Лекция 2.Виды и направления тестирования

# Классификация видов тестирования

Любая классификация составляется по определенному признаку, например:

* По **полу** люди классифицируются на мужчин и женщин;
* По **наличию автомобиля** люди делятся на тех, у кого есть автомобиль, и тех, у кого его нет.

Так же и тестирование можно классифицировать по очень большому количеству признаков. Но для чего вообще нужна классификация тестирования? Она позволяет упорядочить знания и значительно ускоряет процессы планирования тестирования и разработки тест-кейсов, а также позволяет оптимизировать трудозатраты за счёт того, что тестировщику не приходится изобретать очередной велосипед. Видов тестирования очень много, но мы рассмотрим только основные.

Итак, тестирование можно классифицировать…

## 1) По запуску кода на исполнение:

* Статическое

Статическое тестирование производится без запуска программного кода продукта. Тестирование осуществляется путем анализа программного кода (code review) или скомпилированного кода. Анализ может производиться как вручную, так и с помощью специальных инструментальных средств. Целью анализа является раннее выявление ошибок и потенциальных проблем в продукте. С помощью code review на раннем этапе могут быть выявлены ошибки в коде продукта. Как правило code review производится самими разработчиками. И тестировщиками при реализации автоматизированных тестов. Также к статическому тестированию относится тестирование документации.

* Динамическое

В отличии от статического, динамическое тестирование производится путем запуска продукта и проверки его функционала. Проверка осуществляется с помощью ручного или автоматического выполнения заранее подготовленного набора тестов.

## **2) По доступу к коду и** архитектуре **приложения:**

* Метод белого ящика

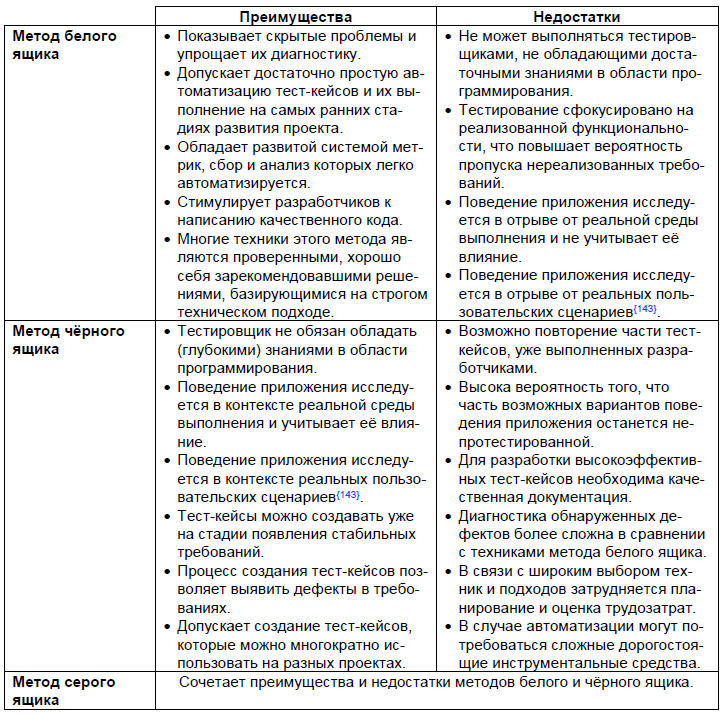
у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.

* Метод черного 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

комбинация методов белого ящика и чёрного ящика, состоящая в том, что к части кода и архитектуры у тестировщика доступ есть, а к части — нет

Таблица сравнения методов:

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. Методы белого и чёрного ящика не являются конкурирующими или взаимоисключающими — напротив, они гармонично дополняют друг друга, компенсируя таким образом имеющиеся недостатки.

