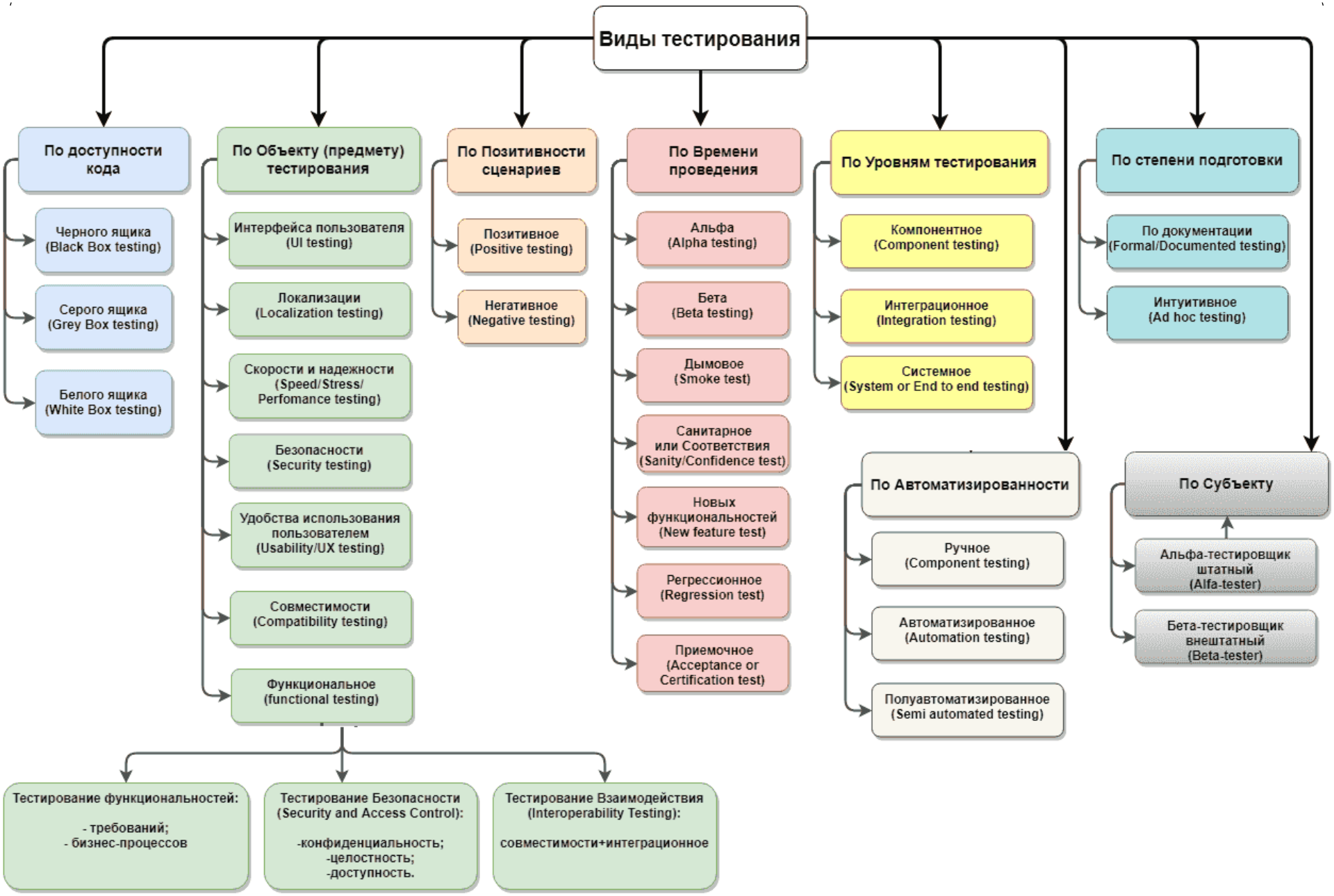
Лекция 2. Классификация видов тестирования ПО


