

Типы тестирования



Тип тестирования

(testing type) - группа активностей тестирования, направленных на тестирование характеристик компонента или системы с определенной целью.





Цели

1. Оценка функциональных характеристик качества системы, таких как полнота, корректность, целесообразность
2. Оценка нефункциональных характеристик качества, таких как надежность, продуктивность работы, безопасность, совместимость и удобство использования
3. Оценка правильности, полноты структуры или архитектуры компонента или системы, их соответствие спецификации
4. Оценка влияния изменений, например, подтверждение того, что дефекты были исправлены (подтверждающее тестирование) и поиск непреднамеренных изменений в поведении, вызванных изменениями в программном обеспечении или окружении (регрессионное тестирование)

Цели*

1. Оценка функциональных характеристик качества системы, таких как полнота, корректность, целесообразность
2. Оценка нефункциональных характеристик качества, таких как надежность, продуктивность работы, безопасность, совместимость и удобство использования
3. Оценка правильности, полноты структуры или архитектуры компонента или системы, их соответствие спецификации
4. Оценка влияния изменений, например, подтверждение того, что дефекты были исправлены (подтверждающее тестирование) и поиск непреднамеренных изменений в поведении, вызванных изменениями в программном обеспечении или окружении (регрессионное тестирование)

**были убраны из версии 2023*



Функциональное и нефункциональное тестирование

Функциональное тестирование оценивает функции, которые компонент или система должны выполнять. Функции дают ответ на вопрос «что делает система».

Основная цель функционального тестирования — это проверка функциональной полноты, функциональной правильности и функционального соответствия.

Нефункциональное тестирование оценивает признаки компонента или системы, отличные от функциональных характеристик. Нефункциональное тестирование – это проверка того, «насколько хорошо работает система».

Основная цель нефункционального тестирования — это проверка нефункциональных характеристик качества программного обеспечения.

Различия функционального и нефункционального Т.



Функциональное Т.

Что система делает?

