

Типы тестирования. Конспект

Примечание от автора курса

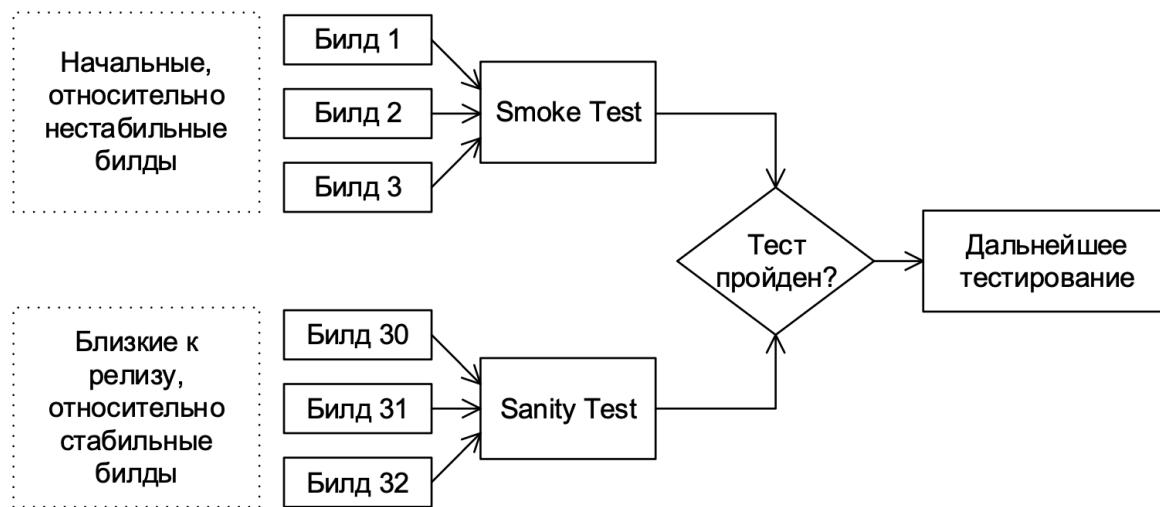
Существует большое количество классификаций тестирования от разных авторов и стандартов. В данном курсе мы сконцентрируемся на типах и методах тестирования, определенных в ISTQB CTFL и пособии Святослава Куликова.

Мы не сможем рассмотреть абсолютно все типы, поэтому рассмотрим только самые основные.

Советую подойти к изучению этой темы последовательно и не стараться разобраться во всех типах тестирования за один раз.

Классификация по (убыванию) степени важности тестируемых функций

- **Дымовое тестирование (smoke test)** направлено на проверку самой главной, самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной саму идею использования приложения (или иного объекта, подвергаемого дымовому тестированию).
- Очень часто можно услышать вопрос о том, чем «smoke test» отличается от «sanity test». В глоссарии ISTQB сказано просто: «sanity test: See smoke test». Но некоторые авторы утверждают, что разница есть и может быть выражена следующей схемой



- **Тестирование критического пути (critical path test)** направлено на исследование функциональности, используемой типичными пользователями в типичной повседневной деятельности.
- **Расширенное тестирование (extended test)** направлено на исследование всей заявленной в требованиях функциональности – даже той, которая низко проранжирована по степени важности. Ещё одним направлением исследования в рамках данного тестирования являются нетипичные, маловероятные, экзотические случаи и сценарии использования функций и свойств приложения, затронутых на предыдущих уровнях.

Источник: [Святослав Куликов "Тестирование ПО. Базовый курс."](#)

Пояснения автора курса

- **Smoke:** пользователь авторизовался, смог найти и добавить товар в корзину, совершил оплату и оформил доставку - в качестве примера для онлайн-магазина.

Проверка основной функциональности обычно не меняется и проводится на каждой новой промежуточной версии продукта, или билде. Иногда этот процесс называют **сертификацией билда**.

Если Smoke не проходит, то остальное тестирование останавливается и ждет доработки кода.

- **Critical path:** тестируются новые или измененные функции, которые попали в промежуточную версию продукта с точки зрения использования конечного пользователя.
- **Extended:** все остальное.

Что касается **Sanity** тестирования, то с одной точки зрения это синоним Smoke.

Есть и другая точка зрения, при которой Smoke является тестированием основной функциональности на нестабильных билдах, а Sanity проверяет то же самое, но уже на стабильных билдах, близких к релизу.

Ретест и регрессионное тестирование

Подтверждающее (re-testing, ретест, повторное тестирование) - тип тестирования, связанного с изменениями, которое выполняется после исправления дефекта для подтверждения того, что отказ, вызванный этим дефектом, не воспроизводится. [ISTQB Glossary]

Регрессионное тестирование (regression testing) - тип тестирования, связанного с изменениями, чтобы найти привнесенные или ранее не обнаруженные дефекты в не менявшихся областях программного обеспечения. [ISTQB Glossary]

Примечание автора курса

Ретест подтверждает факт того, что дефект был исправлен. Он не требует оценку влияния изменений, хотя хороший тестировщик всегда потратит несколько минут на общую оценку работы приложения после такого фикса.