Виды тестирования
По доступности кода
Черного ящика (Black Box testing)
Серого ящика (Grey Box testing)
Белого ящика (White Box testing)
По Объекту (предмету) тестирования
Интерфейса пользователя (UI testing)
Локализации (Localization testing)
Скорости и надежности (Speed/Stress/ Perfomance testing)
Безопасности (Security testing)
Удобства использования пользователем (Usability/UX testing)
Совместимости (Compatibility testing)
Функциональное (functional testing)
Тестирование функциональностей:
- требований;
- бизнес-процессов
Тестирование Безопасности (Security and Access Control):
-конфиденциальность;
-целостность;
-доступность.
Тестирование Взаимодействия (Interoperability Testing):
совместимости+интеграционное
По Позитивности сценариев
Позитивное (Positive testing)
Негативное (Negative testing)
По Времени проведения
Альфа (Alpha testing)
Бета (Beta testing)
Дымовое (Smoke test)
Санитарное или Соответствия (Sanity/Confidence test)
Новых функциональностей (New feature test)
Регрессионное (Regression test)
Приемочное (Acceptance or Certification test)
По Уровням тестирования
Компонентное (Component testing)
Интеграционное (Integration testing)
Системное (System or End to end testing)
По Автоматизированности
Ручное (Component testing)
Автоматизированное (Automation testing)
Полуавтоматизированное (Semi automated testing)
По степени подготовки
По документации (Formal/Documented testing)
Интуитивное (Ad hoc testing)
По Субъекту
Альфа-тестировщик штатный (Alfa-tester)
Бета-тестировщик внештатный (Beta-tester)

1) По знанию внутреннего устройства программы:

- тестирование по методу «черного ящика»;
- тестирование по методу «белого ящика»;
- тестирование по методу «серого ящика».

2) По запуску программы на выполнение :

- статическое тестирование;
- динамическое тестирование.

3) По степени автоматизации тестирования :

- ручное тестирование;
- автоматизированное тестирование;
- полуавтоматизированное тестирование.

4) По хронологии тестирования:

- до передачи пользователю;
- после передачи пользователю.

5) По субъекту тестирования:

- тестирование, проводимое программистом (например, обзоры кода, модульное тестирование);
- тестирование, проводимое тестировщиком (например, функциональное тестирование, тестирование производительности);
- случайное тестирование (англ., ad hoc testing, проводится не участником проекта без предварительной подготовки с целью найти случайные ошибки в программе);
- приемочные испытания (проводятся, как правило, заказчиком).

6) По объекту тестирования:

- тестирование требований к программному продукту;
- тестирование исходного кода (например, обзоры кода);
- модульное тестирование (проверка отдельных программных модулей);
- интеграционное тестирование (проверка взаимодействия программных модулей);
- объектно-ориентированное тестирование (тестирование классов);
- функциональное тестирование (проверка работы заявленной функциональности);

- системное тестирование (проверка работы программы в реальном системном окружении);
- тестирование интерфейса программы;
- тестирование удобства использования;
- локализационное тестирование (проверка работы программы при переходах на другие иностранные языки);
- тестирование производительности;
- тестирование безопасности;
- тестирование на совместимость (или кроссплатформенное и кроссбраузерное тестирование – проверка работы программы на разных операционных системах и в разных браузерах).

7) По позитивности тестов:

- позитивное тестирование;
- негативное тестирование.

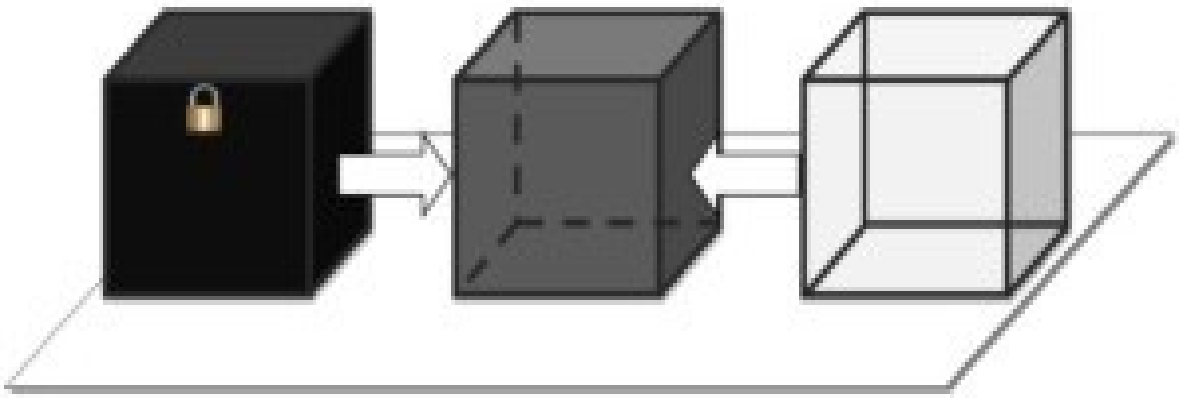
8) По степени изолированности компонент (по уровням тестирования):

- модульное тестирование;
- интеграционное тестирование;
- системное тестирование.

9) По степени подготовки к тестированию:

- [illegible]по тестовым случаям;
- случайное тестирование.

1. White/Black/Grey Box-тестирование



Black Box – Черный ящик

Мы не знаем, как устроена тестируемая система.

Тестирование методом «черного ящика», также известное как тестирование, основанное на спецификации или тестирование поведения – техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы.

