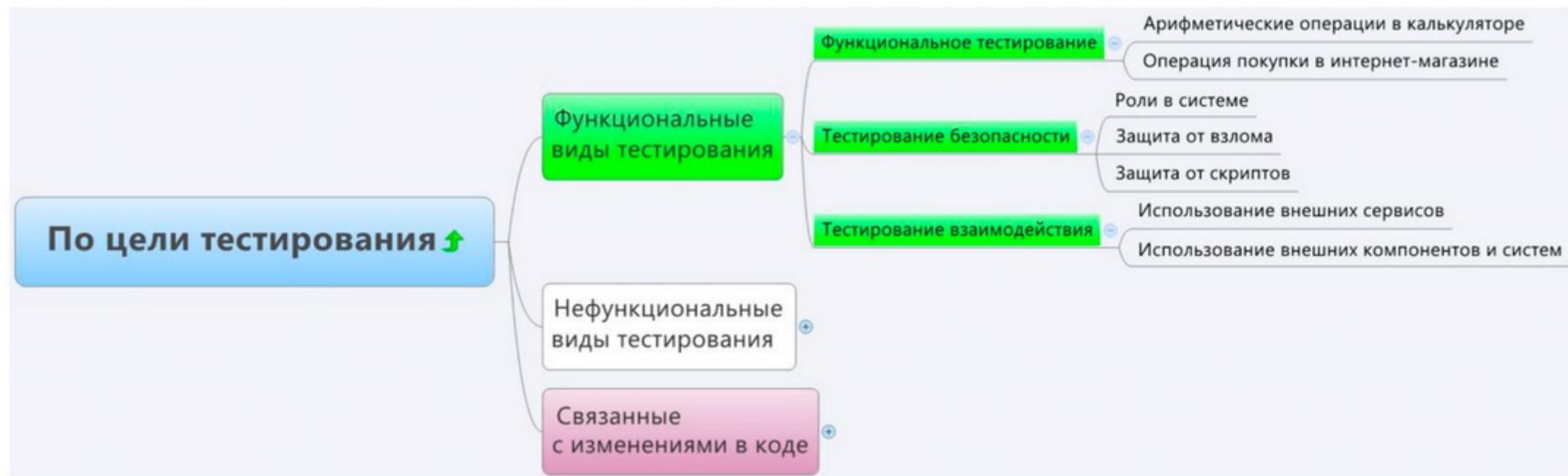
Ожидаемым результатом при таком teste является или число в шестнадцатеричном формате, или ошибка, гласящая, что введенное значение – не число.

Негативное тестирование проверяет, что приложение не выдает странных ошибок или не падает, когда к нему на вход подают что-то необычное.



По цели тестирования

Функциональное тестирование проверяет, правильно ли выполняет ПО функции, описанные в требованиях, а также как выполняется взаимодействие с другими системами.



По принципам работы приложения

К функциональным видам тестирования относят:

Функциональное тестирование основано на функциях, выполняемых системой.

Например, калькулятор совершает арифметические операции.

Тестирование безопасности – проверка ПО на безопасность системы, правильного ограничения возможностей пользователей, защита от хакерских атак.

Например, только администратор в системе может выдавать права и заводить новых пользователей.

Тестирование взаимодействия. Проверка взаимодействия ПО с различными внешними компонентами и системами.

