Виды тестирования

- **По доступу кода**
  - Метод белого ящика
    - Метод, когда тестировщик имеет доступ к коду и может тестировать не только front, но и backend
  - Метод черного ящика
    - Метод, когда тестировщик не имеет доступа к коду, работа идет только с front'ом
  - Метод серого ящика
    - Метод, когда тестируется и frontend и backend. Возможна проверка соответствия визуальной составляющей и программного кода.
- **По запуску кода на исполнение**
  - Статическое
    - Направлено на тестирование технических требований. Может использоваться на этапе анализа документации.
  - Динамическое
    - Метод тестирования на существующем программном коде.
- **По уровням детализации приложения**
  - Модульное тестирование (Unit test)
    - Тестирование изолированного элемента, называемого "модулем" с полным доступом к программному коду. Как правило, пишется разработчиками, так как в этом методе тестируется исходный код.
  - Интеграционное тестирование
    - Направлен на проверку корректности взаимодействия нескольких модулей.
  - Системное тестирование
    - Процесс тестирования системы, на котором производится не только функциональное тестирование, но и оценка качеств системы (устойчивость, надежность, безопасность, производительность). Работает ли система корректно после интеграции всех модулей.
  - Приемочное тестирование
    - Проверяется соответствие системы требованиям, потребностям и бизнес-процессам пользователя. Может проводиться как самим тестировщиком, так и заказчиком.
- **Классификация по степени автоматизации**
  - Ручное (мануальное) тестирование
    - Полностью ручное тестирование каждого элемента, когда тестировщик сам нажимает каждую кнопку.
  - Автоматизированное тестирование
    - Написание функций, которые сами ищут и проверяют необходимые элементы. Для этого необходимо использовать языки программирования и специализированные программы.
- **По принципам работы с приложением**
  - Позитивное тестирование
    - Проведение теста, в котором при проведении правильных действий мы получаем ожидаемый результат.
  - Негативное тестирование
    - Проведение теста, в котором при проведении правильных действий мы получаем результат, отличный от ожидаемого.
- **По уровню функционального тестирования**
  - Smoke-test
    - Проверка запуска программного кода
  - Санитарное тестирование
    - Проверка основных бизнес-требований
      - Как правило, производится вместе со Smoke-тестированием, так как нецелесообразно проводить отдельное тестирование только на запуск программного кода
  - Тестирование критического пути
    - Проверка функциональности, используемой пользователями в повседневной деятельности.
  - Расширенное тестирование
    - Направлено на исследование всей заявленной в требованиях функциональности. По своей сути, напоминает приемочное тестирование.
- **В зависимости от исполнителей**
  - Альфа-тест
    - Тестирование закрытое, производится членами команды. Как правило, закрыто подпиской о неразглашении.
  - Бета-тест
    - Открытое тестирование, в котором можно использовать приглашенных пользователей.
- **В зависимости от целей тестирования**
  - Функциональное тестирование
    - Модульное тестирование
    - Smoke-test
    - Санитарное тестирование
    - Регрессионное тестирование
      - Проводится для того, чтобы убедиться, что добавление нового кода, какого-либо улучшения или исправление какого-либо бага не нарушило существующую функциональность и не вызывает какого либо типа нестабильности. Проверка старого функционала.
    - Альфа-тест
    - Бета-тест
    - Системное тестирование
    - Интеграционное тестирование
    - Критическое тестирование
    - E2E тест
      - E2E-тестирование это подтип функционального, проверка всей системы «из конца в конец», end-to-end, поэтому такое название. Таких тестов еще меньше количественно, но они еще сложнее чем интеграционные и тем более модульные (и требуют больше опыта от тестировщика).
    - Тестирование API
      - Как и юнит-тестирование, этот тип относится к так называемому «code level testing», то есть имеет дело непосредственно с исходным кодом приложения. Разница с юнит- в том, что юнит-тесты обычно делают разработчики, а API тестирует QA-команда.
  - Нефункциональное тестирование
    - Тестирование производительности
      - Тестирование для определения, какие факторы как влияют на производительность.
    - Нагрузочное тестирование
      - Тестирование, при котором проверяется работа системы свыше заданного лимита и когда она даст сбой.
    - Стресс-тест
      - Тестирование, при котором проверяется стабильность работы системы при максимально допустимой нагрузке.
    - Инсталляционное тестирование
      - Тестирование направлено на проверку успешной установки и настройки обновления и удаления приложения.
        - Больше предназначено для мобильной версии.
    - Тестирование интерфейса (UI)
      - Шрифты
      - Цвета
      - Размеры элементов
      - Тестирование визуальной части
    - Тестирование удобства использования (UX)
      - Тестирование того, насколько удобно, привлекательно, приятно использовать приложение
    - Локализация
      - Проверка адаптации ПО для определенной аудитории, с учетом ее культурного наследия
    - Тестирование надежности
      - Проверка работоспособности приложения при длительном тестировании, например, на средней или стрессовой загрузке.
    - Тестирование безопасности
      - Тестирование проверки безопасности системы (не санкционный доступ, вирусы, хакерские атаки и т.д.)
    - Восстановление
      - Тестирование, как система сама восстанавливается после сбоя
    - Тестирование документации
- **Ad hog testing**
  - Buddy Testing
    - Суть Buddy Testing в том, что как минимум два «компаньона» (в переводе с английского buddy — приятель, компаньон) одновременно пытаются выявить баги в одном и том же модуле.

      Buddy Testing можно считать комбинацией системного и модульного тестирования. Оно проводится после юнит-тестирования модуля.

      Компаньонами обычно бывают разработчик и тестировщик. Они вместе работают над модулем для создания валидных тест-кейсов.
  - Monkey Testing
    - «Обезьянье» тестирование часто применяют при проверке отдельных модулей. Суть его в том, что тестировщики тестируют приложение или продукт случайным образом, без тест-кейсов.

      Основная задача тестировщика — проанализировать работу приложения совершенно рандомным образом. Это помогает удостовериться, что система способна выдержать любой сбой.

      Тестировщики передают в программу случайные входные данные и наблюдают за результатами. Выходные данные помогают выявить ошибки, несоответствия и сбои в системе.
  - Парное тестирование
    - Парное тестирование похоже на Buddy Testing, но здесь над модулем работают два тестировщика, а не тестировщик и разработчик. Кроме того, Buddy Testing — комбинация модульного и системного тестирования, а парное тестирование — чисто модульное.

      Суть парного тестирования в том, что тестировщики работают вместе на одной машине и при этом делятся идеями и знаниями. Последнее особенно полезно, когда уровень знаний у тестировщиков различается. В таком случае менее опытный может многому научиться у старшего коллеги.

      Работая в паре, тестировщики могут распределять роли: скажем, один проводит тесты, а другой делает записи.

Presented with xmind