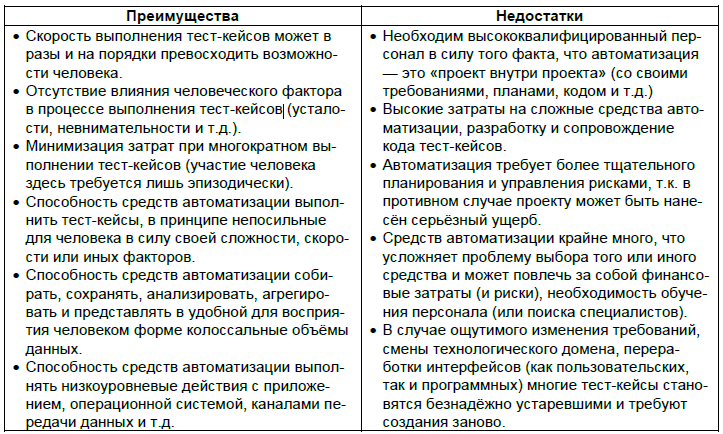
## **3) По** степени **автоматизации:**

* Ручное

Ручное тестирование – это тестирование, в котором тест-кейсы выполняются человеком вручную без использования средств автоматизации. Несмотря на то что это звучит очень просто, от тестировщика в те или иные моменты времени требуются такие качества, как терпеливость, наблюдательность, креативность, умение ставить нестандартные эксперименты, а также умение видеть и понимать, что происходит «внутри системы», т.е. как внешние воздействия на приложение трансформируются в его внутренние процессы.

* Автоматизированное

Автоматизированное тестирование – это набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования. Тест-кейсы частично или полностью выполняет специальное инструментальное средство, однако разработка тест-кейсов, подготовка данных, оценка результатов выполнения, написания отчётов об обнаруженных дефектах — всё это и многое другое по-прежнему делает человек

У автоматизированного тестирования есть много как сильных, так и слабых сторон. Преимущества и недостатки предоставлены в таблице ниже:  


Если же выразить все преимущества и недостатки автоматизации тестирования одной фразой, то получается, что автоматизация позволяет ощутимо увеличить тестовое покрытие, но при этом столь же ощутимо увеличивает риски.

## **4) По уровню** детализации **приложения:**

* Модульное (компонентное, юнит)

направлено на проверку отдельных небольших частей приложения, которые (как правило) можно исследовать изолированно от других подобных частей. При выполнении данного тестирования могут проверяться отдельные функции или методы классов, сами классы, взаимодействие классов, небольшие библиотеки, отдельные части приложения. Часто данный вид тестирования реализуется с использованием специальных технологий и инструментальных средств автоматизации тестирования, значительно упрощающих и ускоряющих разработку соответствующих тест-кейсов.

* Интеграционное

направлено на проверку взаимодействия между несколькими частями приложения (каждая из которых, в свою очередь, проверена отдельно на стадии модульного тестирования). К сожалению, даже если мы работаем с очень качественными отдельными компонентами, «на стыке» их взаимодействия часто возникают проблемы. Именно эти проблемы и выявляет интеграционное тестирование.

* Системное

направлено на проверку всего приложения как единого целого, собранного из частей, проверенных на двух предыдущих стадиях. Здесь не только выявляются дефекты «на стыках» компонентов, но и появляется возможность полноценно взаимодействовать с приложением с точки зрения конечного пользователя, применяя множество других видов тестирования