Регрессионное тестирование проводится после любого изменения (новая функциональность, исправление дефекта, удаление функциональности), чтобы убедиться в том, что приложение работает стабильно. Оно не включает в себя ретест, а проводится уже после него.

Особенности регрессионного тестирования из практики

Регрессионное тестирование является одной из самых главных активностей тестировщика и проводится перед каждым релизом (выпуском продукта для конечного пользователя).

Если релиз запланирован на каждую неделю, то регрессия проводится с такой же периодичностью, поэтому тесты для регрессии, как и дымные тесты очень часто автоматизируют для экономии времени.

Обычно для проведения регрессии выделяют несколько дней, код проекта "замораживают"(code freeze) и в него нельзя вносить изменения, кроме исправлений критичных дефектов.

Важно понимать: какие тесты в конечном итоге попадут в регрессионный набор?

Регрессионные тесты выбираются из уже существующих тестовых наборов на основании следующих принципов:

- Тесты, проверяющие часть приложения, в которые вносились изменения

Например, если дефект или новая функция локализованы в модули регистрации, необходимо убедиться, что остальные компоненты этого модуля работоспособны

- Тесты с высоким приоритетом

Мы не можем рисковать работоспособностью самых важных функций приложения, поэтому необходимо дополнительно проверять и их

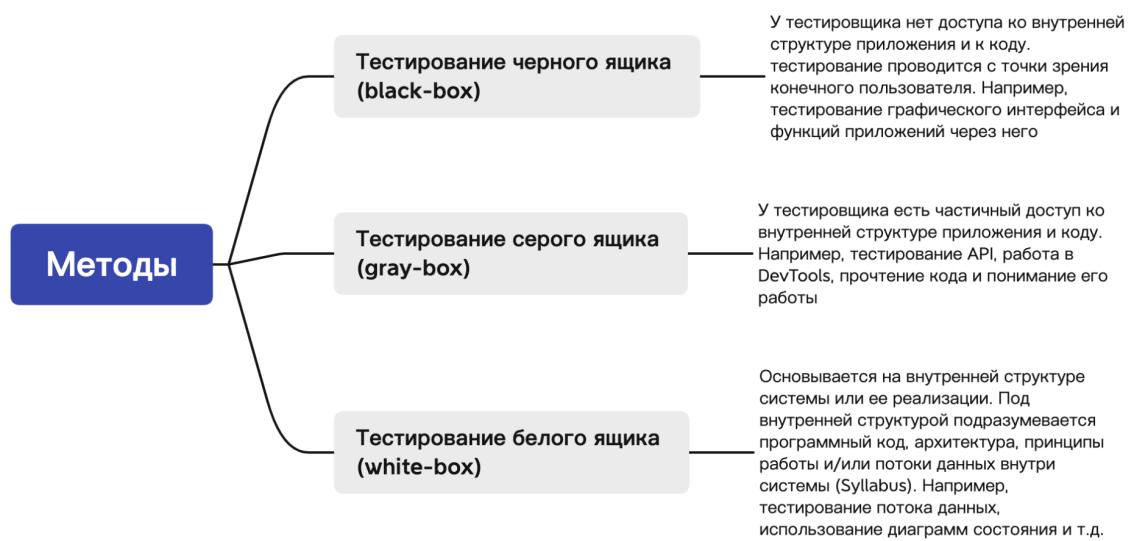
- Тесты, которые проверяют модули с наибольшей концентрацией дефектов

Согласно принципу скопления дефектов необходимо убедиться, что в уязвимой части приложения нет дополнительных проблем.

Методы тестирования

Ручные тестировщики чаще всего работают с методами черного и серого ящика.

Метод тестирования серого ящика не упоминается в ISTQB CTFL.



Функциональное и нефункциональное тестирование

Функциональное тестирование системы (functional testing) включает тесты по оценке функций, которые должна выполнять система. Данный вид тестирования можно проводить на всех уровнях.

Нефункциональное тестирование системы (non-functional testing) выполняется для оценки таких характеристик системы и программного обеспечения, как удобство использования, производительность или безопасность. За классификацией характеристик качества программного обеспечения следует обратиться к стандарту ИСО (ISO/IEC 25010). Данный вид тестирования можно проводить на всех уровнях.

Источник: *ISTQB CTFL Syllabus 2018*

Примечание автора курса

Функциональное тестирование отвечает на вопрос "Что делает система?".