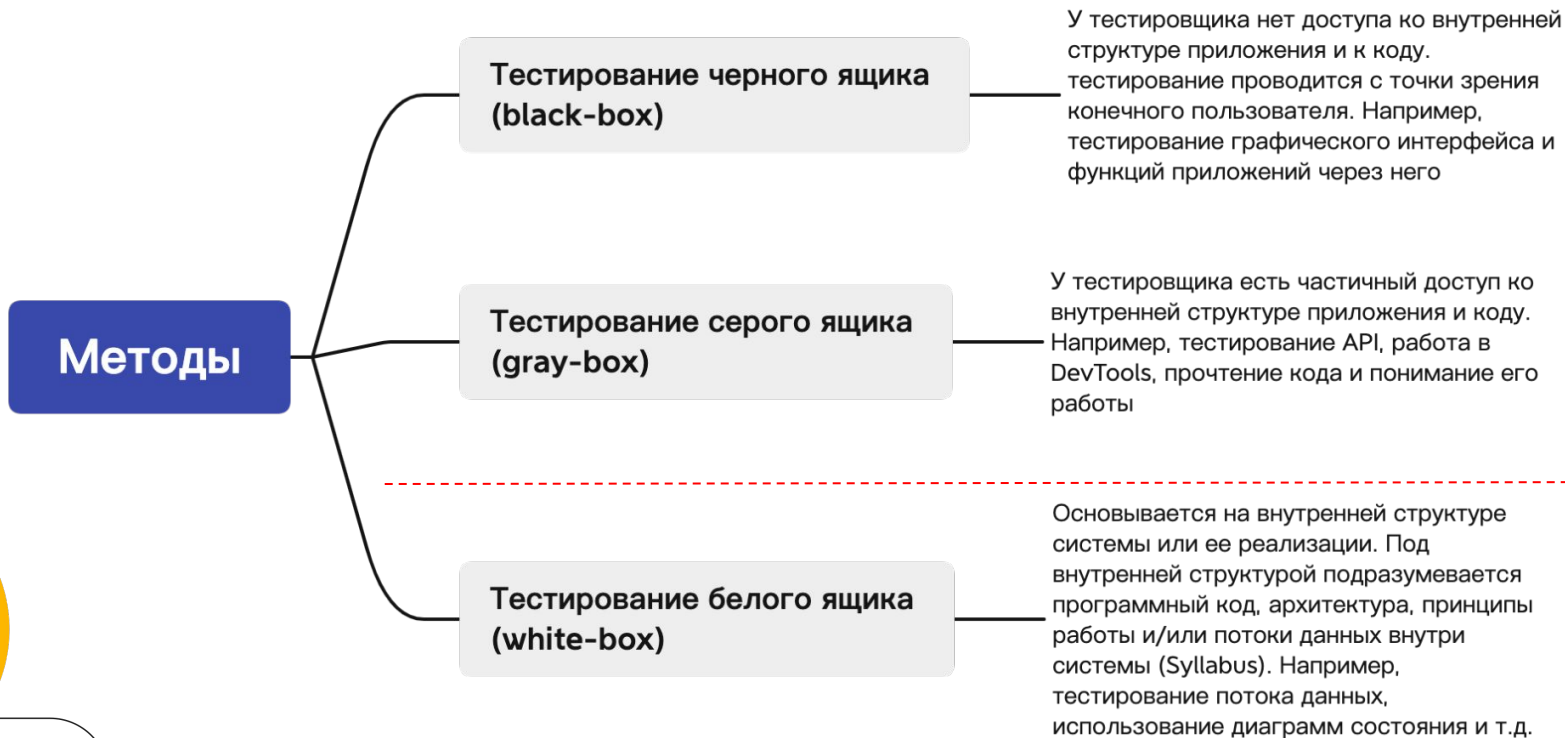


Нефункциональное Т.

Как система это делает?



Методы тестирования





Тестирование, связанное с изменениями

Подтверждающее тестирование (confirmation testing, re-test) проверяет, что исходный дефект был успешно исправлен. В зависимости от степени риска тестирование может проводиться несколькими способами, среди которых:

- выполнение всех тестовых сценариев, завершившихся с ошибкой из-за дефекта
- добавление новых тестовых сценариев для покрытия исправляющих дефект изменений

Регрессионное тестирование (regression testing)- подтверждает, что внесенные в систему изменения не имели негативных последствий, включая исправление дефекта, уже прошедшего подтверждающее тестирование.

Регрессионное тестирование может не ограничиваться только самим объектом тестирования, но и распространяться на окружение.





Тестирование, связанное с изменениями

Регрессия багов (Bug regression) - попытка доказать, что исправленная ошибка на самом деле не исправлена;

Регрессия старых багов (Old bugs regression) - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться;

Регрессия побочного эффекта (Side effect regression) - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения;



Как формировать регрессионные наборы?

1. Высокоприоритетные задачи
2. Тестирование функций и модулей, которые чаще всего используются пользователями
3. Тестирование функций и модулей, в которых часто и регулярно обнаруживаются дефекты
4. Тестирование функций и модулей, которые связаны с изменением

Для удобства можно использовать **анализ влияний** (анализ изменений, impact analysis) - определение всех объектов, подверженных влиянию изменения, а также оценка ресурсов, необходимых для реализации изменения (ISTQB Glossary)



Классификация по запуску кода на исполнение

Статическое тестирование (static testing) - тестирование без запуска на исполнение тестируемого объекта (без запуска кода).

Примеры: тестирование документации, прототипов, кода (в рамках code review), тестовых данных. Есть специальные техники для статического анализа (см. ISTQB CTFL Syllabus)

Динамическое тестирование (dynamic testing) - тестирование, проводимое во время выполнения тестируемого элемента (с запуском кода).

Пример: проверка реального поведения приложения при запуске кода на разных уровнях тестирования



Классификация по степени автоматизации

Ручное тестирование (manual testing) - тестирование, в котором тест-кейсы выполняются человеком вручную без использования средств автоматизации.

Примеры: практически все, что мы освоим с вами в курсе :)

Автоматизированное тестирование (automated testing, test automation) - набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования.

Пример: в данном случае автоматизируют проверки, используя определенный язык программирования, которые до этого написал человек

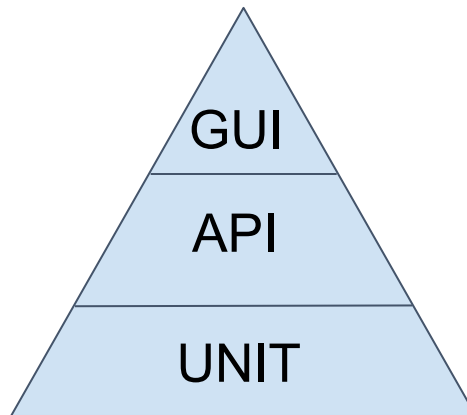
Важно помнить, что автоматизировать все нельзя, поэтому и существует ручное тестирование. В том числе есть специалисты, которые совмещают в себе две функции (General QA, Fullstack QA, Hybrid QA).

