Метод Черного и Белого Ящика

Лекция 2

Что такое «черный ящик» согласно терминологии

Black-box тестирование – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы. *Метод тестирования «черного ящика»* – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство.

Где используется метод «черного ящика»?

- **1. Интеграционное тестирование.**
  Тестирование, в котором программные и аппаратные компоненты объединяются и тестируются для оценки взаимодействия между ними. При использовании метода «черного ящика» тестировщик проверяет, корректно ли работают все компоненты в целом тогда, когда они интегрированы в большую систему. И действительно, нормальная работа каждой составляющей по отдельности – это еще не гарантия того, что они будут работать вместе в рамках всего проекта. Например, данные могут не отправиться через интерфейс, или интерфейс не отработает согласно документации. При планировании таких тестов тестировщики опираются на спецификацию.
- **2. Функциональное тестирование.**
  Используя этот метод, тестировщик проверяет, выполняет ли программное обеспечение все заявленные функции и требования клиента в полном объеме согласно документации.

- **3. Стресс-тестирование.**
  Предположим, что у нас есть букмекерская онлайн-контора, в документации к которой заявлена возможность одновременной регистрации 1000 пользователей. В этом случае стрессовым тестированием будет непрерывный поток автоматизированных регистраций (как минимум, 1000 регистраций в минуту) на протяжении 12 часов.
- **4. Usability-тестирование.**
  Пусть в упомянутой букмекерской конторе есть функционал «Купон»: мы проверяем, сколько времени уходит у пользователя для добавления ставки в купон, ввода суммы и завершения ставки.
- **5. Тестирование производительности.**
  Таким видом тестирования мы можем проверить: есть ли утечки памяти, насколько быстро система работает и выдает обратную связь, не потребляет ли наше ПО слишком много трафика и не создает ли избыточное количество подключений.
- **6. Приемочное тестирование.**
  После проверки ПО тестировщиками его отдают заказчику, который запускает приемочные тесты «черного ящика» на основе ожиданий от функциональности. Как правило, набор тестов в этом случае определяет сам заказчик, за ним же остается право отказаться от приемки (если его не устроили результаты тестирования).

- **7. Регрессионное тестирование.**
  Проводится на протяжении всего цикла разработки. Цель такого тестирования – проверить работоспособность нового кода и выяснить, не привел ли он к ошибкам или поломкам в старом функционале.

При выборе набора регрессионных тестов следует использовать следующие рекомендации:

- делаем репрезентативную выборку тестов, в которой используются все функции ПО;
- выбираем тесты, сосредоточенные на программных компонентах/функциях, которые подверглись изменениям;
- используем дополнительные тестовые примеры, уделяя основное внимание функциям, на которые с наибольшей вероятностью повлияли изменения.

Хочу обратить ваше внимание на то, что регрессионное тестирование не всегда проводится только методом «черного ящика»; для регресса также используется метод «белого ящика», особенно при поиске функций, на которые с большой вероятностью повлияли изменения.

- **8. Beta-тестирование.**

Это тестирование также проводится методом «черного ящика». Практически готовое ПО отдают для «обкатки» желающим для выявления максимального количества ошибок еще до того, как оно попадет к конечному пользователю.

Что это дает:

- идентификацию непредвиденных ошибок (так как бета-тестеры используют ПО нестандартно);
- широкий набор окружений для проверки, который трудно обеспечить иными методами (разные операционные системы, разные настройки, разные версии браузеров);
- снижение расходов (так как работа бета-тестеров, как правило, не оплачивается).

Техники тестирования «черным ящиком»

- **1. Эквивалентное разбиение.**
  Эта техника включает в себя разделение входных значений на допустимые и недопустимые разделы и выбор репрезентативных значений из каждого раздела в качестве тестовых данных. Она может быть использована для уменьшения количества тестовых случаев. Допустим, у нас есть целая переменная N в диапазоне от -99 до 99: позитивными классами эквивалентности будут [-99, -10], [-9, -1], 0, [1, 9], [10, 99], а недействительными (негативными) – <-99, >99, пустое значение, нечисловые строки.

- **2. Анализ граничных значений.**
  Техника, которая включает в себя определение границ входных значений и выбор в качестве тестовых данных значений, находящихся на границах, внутри и вне границ. Многие системы имеют тенденцию вести себя некорректно при граничных значениях, поэтому оценка значений границ приложения очень важна. При проверке мы берем следующие величины: минимум, (минимум-1), максимум, (максимум+1), стандартные значения. Например, в том же случае -99 <= N <= 99 будет использоваться набор: -100, -99, -98, -10, -9 -1, 0, 1, 9, 10, 98, 99, 100.