тестирование черного ящика – это:

- тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы;
- тест-дизайн, основанный на технике черного ящика – процедура написания или выбора тест-кейсов на основе анализа функциональной или нефункциональной спецификации компонента или системы без знания ее внутреннего устройства.

Целью этой техники является поиск ошибок в таких категориях:

- неправильно реализованные или недостающие функции;
- ошибки интерфейса;
- ошибки в структурах данных или организации доступа к внешним базам данных;
- ошибки поведения или недостаточная производительность системы;

Нужно концентрироваться на том, что программа делает, а не на том, как она это делает.

Пример:

Тестировщик проводит тестирование веб-сайта, не зная особенностей его реализации, используя только предусмотренные разработчиком поля ввода и кнопки. Источник ожидаемого результата – спецификация.

Поскольку это тип тестирования, то он может включать и другие его виды. Тестирование черного ящика может быть как функциональным, так и нефункциональным. Функциональное тестирование предполагает проверку работы функций системы, а нефункциональное – общие характеристики нашей программы.

Техника черного ящика применима на всех уровнях тестирования.

Техники тест-дизайна, основанные на использовании черного ящика, включают:

- классы эквивалентности;
- анализ граничных значений;
- таблицы решений;
- диаграммы изменения состояния;
- тестирование всех пар.

Преимущества:

- тестирование производится с позиции конечного пользователя и может помочь обнаружить неточности и противоречия в спецификации;
- тестировщику нет необходимости знать языки программирования и углубляться в особенности реализации программы;
- тестирование может производиться специалистами, независимыми от отдела разработки, что помогает избежать предвзятого отношения;
- можно начинать писать тест-кейсы, как только готова спецификация.

Недостатки:

- тестируется только очень ограниченное количество путей выполнения программы;
- без четкой спецификации (а это скорее реальность на многих проектах) достаточно трудно составить эффективные тест-кейсы;
- некоторые тесты могут оказаться избыточными, если они уже были проведены разработчиком на уровне модульного тестирования.

White Box – Белый ящик

Нам известны все детали реализации тестируемой программы.

Метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику. Выбираем входные значения, основываясь на знании кода, который будет их обрабатывать. Знаем, каким должен быть результат этой обработки. Тестирование белого ящика – углубление во внутреннее устройство системы за пределы ее внешних интерфейсов.

тестирование белого ящика – это:

- тестирование, основанное на анализе внутренней структуры компонента или системы;
- тест-дизайн, основанный на технике белого ящика – процедура написания или выбора тест-кейсов на основе анализа внутреннего устройства системы или компонента.

Тестируемая программа для тестировщика – прозрачный ящик, содержимое которого он прекрасно видит.

Пример:

Тестировщик, который, как правило, является программистом, изучает реализацию кода поля ввода на веб-странице, определяет все предусмотренные (как правильные, так и неправильные) и не предусмотренные пользовательские вводы и сравнивает фактический результат выполнения программы с ожидаемым. При этом ожидаемый результат определяется именно тем, как должен работать код программы.

Тестирование методом белого ящика похоже на работу механика, который изучает двигатель машины, чтобы понять, почему она не заводится.

Преимущества:

- тестирование может производиться на ранних этапах: нет необходимости ждать создания пользовательского интерфейса;
- можно провести более тщательное тестирование с покрытием большого количества путей выполнения программы.

Недостатки:

- для выполнения тестирования белого ящика необходимо большое количество специальных знаний;
- при использовании автоматизации тестирования на этом уровне поддержка тестовых скриптов может оказаться достаточно накладной, если программа часто изменяется.

Grey Box – Серый ящик

Нам известны только некоторые особенности реализации тестируемой системы.

Метод тестирования программного обеспечения, который предполагает комбинацию White Box и Black Box подходов. Т.е. внутреннее устройство программы нам известно лишь частично. Предполагается, например, доступ ко внутренней структуре и алгоритмам работы ПО для написания максимально эффективных тест-кейсов, но само тестирование проводится с помощью техники черного ящика, то есть с позиции пользователя.

Пример:

Тестировщик изучает код программы с тем, чтобы лучше понимать принципы ее работы и изучить возможные пути ее выполнения. Такое знание поможет написать тест-кейс, который наверняка будет проверять определенную функциональность.

2. Статическое и динамическое тестирование

Статическое тестирование

- **Статистическое тестирование** –тип тестирования, который предполагает, что программный код во время тестирования не будет выполняться. При этом, само тестирование может быть как ручным, так и автоматизированным.
- Статическое тестирование начинается на ранних этапах жизненного цикла ПО и является, соответственно, частью процесса верификации. Для этого типа тестирования в некоторых случаях даже не нужен компьютер, например, при проверке требований.