Что чаще всего автоматизируют?

Все рутинные, важные и времязатратные операции

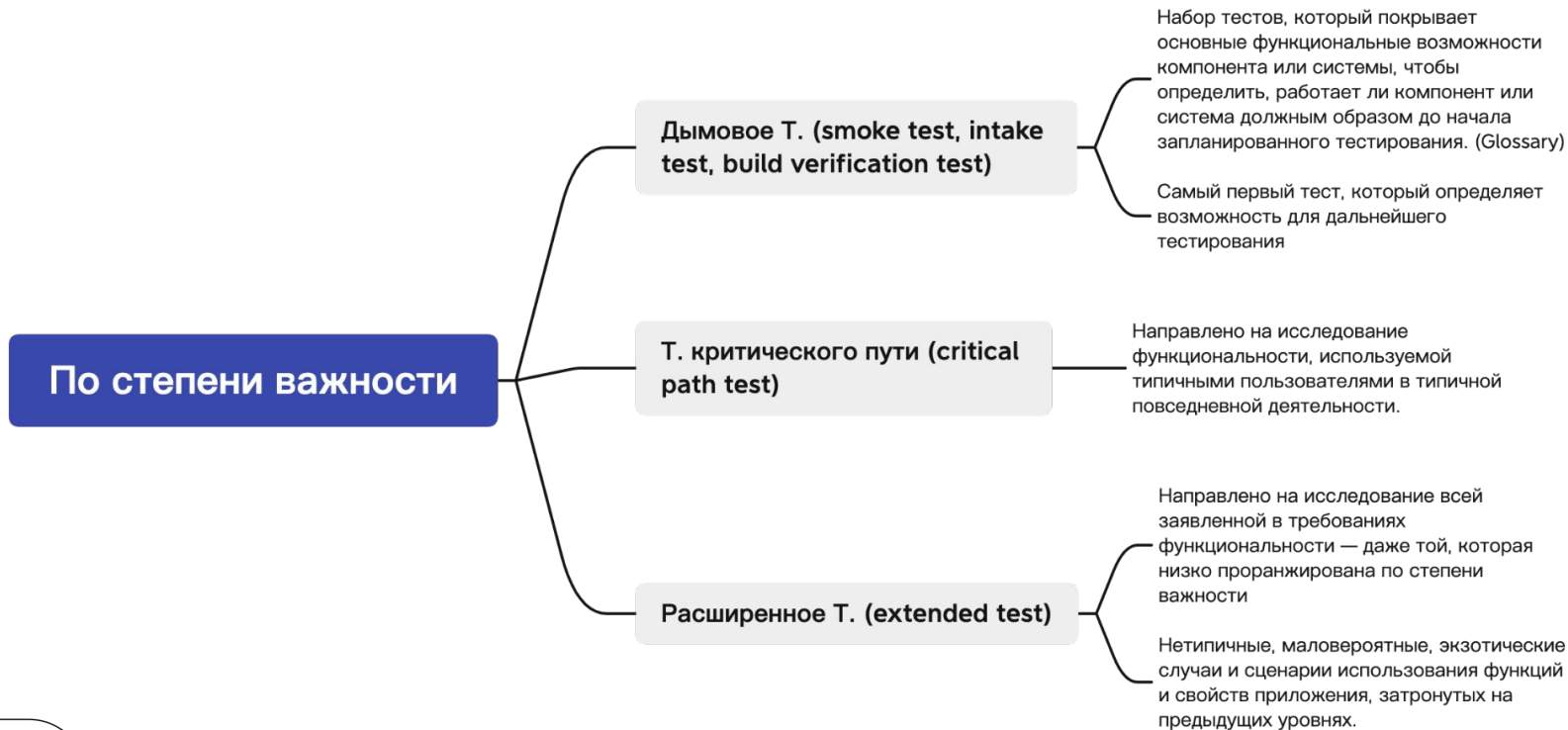
1. Дымовое тестирование
2. Регрессионное тестирование
3. Сквозное тестирование (end-to-end testing, E2E) - это вид тестирования, используемый для проверки программного обеспечения от начала до конца

*Также всегда нужно помнить
про пирамиду автоматизации
тестирования, правильную
пропорцию и этапы
разработки автотестов*





По степени важности тестируемых функций



Как это может выглядеть на проекте?