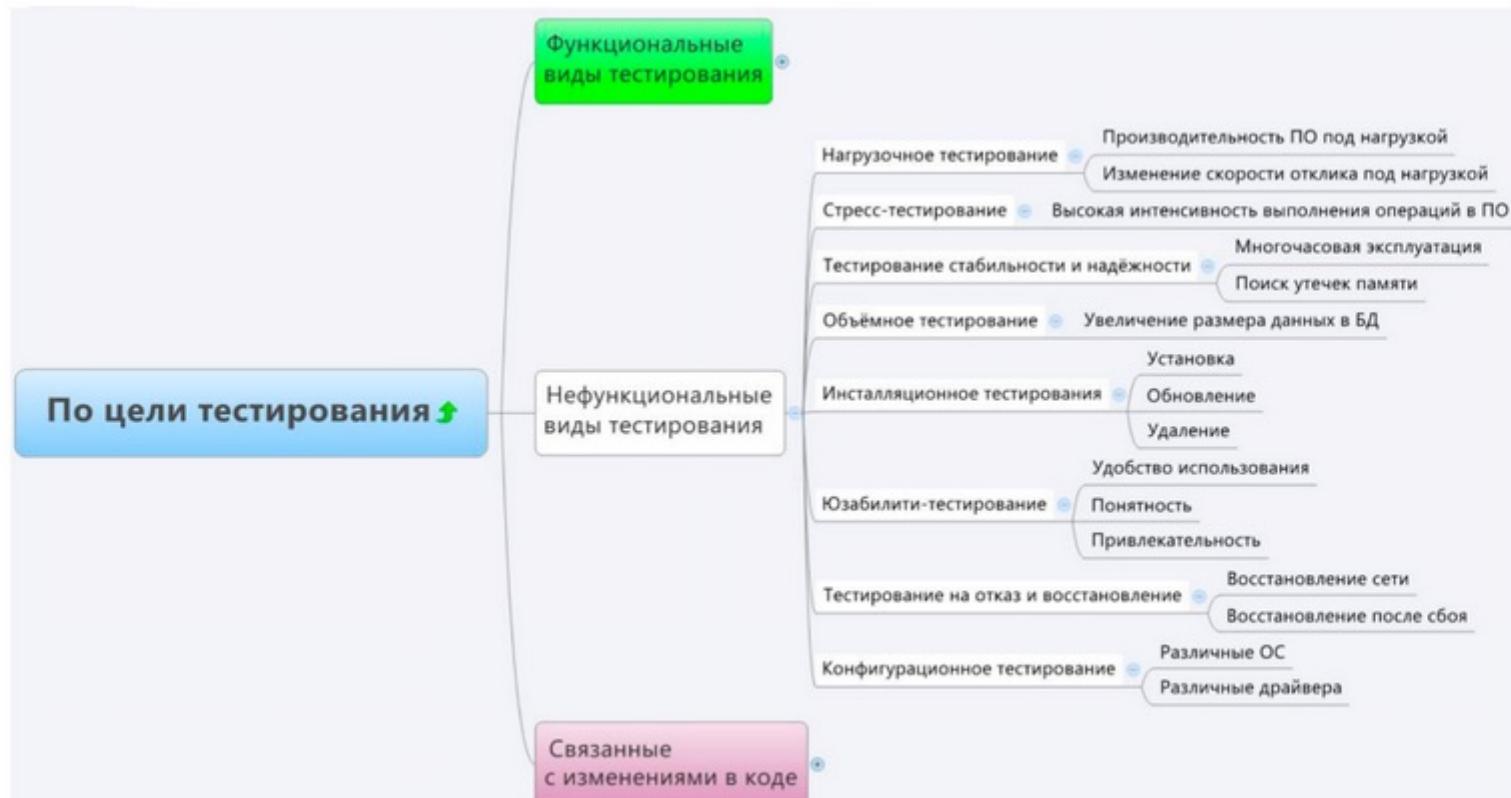
Например, можно ли использовать сервис авторизации «Вконтакте» для регистрации на сторонних сайтах.



По цели тестирования

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами.

В целом это тестирование того, как система работает.



По принципам работы приложения

Основные виды нефункционального тестирования:

Нагрузочное тестирование. Определение, как быстро работает ПО под заданной нагрузкой.

Например, при тестировании производительности анализируют изменение времени отклика ПО при определенном количестве одновременно работающих пользователей;

Стресс-тестирование. Тестирование, которое позволяет проверить, насколько приложение и система в целом работоспособны в условиях стресса. Оценивается и способность к регенерации, то есть возвращению к нормальному состоянию после стресса (повышения интенсивности выполнения операций до очень высоких значений);



По принципам работы приложения

Основные виды нефункционального тестирования:

Тестирование стабильности и надежности.

Проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может играть второстепенную роль. При этом на первый план выходит отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты, влияющие именно на стабильность работы;

Объемное тестирование. Оценка производительности при увеличении объемов данных в базе данных приложения. При этом измеряется время выполнения выбранных операций и количество пользователей, одновременно работающих с приложением;



По принципам работы приложения

Основные виды нефункционального тестирования:

Тестирование на отказ и восстановление.

Тестирование, которое проверяет ПО с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказом сети).

Цель – проверка систем восстановления (или дублирующих основной функционал систем), которые при сбоях обеспечат сохранность и целостность данных тестируемого продукта.

Тестирование на отказ и восстановление очень важно для систем, работающих по принципу «24x7»;



По принципам работы приложения

Основные виды нефункционального тестирования:

Инсталляционное тестирование.

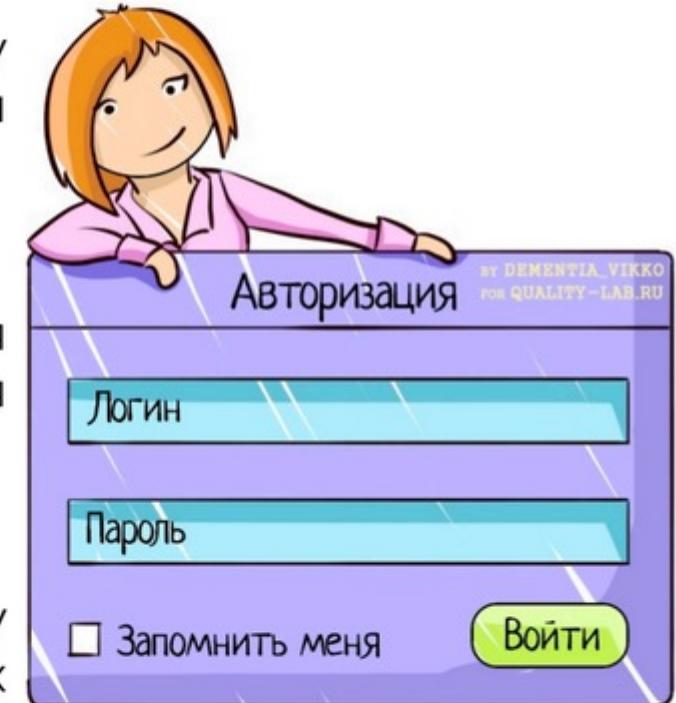
Тестирование установки ПО, направленное на проверку успешной инсталляции и настройки, а также обновления и удаления ПО;