Большинство статических техник могут быть использованы для «тестирования» любых форм документации, включая вычитку кода, инспекцию проектной документации, функциональной спецификации и требований.

Даже статическое тестирование может быть автоматизировано, например, можно использовать автоматические средства проверки синтаксиса программного кода.

Виды статического тестирования:

- вычитка исходного кода программы;
- проверка требований.

Динамическое тестирование

- **Динамическое тестирование** – тип тестирования, который предполагает запуск программного кода. Таким образом, анализируется поведение программы во время ее работы.
- Для выполнения динамического тестирования необходимо, чтобы тестируемый программный код был написан, скомпилирован и запущен. При этом, может выполняться проверка внешних параметров работы программы: загрузка процессора, использование памяти, время отклика и т.д., то есть ее производительность.
- Динамическое тестирование является частью процесса валидации программного обеспечения.

Кроме того, динамическое тестирование может включать разные подвиды, каждый из которых зависит от:

- Доступа к коду (тестирование черным, белым и серым ящиками).
- Уровня тестирования (модульное, интеграционное, системное и приемочное тестирование).
- Сферы использования приложения (функциональное, нагрузочное, тестирование безопасности и пр.).

3. Ручное и автоматизированное тестирование

При **ручном тестировании (manual testing)** тестировщики вручную выполняют тесты, не используя никаких средств автоматизации. Ручное тестирование – самый низкоуровневый и простой тип тестирования, не требующий большого количества дополнительных знаний.

Перед тем, как автоматизировать тестирование любого приложения, необходимо сначала выполнить серию тестов вручную. Это требует значительных усилий, но без него мы нельзя убедиться в том, возможна ли автоматизация в принципе. Один из фундаментальных принципов тестирования: 100% автоматизация невозможна.

Автоматизированное тестирование (automation testing) предполагает использование специального программного обеспечения (помимо тестируемого) для контроля выполнения тестов и сравнения ожидаемого фактического результата работы программы. Этот тип тестирования помогает автоматизировать часто повторяющиеся, но необходимые для максимизации тестового покрытия, задачи.

Виды автоматизированного тестирования:

- **Автоматизация тестирования кода** – тестирование на уровне программных модулей, классов и библиотек (фактически, автоматические юнит-тесты).
- **Автоматизация тестирования графического пользовательского интерфейса** – специальная программа (фреймворк автоматизации тестирования) позволяет генерировать пользовательские события– нажатия клавиш, клики мышкой, и отслеживание реакции программы на эти действия: соответствует ли она спецификации.
- **Автоматизация тестирования API** – тестирование программного интерфейса программы. Тестируются интерфейсы, предназначенные для взаимодействия, например, с другими программами или с пользователем. Как правило используются специальные фреймворки.

Сравнение ручного и автоматизированного тестирования

- **Автоматизация сохраняет время, силы и деньги.** Автоматизированный тест можно запускать снова и снова, прилагая минимум усилий.
- **Вручную можно протестировать практически любое приложение, в то время как автоматизировать стоит только стабильные системы.** Автоматизированное тестирование используется, главным образом, для регрессии. Кроме того, некоторые виды тестирования, могут быть выполнены только вручную.
- **Мануальное тестирование может быть повторяющимся и скучным.** В то же время, автоматизация может помочь этого избежать – за вас все сделает компьютер.

4. Альфа и бета тестирование

До передачи пользователю:

- альфа-тестирование,
- тест приемки,
- тестирование новых функциональностей,
- регрессионное тестирование;

После передачи пользователю:

- бета-тестирование.

Альфа-тестирование

тестирование ПО, которое проводится на ранней стадии разработки продукта или когда процесс разработки приближается к завершению.

Альфа-тесты проводят сотрудники компании, она предоставляет для этого тестовое окружение.

Тест приемки

проверяет соответствие системы требованиям и проводится с целью определения, удовлетворяет ли система приемочным критериям; вынесения решения заказчиком или другим уполномоченным лицом о приемке приложения.

Тестирование новых функциональностей

проверяет корректность реализации новых задач.

Регрессионное тестирование

направлено 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), для подтверждения того факта, что существующая ранее функциональность работает, как и прежде. Регрессионными могут быть как функциональные, так и нефункциональные тесты

Бета-тестирование

означает тестирование, при котором разработка и тестирование по существу завершены, и до окончательного выпуска продукта необходимо обнаружить оставшиеся ошибки и проблемы.

Бета-тесты проводят реальные пользователи в реальном окружении, чаще всего собственном (на своем компьютере/смартфоне).

Выполняется распространение предварительной версии для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок.

Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.