Smoke

Фаза: получение нового билда

Тестирование основной функциональности и flow (потока) пользователя, например, от регистрации и до оформление товара

Объем Т. может зависеть от набора тестируемых функций. Могут запускать не все дымовые тесты

Extended

Фаза: тестирование новой функциональности после проведения СР

Чаще всего под этим видом тестирования имеют в виду тестирование в нестандартных условиях, например, тестирование входа в систему через VPN или используя китайскую раскладку с электронной клавиатуры

Тестирование высокоприоритетных компонентов и функций, описанных в задаче, например, полное тестирование функциональности входа в систему

Фаза: тестирование новой функциональности после проведения Smoke

СР



Smoke vs. Sanity

Согласно ISTQB Glossary это одно и то же, однако есть и другое мнение авторов. В сети за Sanity закрепилось название санитарное. Однако в ISTQB это “Тест работоспособности”

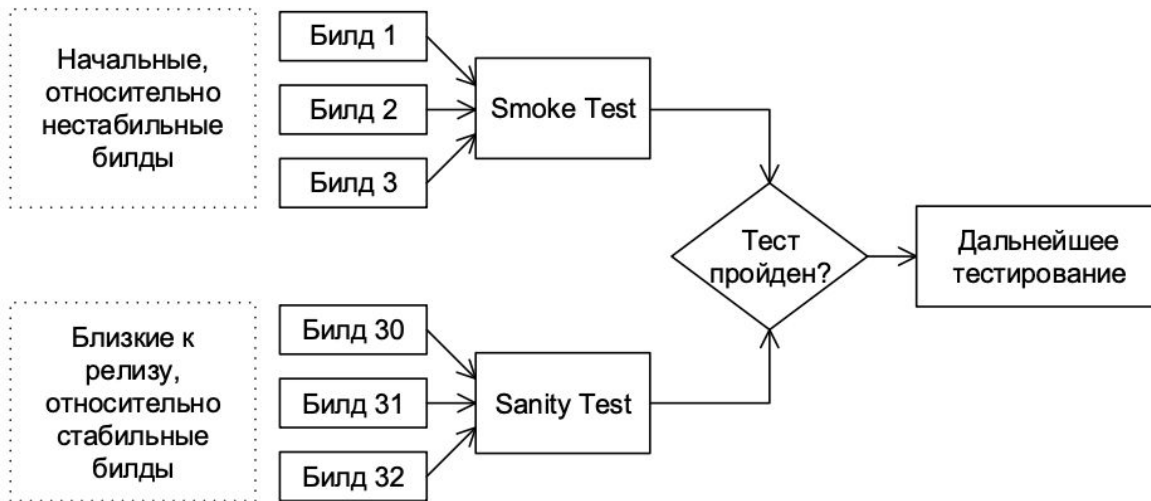


Рисунок 2.3.f — Трактовка разницы между smoke test и sanity test

Святослав Куликов “Тестирование ПО. Базовый курс.”



Классификация по принципам работы с приложением

Позитивное тестирование (positive testing) - 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 тестирование (negative testing, invalid testing) — 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).

Деструктивное тестирование (destructive testing) - одна из форм негативного тестирования с целью нарушить работоспособность приложения и обнаружить точку отказа.

Примеры

Позитивное тестирование: регистрация пользователя со всеми заполненными полями и верными данными

Негативное тестирование: регистрация пользователя с пустыми полями и/или неверными данными

Деструктивное тестирование: регистрация пользователя с отправкой очень длинного имени и/или загрузкой очень большого файла

Нужно понимать, что идеи для негативного тестирования могут быть в требованиях как в явном, так и не явном виде.

Наличие требования на валидацию данных после ввода - один из самых очевидных кандидатов для такого вида тестирования.





Классификация по природе приложения

- Тестирование веб-приложений (web-applications testing)
- Тестирование мобильных приложений (mobile-applications testing)
- Тестирование настольных приложений (desktop-applications testing)
- Тестирование игр (games-testing)
- Embedded-testing (встроенное тестирование?) - тестируется не только программное обеспечение, но и его работа на определенном аппаратном обеспечении
- Тестирование по бизнес-доменам. Например, банки, медицина, образование.