## **5) По степени важности тестируемых функций:**

* Дымовое тестирование или еще его называют Смоук

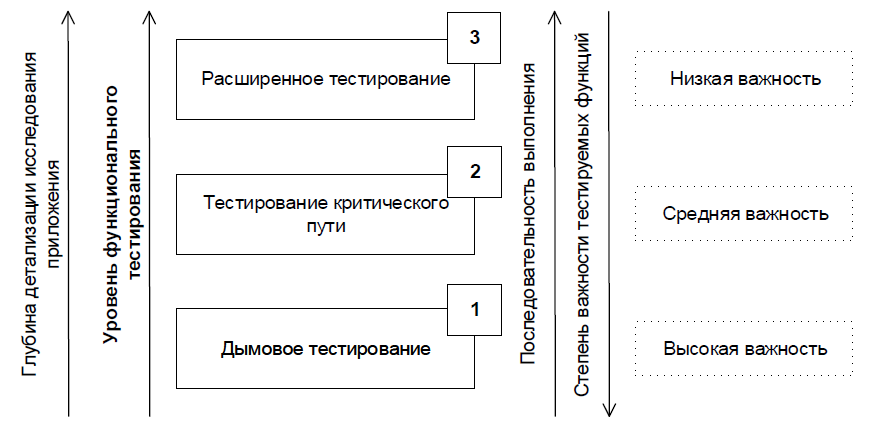
Дымовое тестирование направлено на проверку самой главной, самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной саму идею использования приложения (или иного объекта, подвергаемого дымовому тестированию). Дымовое тестирование проводится после выхода нового билда, чтобы определить общий уровень качества приложения и принять решение о (не)целесообразности выполнения тестирования критического пути и расширенного тестирования. Поскольку тест-кейсов на уровне дымового тестирования относительно немного, а сами они достаточно просты, но при этом очень часто повторяются, они являются хорошими кандидатами на автоматизацию.

* Тестирование критического пути

Тестирование критического пути направлено на исследование функциональности, используемой типичными пользователями в типичной повседневной деятельности. Существует большинство пользователей, которые чаще всего используют некое подмножество функций приложения. Именно эти функции и нужно проверить, как только мы убедились, что приложение «в принципе работает» (дымовой тест прошёл успешно). Если по каким-то причинам приложение не выполняет эти функции или выполняет их некорректно, очень многие пользователи не смогут достичь множества своих целей

* Расширенное тестирование

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. Ещё одним направлением исследования в рамках данного тестирования являются нетипичные, маловероятные, экзотические случаи и сценарии использования функций и свойств приложения, затронутых на предыдущих уровнях



## **6) По принципам работы с приложением:**

* Позитивное

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. Если позитивные тест-кейсы завершаются ошибками, это тревожный признак — приложение работает неверно даже в идеальных условиях (и можно предположить, что в неидеальных условиях оно работает ещё хуже). Для ускорения тестирования **несколько позитивных тест-кейсов можно объединять** (например, перед отправкой заполнить все поля формы верными значениями) — иногда это может усложнить диагностику ошибки, но существенная экономия времени компенсирует этот риск

* Негативное

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). Поскольку в реальной жизни таких ситуаций значительно больше (пользователи допускают ошибки, злоумышленники осознанно «ломают» приложение, в среде работы приложения возникают проблемы и т.д.), негативных тест-кейсов оказывается значительно больше, чем позитивных (иногда — в разы или даже на порядки). В отличие от позитивных **негативные тест-кейсы не стоит объединять**, т.к. подобное решение может привести к неверной трактовке поведения приложения и пропуску (необнаружению) дефектов. Негативное тестирование направлено на исследование работы приложения

## 7) по **степени** формализации

* Тестирование **на основе тест-кейсов**

формализованный подход, в котором тестирование производится на основе заранее подготовленных тест-кейсов, наборов тест-кейсов и иной документации. Это самый распространённый способ тестирования, который также позволяет достичь максимальной полноты исследования приложения за счёт строгой систематизации процесса, удобства применения метрик и широкого набора выработанных за десятилетия и проверенных на практике рекомендаций.