- **Тестирование таблицы переходов.**
  При данной технике сценарии тестирования выбираются на основе выполнения корректных и некорректных переходов состояний. Допустим, мы хотим записаться на прием к врачу и зарезервировать время своего приема: заходим в форму, выбираем удобное для нас время и нажимаем кнопку «Записаться». Сразу после этого выбранное нами время становится недоступно для другой записи, так как первая запись привела к изменению в базе.
- **4. Тестирование по сценариям использования.**
  Эта техника используется при написании тестов для индивидуального сценария пользователя с целью проверки его работы.

Достоинства метода

- Тестирование методом «черного ящика» позволяет найти ошибки, которые невозможно обнаружить методом «белого ящика». Простейший пример: разработчик забыл добавить какую-то функциональность. С точки зрения кода все работает идеально, но с точки зрения спецификации это – сверхкритичный баг.
- «Черный ящик» позволяет быстро выявить ошибки в функциональных спецификациях (в них описаны не только входные значения, но и то, что мы должны в итоге получить). Если полученный при тестировании результат отличается от заявленного в спецификации, то у нас появляется повод для общения с аналитиком для уточнения конечного результата.

- Тестировщику не нужна дополнительная квалификация. Часто мы пользуемся различными сервисами и приложениями, не очень в них разбираясь. Для того, чтобы открыть инстаграм и обработать свою фотографию, нам совсем не нужно знать способ реализации фильтров. Мы хотим открыть фотографию, выбрать фильтр и получить красивую картинку на выходе. Задача тестировщика, который тестирует эту функцию в инстаграм, – убедиться, что пользователь получит эту самую красивую картинку в соответствии с выбранным фильтром. При этом нам совсем не обязательно иметь какую-либо специализацию – нужны лишь телефон и инстаграм.
- Тестирование проходит «с позиции пользователя». Пользователь всегда прав, он конечный потребитель практически любого ПО, а значит, ему должно быть удобно, комфортно и понятно.
- Составлять тест-кейсы можно сразу после подготовки спецификации. Это значительно сокращает время на тестирование: к тому моменту, как продукт готов к тестированию, тест-кейсы уже разработаны, и тестировщик может сразу приступать к проверке.

Недостатки метода

- Основным недостатком метода «черного ящика» является возможность пропуска границ и переходов, которые не очевидны из спецификации, но есть в реализации кода (собственно, это и заставляет тестировщиков использовать метод «белого ящика»). Вспоминается случай, когда система получала котировки валют с биржи Forex и округляла до 3 знаков после запятой. Система успешно прошла тестирование методом «черного ящика» (так как ни одна валюта не выходила за соответствующие границы) и хорошо работала до тех пор, пока курс доллара к биткоин не вышел за границы 1000 долларов. Тестирование «белым ящиком» выявило бы ошибку: специалист увидел бы, что коэффициент конверсии валюты ограничен 3 знаками.
- Можно протестировать только небольшое количество возможных вводных (входящих) значений; многие варианты остаются без проверки.
- Тесты могут быть избыточными, если разработчик уже проверил данную функциональность (например, Unit-тестом).
- При отсутствии четкой и полной спецификации проектировать тесты и тест-сценарии оказывается затруднительно.

Подведем итоги

Из представленной информации мы можем сделать следующий вывод: метод «черного ящика» является эффективным при различных видах тестирования; но следует помнить, что некоторые ошибки невозможно найти, используя только этот метод (например, ошибки во внутренней структуре кода).

Проведение «black-box» тестирования увеличивает уверенность в том, что приложение надежно работает на широком диапазоне входных данных, так как набор тестовых данных зависит только от спецификации, а не от особенностей внутренней реализации продукта (как в случае применения методов «белого» и «серого» ящиков).

Метод «черного ящика» выгодно применять, если вы ищете:

- неправильно реализованные функции приложения или сервиса;
- ошибки в пользовательском интерфейсе;
- ошибки в функциональных спецификациях.

Для реализации наиболее полной проверки я рекомендую использовать методы «черного» и «белого» ящиков одновременно. Это позволит увеличить покрытие возможных сценариев, снизить риск пропуска ошибки, а также качественно улучшить результаты тестирования, так как приложение или сервис будет проверено двумя разными методами – с позиций пользователя и внутреннего устройства системы.