Классификация по степени формализации

Тестирование на основе тест-кейсов (scripted testing, test case based testing) — формализованный подход, в котором тестирование производится на основе заранее подготовленных тест-кейсов, наборов тест-кейсов и иной документации.

Исследовательское тестирование (exploratory testing) — частично формализованный 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 (интуитивное) тестирование (ad hoc testing) — полностью неформализованный подход, в котором не предполагается использования ни тест-кейсов, ни чек-листов, ни сценариев — тестировщик полностью опирается на свой профессионализм и интуицию (experience-based testing) для спонтанного выполнения с приложением действий, которые, как он считает, могут обнаружить ошибку.

Классификация по целям и задачам

Инсталляционное тестирование (installation testing, installability testing) — тестирование, направленное на выявление дефектов, влияющих на протекание стадии инсталляции (установки) приложения.

Включает в себя следующие процессы:

1. Установка ПО
2. Удаление ПО
3. Обновление ПО
4. Откат на предыдущую версию
5. Повторный запуск установки после возникновения ошибки или исправления уже возникших проблем
6. Автоматическая установка
7. Установка отдельного компонента из общего пакета программ

Классификация по целям и задачам

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

Что нужно тестировать:

1. Общая доступность
2. Скорость, производительность
3. Удобство навигации и интерфейс
4. Плавность



Классификация по целям и задачам

Тестирование доступности (accessibility testing, АИТ) — тестирование, направленное на исследование пригодности продукта к использованию людьми с ограниченными возможностями (слабым зрением и т.д.).

Что можно тестировать:

1. Использование вспомогательных технологий в ПО (распознавание речи, экранная клавиатура и лупа, скринридеры)
2. Возможность использовать приложение одной рукой
3. Настройки специальной цветопередачи
4. Наличие понятных инструкций и руководства пользователя

Доступность сайта можно тестировать специальными инструментами, например, WAVE, TAV, Accessibility Valet, Accessibility Developer Tools



РАЗМЕР ШРИФТА 17 PX

A - A +

ЦВЕТА САЙТА

Ц Ц

ИЗОБРАЖЕНИЯ

ДОПОЛНИТЕЛЬНО

Настройки

Налоговые инспекции

RU

13:57 100 %

Основные Универс. доступ

ЗРЕНИЕ

VoiceOver

Выкл. >

Увеличение

Выкл. >

Инверсия цвета

Проговаривание

Выкл. >

Автопроизношение

Автоматически проговаривать автоматические исправления и написание с заглавной буквы.

Увеличенный текст

Выкл. >

Жирный шрифт

Формы кнопок

Спец. возможности

ЭКРАН

Размер шрифта

По умолчанию

Масштаб изображения на экране

По умолчанию

Увеличение

Отключено

Коррекция цвета

Отключено

Инверсия цветов

Отключено

Крупный указатель мыши

Удалить анимации



Классификация по целям и задачам

Тестирование безопасности (security testing) — тестирование, направленное на проверку способности приложения противостоять злонамеренным попыткам получения доступа к данным или функциям, права на доступ к которым у злоумышленника нет.

От чего нужно защищать ПО:

1. SQL-инъекции
2. XSS-инъекции
3. Перехват трафика
4. Брутфорсинг (полный перебор данных для получения доступа)

Есть так называемый [OWASP TOP 10](#), в котором собраны самые популярные и опасные уязвимости, а также рекомендации по борьбе с ними и тестированию



Классификация по целям и задачам

Тестирование интернационализации (internationalization testing, i18n testing, globalization testing, localizability testing) — тестирование, направленное на проверку готовности продукта к работе с использованием различных языков и с учётом различных национальных и культурных особенностей.

Тестирование локализации (localization testing, l10n) — тестирование, направленное на проверку корректности и качества адаптации продукта к использованию на том или ином языке с учётом национальных и культурных особенностей.