* Исследовательское тестирование

частично формализованный подход, в рамках которого тестировщик выполняет работу с приложением по выбранному сценарию{143}, который, в свою очередь, дорабатывается в процессе выполнения с целью более полного исследования приложения. Ключевым фактором успеха при выполнении исследовательского тестирования является именно работа по сценарию, а не выполнение разрозненных бездумных операций

* Свободное (интуитивное) тестирование

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 testing159) для спонтанного выполнения с приложением действий, которые, как он считает, могут обнаружить ошибку. Этот вид тестирования используется редко и исключительно как дополнение к полностью или частично формализованному тестированию в случаях, когда для исследования некоторого аспекта поведения приложения (пока?) нет тест-кейсов

## **8) По целям и задачам:**

Данный вид классификации является искусственным, поскольку «внутри» речь будет идти об одних и тех же видах тестирования, отличающихся в данном контексте лишь концентрацией на соответствующих функциях и особенностях приложения, использованием специфических инструментов и отдельных техник.

* Позитивное тестирование
* Негативное тестирование
* Функциональное тестирование

вид тестирования, направленный на проверку корректности работы функциональности приложения (корректность реализации функциональных требований). Часто функциональное тестирование ассоциируют с тестированием по методу чёрного ящика, однако и по методу белого ящика вполне можно проверять корректность реализации функциональности.

* Нефункциональное тестирование

вид тестирования, направленный на проверку нефункциональных особенностей приложения (корректность реализации нефункциональных требований), таких как удобство использования, совместимость, производительность, безопасность и т.д.

* Инсталляционное тестирование

тестирование, направленное на выявление дефектов, влияющих на протекание стадии установки приложения. В общем случае такое тестирование проверяет множество сценариев и аспектов работы инсталлятора в таких ситуациях, как: новая среда исполнения, в которой приложение ранее не было инсталлировано; обновление существующей версии («апгрейд»); изменение текущей версии на более старую («даунгрейд»); повторная установка приложения с целью устранения возникших проблем («переинсталляция»); повторный запуск инсталляции после ошибки, приведшей к невозможности продолжения инсталляции; удаление приложения; установка нового приложения из семейства приложений; автоматическая инсталляция без участия пользователя)

* Регрессионное тестирование

тестирование, направленное на проверку того факта, что в ранее работоспособной функциональности не появились ошибки, вызванные изменениями в приложении или среде его функционирования - исправление одной ошибки с большой вероятностью влечёт появление новой. Поэтому регрессионное тестирование является неотъемлемым инструментом обеспечения качества и активно используется практически в любом проекте.

* Повторное тестирование

выполнение тест-кейсов, которые ранее обнаружили дефекты, с целью подтверждения устранения дефектов. Фактически этот вид тестирования сводится к действиям на финальной стадии жизненного цикла отчёта о дефекте, направленным на то, чтобы перевести дефект в состояние «проверен» и «закрыт».

* Приемочное тестирование

формализованное тестирование, направленное на проверку приложения с точки зрения конечного пользователя/заказчика и вынесения решения о том, принимает ли заказчик работу у исполнителя (проектной команды)

* Тестирование удобства пользователя

тестирование, направленное на исследование того, насколько конечному пользователю понятно, как работать с продуктом, а также на то, насколько ему нравится использовать продукт. И это не оговорка — очень часто успех продукта зависит именно от эмоций, которые он вызывает у пользователей. Для эффективного проведения этого вида тестирования требуется реализовать достаточно серьёзные исследования с привлечением конечных пользователей, проведением маркетинговых исследований и т.д.)

* Тестирование безопасности

тестирование, направленное на проверку способности приложения противостоять злонамеренным попыткам получения доступа к данным или функциям, права на доступ к которым у злоумышленника нет.

* Тестирование совместимости

тестирование, направленное на проверку способности приложения работать в указанном окружении. Здесь может проверяться совместимость с аппаратной платформой, с разными браузерами, мобильными устройствами.

* Исчерпывающее тестирование

тестирование приложения со всеми возможными комбинациями всех возможных входных данных во всех возможных условиях выполнения. Для сколь бы то ни было сложной системы нереализуемо, но может применяться для проверки отдельных крайне простых компонентов.

* Тестирование надёжности

тестирование способности приложения выполнять свои функции в заданных условиях на протяжении заданного времени или заданного количества операций.