Нефункциональное тестирование отвечает на вопрос "Как система это делает?".

Классификация по степени автоматизации

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

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. [Куликов]

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).

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.

Пример: ввод валидных значений в поле имя и пароль для успешной регистрации пользователя.

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).

Пример: ввод невалидных значений в поле имя и пароль, появление сообщения об ошибке, неуспешная регистрация.

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

Пример: нагрузка приложения выше его предела, чтобы оно перестало работать

В реальном тестировании следует использовать несколько правил:

1. Любое тестирование необходимо начинать с позитивных проверок. Наша первоочередная задача убедиться, что приложение работает в стандартных условиях и готово для использования.
2. Нельзя объединять позитивные и негативные тесты, так как это затрудняет локализацию дефектов.

Источник определений: [Святослав Куликов "Тестирование ПО. Базовый курс."](#)

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 запуску кода на исполнение

Статическое тестирование (static testing) - тестирование, которое не предполагает выполнение тестируемого компонента или системы

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

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

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

Источники: ISTQB CTFL Syllabus 2018 + Святослав Куликов "Тестирование ПО. Базовый курс."

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.

Источник: Святослав Куликов “Тестирование ПО. Базовый курс.”

Примеры от автора

1. **T. на основе тест-кейсов:** у вас есть четкие требования, вы написали тестовую документацию с конкретными шагами и проверками и приступаете к тестированию.
2. **Исследовательское Т.:** у вас есть требования или общее понимание работы продукта, вы набросали первичные проверки, которые будете дорабатывать по ходу тестирования.
3. **Свободное Т.:** у вас могут быть требования или полностью отсутствовать, вы не создаете набросок тестирования или официальную документацию, а тестируете приложение основываясь только на свой опыт.

Частные случаи: monkey testing, gorilla testing, о которых вы можете почитать дополнительно.

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

Источник: Святослав Куликов "Тестирование ПО. Базовый курс."

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

Также этот тип тестирования относится к нефункциональному тестированию.

Источник: Святослав Куликов "Тестирование ПО. Базовый курс."

Тестирование доступности (accessibility testing, A11Y) – 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

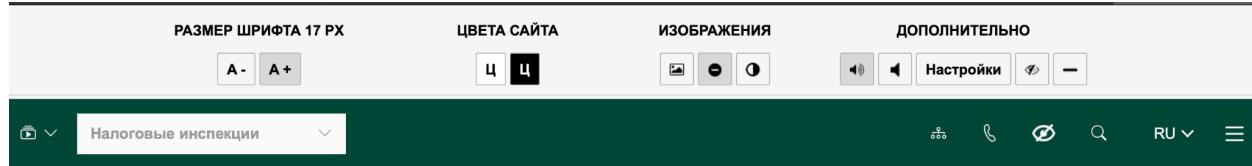
Доступность сайта можно тестировать специальными инструментами, например, WAVE, TAV, Accessibility Valet, Accessibility Developer Tools.

Также этот тип тестирования относится к нефункциональному тестированию.

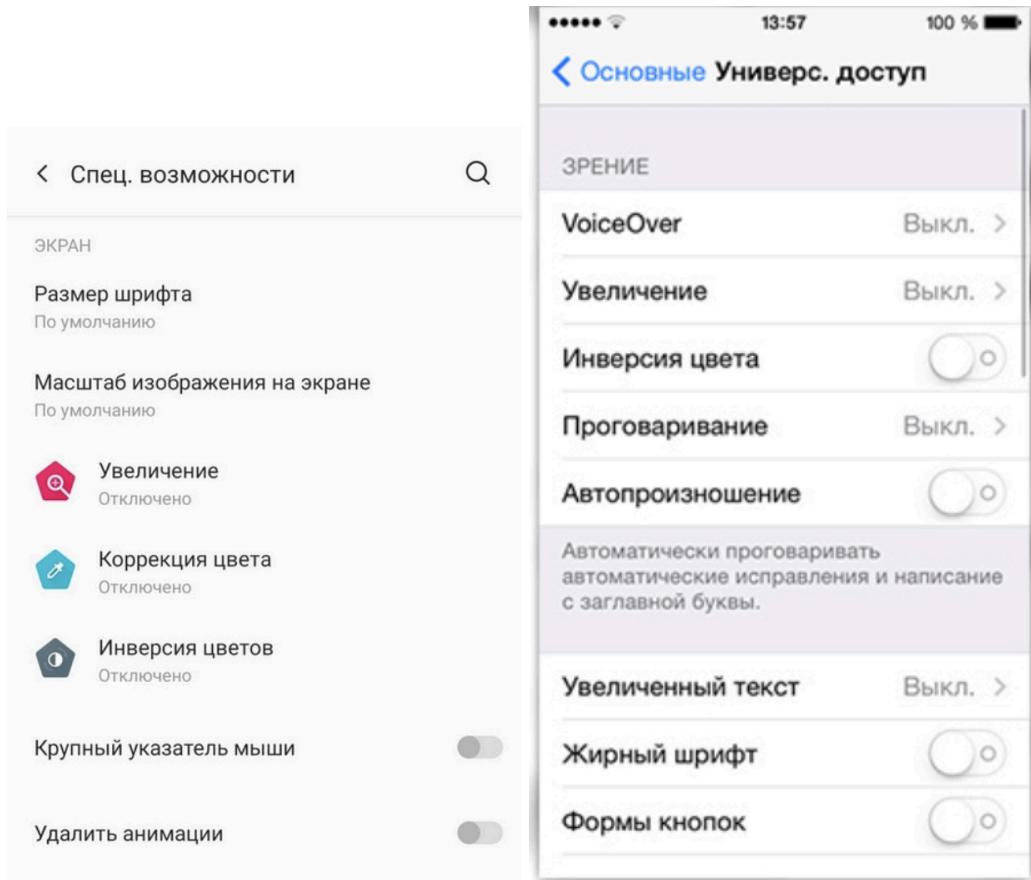
Источник: Святослав Куликов “Тестирование ПО. Базовый курс.”