Различия i18n и l10n

i18n

Т. интернационализации

Готовность к адаптации в
новых локалях

Примеры:

Направление текста
Возможность использования
разных кодировок
Конечный вид интерфейса

l10n

Т. локализации

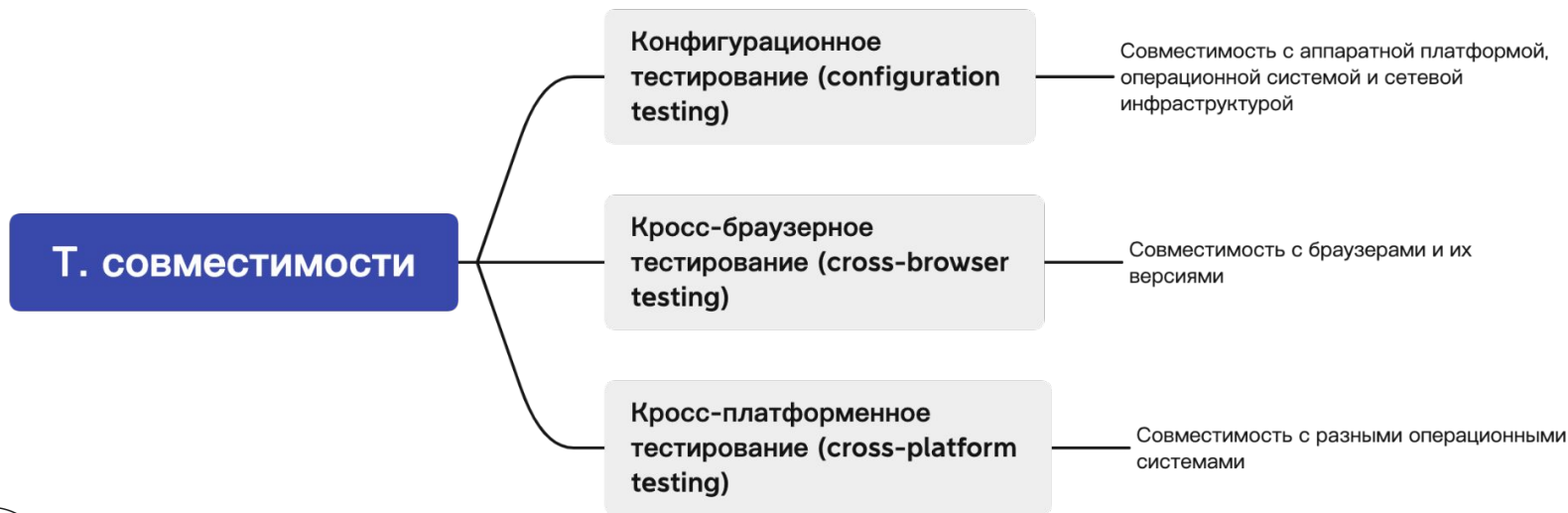
Сама адаптация к локалям

Примеры:

Перевод
Валюта
Цвета
Нормативы
Единицы измерения
Формат даты и времени

Классификация по целям и задачам

Тестирование совместимости (compatibility testing, interoperability testing) — тестирование, направленное на проверку способности приложения работать в указанном окружении.





Классификация по целям и задачам

Тестирование надёжности (reliability testing) — тестирование способности приложения выполнять свои функции в заданных условиях на протяжении заданного времени или заданного количества операций.

Тестирование восстанавливаемости (recoverability testing) — тестирование способности приложения восстанавливать свои функции и заданный уровень производительности, а также восстанавливать данные в случае возникновения критической ситуации, приводящей к временной (частичной) утрате работоспособности приложения.

Тестирование отказоустойчивости (failover testing) — тестирование, заключающееся в эмуляции или реальном создании критических ситуаций с целью проверки способности приложения задействовать соответствующие механизмы, предотвращающие нарушение работоспособности, производительности и повреждения данных.

Классификация по целям и задачам

Тестирование производительности (performance testing) — исследование показателей скорости реакции приложения на внешние воздействия при различной по характеру и интенсивности нагрузке.

Подвиды:

1. Нагрузочное тестирование (load testing, capacity testing)
2. Тестирование масштабируемости (scalability testing)
3. Объёмное тестирование (volume testing)
4. Стрессовое тестирование (stress testing)
5. Конкурентное тестирование (concurrency testing)



Тестирование производительности

Нагрузочное тестирование — исследование способности приложения сохранять заданные показатели качества при нагрузке в допустимых пределах и некотором превышении этих пределов (определение «запаса прочности»).

Тестирование масштабируемости — исследование способности приложения увеличивать показатели производительности в соответствии с увеличением количества доступных приложению ресурсов.

Объёмное тестирование — исследование производительности приложения при обработке различных (как правило, больших) объёмов данных.