* Тестирование отказоустойчивости

тестирование, заключающееся в эмуляции или реальном создании критических ситуаций с целью проверки способности приложения задействовать соответствующие механизмы, предотвращающие нарушение работоспособности, производительности и повреждения данных.

* Тестирование производительности

исследование показателей скорости реакции приложения на внешние воздействия при различной по характеру и интенсивности нагрузке. Нагрузочное тестирование (исследование способности приложения сохранять заданные показатели качества при нагрузке в допустимых пределах и некотором превышении этих пределов (определение «запаса прочности») – это один из подвидов тестирования производительности.

Как уже оговаривали ранее видов тестирования очень много. Рассмотрели только основные.

## **9) По привлечению конечных пользователей**

* **Альфа-тестирование** - выполняется внутри организации-разработчика с возможным частичным привлечением конечных пользователей. Может являться формой внутреннего приёмочного тестирования. Суть этого вида вкратце: продукт уже можно периодически показывать внешним пользователям, но он ещё достаточно «сырой», потому основное тестирование выполняется организацией-разработчиком.
* **Бета-тестирование** - выполняется вне организации-разработчика с активным привлечением конечных пользователей/заказчиков. Может являться формой внешнего приёмочного тестирования. Суть этого вида вкратце: продукт уже можно открыто показывать внешним пользователям, он уже достаточно стабилен, но проблемы всё ещё могут быть, и для их выявления нужна обратная связь от реальных пользователей.
* **Гамма-тестирование** — финальная стадия тестирования перед вы пуском продукта, направленная на исправление незначительных дефектов, обнаруженных в бета-тестировании. Как правило, также выполняется с максимальным привлечением конечных пользователей/заказчиков. Может являться формой внешнего приёмочного тестирования. Суть этого вида вкратце: продукт уже почти готов, и сейчас обратная связь от реальных пользователей используется для устранения последних недоработок.

# Классификация по хронологии

Единой хронологии тестирования, которая бы подходила для любой методологии управления проектами, любого проекта, любой стратегии тестирования не существует.

Общая универсальная логика последовательности тестирования состоит в том, чтобы начинать исследование **каждой задачи с простых позитивных тест-кейсов, к которым постепенно добавлять негативные** (но тоже достаточно простые). **Лишь после того, как наиболее типичные ситуации покрыты простыми тест-кейсами, следует переходить к более сложным** (опять же, начиная с позитивных). Такой подход — не догма, но к нему стоит прислушаться, т.к. углубление на начальных этапах в негативные (к тому же — сложные) тест-кейсы может привести к ситуации, в которой приложение отлично справляется с кучей неприятностей, но не работает на элементарных повседневных задачах!

Например, последовательность тестирования, построенная по концентрации внимания на требованиях и их составляющих:

1) Тестирование требований, которое может варьироваться от беглой оценки в стиле «всё ли нам понятно» до весьма формальных подходов, в любом случае первично по отношению к тестированию того, как эти требования реализованы.

2) Тестирование реализации функциональных составляющих требований логично проводить до тестирования реализации нефункциональных составляющих, т.к. если что-то просто не работает, то проверять производительность, безопасность, удобство и прочие нефункциональные составляющие бессмысленно, а чаще всего и вовсе невозможно.

3) Тестирование реализации нефункциональных составляющих требований часто становится логическим завершением проверки того, как реализовано то или иное требование.

Типичные общие сценарии используются, когда нет явных предпосылок к реализации другой стратегии. Например:

* **Типичный общий сценарий 1**:

1) Дымовое тестирование

2) Тестирование критического пути

3) Расширенное тестирование

* **Типичный общий сценарий 2**:

1) Модульное тестирование

2) Интеграционное тестирование

3) Системное тестирование

В завершение отметим еще раз, что классификация видов тестирования необходима для упорядочения огромного объёма информации о различных видах деятельности тестировщиков и для облегчения запоминания соответствующих фактов.