Юзабилити-тестирование.

Оценка ПО с точки зрения удобства использования, изучения и освоения, понятности и привлекательности для пользователей в контексте заданных условий;

Конфигурационное тестирование.

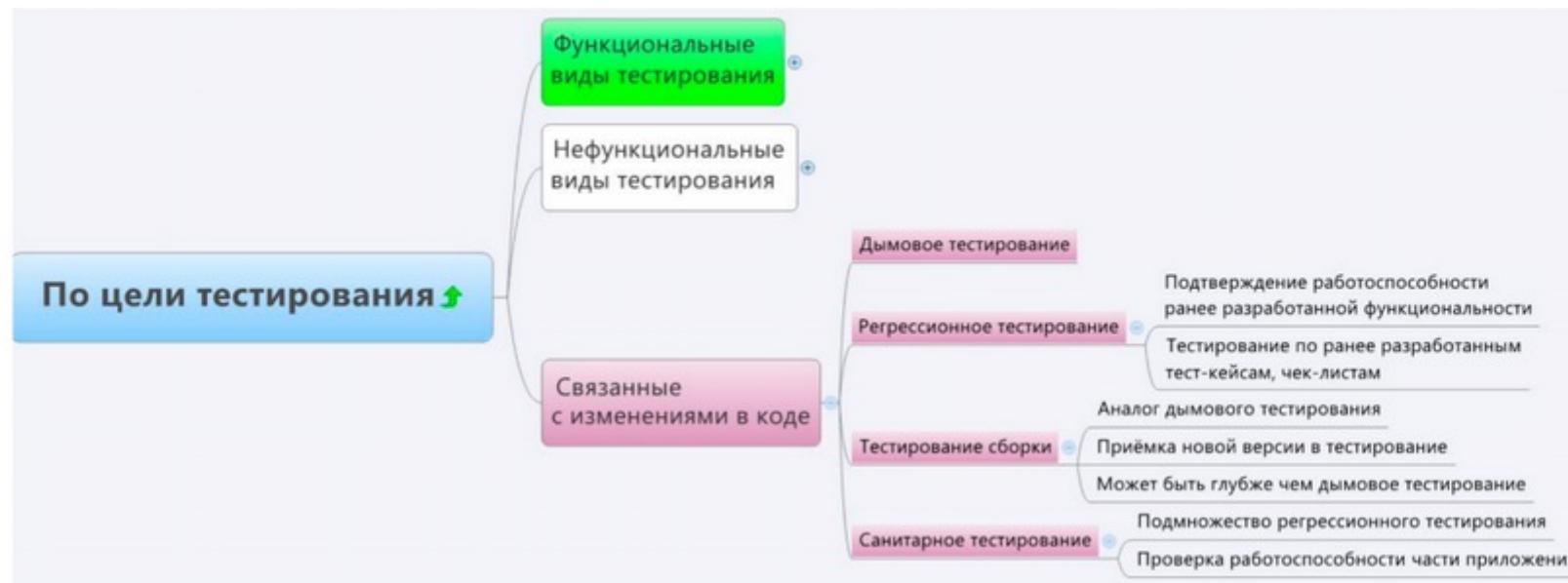
Специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т. д.).



По цели тестирования

После внесения изменений (исправления бага/дефекта) программное обеспечение должно быть протестировано заново для подтверждения, что проблема действительно решена.

Ниже перечислены виды тестирования, которые необходимо проводить после установки программного обеспечения:



По принципам работы приложения

Дымовое тестирование

Санитарное тестирование. Узконаправленное тестирование, достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям.

Используется для определения работоспособности определенной части приложения после изменений, произведенных в ней или окружающей среде.



По принципам работы приложения

Регрессионное тестирование.

Это 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).

Проводится, чтобы подтвердить, что прежняя функциональность работает. Это дает гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность.



По принципам работы приложения

Тестирование сборки.

Определяет соответствие выпущенной версии начальным критериями качества.

По своим целям является аналогом дымового тестирования, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию.

Оно может проникать глубже – в зависимости от требований к качеству выпущенной версии;