Тестирование производительности

Стрессовое тестирование — исследование поведения приложения при нештатных изменениях нагрузки, значительно превышающих расчётный уровень, или в ситуациях недоступности значительной части необходимых приложению ресурсов.

Конкурентное тестирование — исследование поведения приложения в ситуации, когда ему приходится обрабатывать большое количество одновременно поступающих запросов, что вызывает конкуренцию между запросами за ресурсы (базу данных, память, канал передачи данных, дисковую подсистему и т.д.).



Тестирование производительности





Нефункциональное тестирование

В практике тестирования принято разделение тестирования на тестирование функционального показателя качества, называемое «функциональным тестированием», и тестирование других показателей качества, называемое «нефункциональным тестированием». Тип тестирования, используемого для определения показателя качества, отличного от функциональной пригодности, обычно называют нефункциональным типом тестирования и к нему можно отнести такие типы тестирования, как нагрузочное тестирование, стрессовое тестирование, тестирование на возможность проникновения, тестирование удобства использования и т. д.

Нефункциональное тестирование

ГОСТ Р ИСО/МЭК 25010—2015



Рисунок 4 — Модель качества продукта



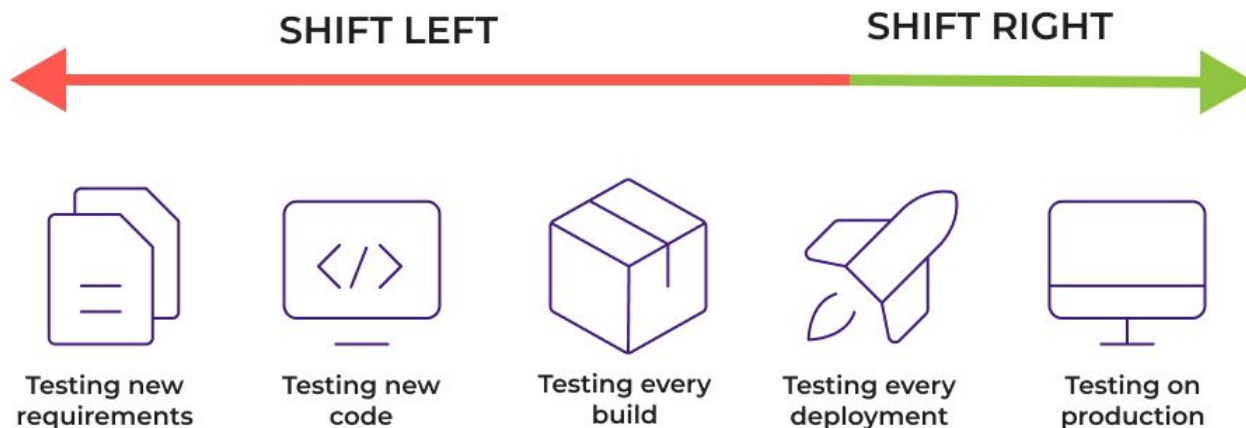
Нефункциональное тестирование

Существует трактовка, по которой к нефункциональному тестированию можно отнести тестирование безопасности и нагрузочное тестирование, но только в тех случаях, когда они представляют основную функциональность приложения, а не просто являются дополнительными характеристиками.

Например, защита денежных транзакций, функции антивируса, возможность одновременной работы над задачей несколькими пользователями.

Использовать с осторожностью и только, если спросят про такие кейсы. В стандартах это не упоминается.

SHIFT LEFT и SHIFT RIGHT TESTING



*Сдвиг влево также упоминается ISTQB Syllabus версии 2023

<https://www.encora.com/insights/shift-left-and-shift-right-in-software-testing>



Реальная жизнь и советы

1. Не нужно знать абсолютно все типы тестирования. Чаще всего на интервью вас спросят: какие виды тестирования вы знаете? Советую начинать с нефункционального и функционального тестирования, а дальше переходить к более мелким видам, постепенно раскрывая матрешку
2. При выходе на проект начинайте всегда с исследовательского тестирования для того, чтобы определить будущую стратегию тестирования
3. На практике используется ограниченное количество видов тестирования. Особый упор при запоминании сделайте на тестирование, связанное с изменениями, Smoke, тестирование по важности функций, методы тестирования
4. Большая часть вашей работы будет крутиться вокруг регрессионного и функционального тестирования