Различные примеры того, что можно протестировать:

- Панель для изменения размера шрифта, цветов сайта, формата изображений, звуковой навигации в веб-приложении



- Универсальный доступ и специальные возможности на различных мобильных платформах



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)

Также этот тип тестирования относится к нефункциональному тестированию.

Источник: Святослав Куликов "Тестирование ПО. Базовый курс."

Дополнение от автора курса

Существует OWASP TOP 10, в котором собраны самые популярные и опасные уязвимости, а также рекомендации по борьбе с ними и тестированию. Подробнее о нем можно узнать в этом [видео](#)

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.

Оба типа тестирования относятся к нефункциональному тестированию.

Источник: Святослав Куликов “Тестирование ПО. Базовый курс.”

Различия i18n и l10n



Т. интернационализации

Готовность к адаптации в новых локалях

Примеры:

- Направление текста
- Возможность использования разных кодировок
- Конечный вид интерфейса

Т. локализации

Сама адаптация к локалям

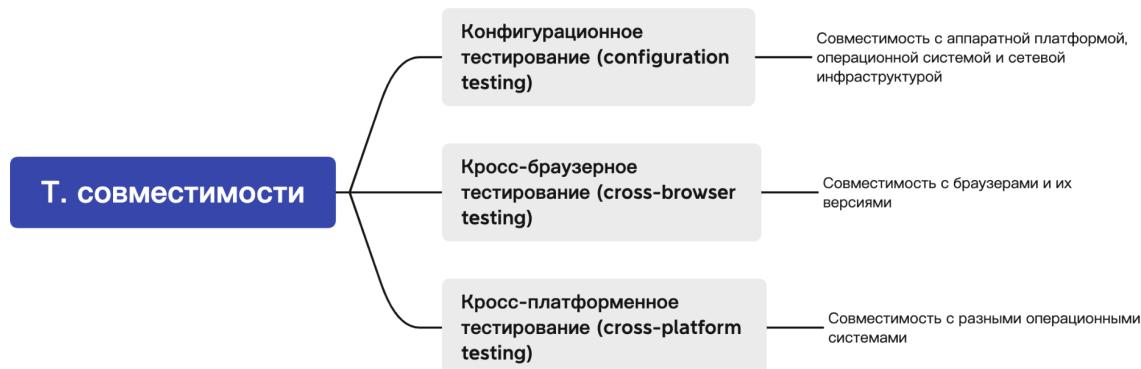
Примеры:

- Перевод
- Валюта
- Цвета
- Нормативы
- Единицы измерения
- Формат даты и времени

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

Также относится к нефункциональному тестированию.

Источник: Святослав Куликов "Тестирование ПО. Базовый курс."



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.

Также относится к нефункциональному тестированию.

Источник: Святослав Куликов "Тестирование ПО. Базовый курс."

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. Стressовое тестирование (stress testing)
5. Конкурентное тестирование (concurrency testing)

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.

Стressовое тестирование – исследование поведения приложения при непривычных 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.).

Все описанные виды тестирования также относятся к нефункциональным.

Источник: Святослав Куликов "Тестирование ПО. Базовый курс."

Примеры для различных типов тестирования производительности от автора курса



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. [ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119-1:2013]

ГОСТ Р ИСО/МЭК 25010—2015



Рисунок 4 — Модель качества продукта

Двоякая природа тестирования безопасности и производительности

Существует трактовка, по которой к функциональному тестированию можно отнести тестирование безопасности и тестирование производительности, 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.

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