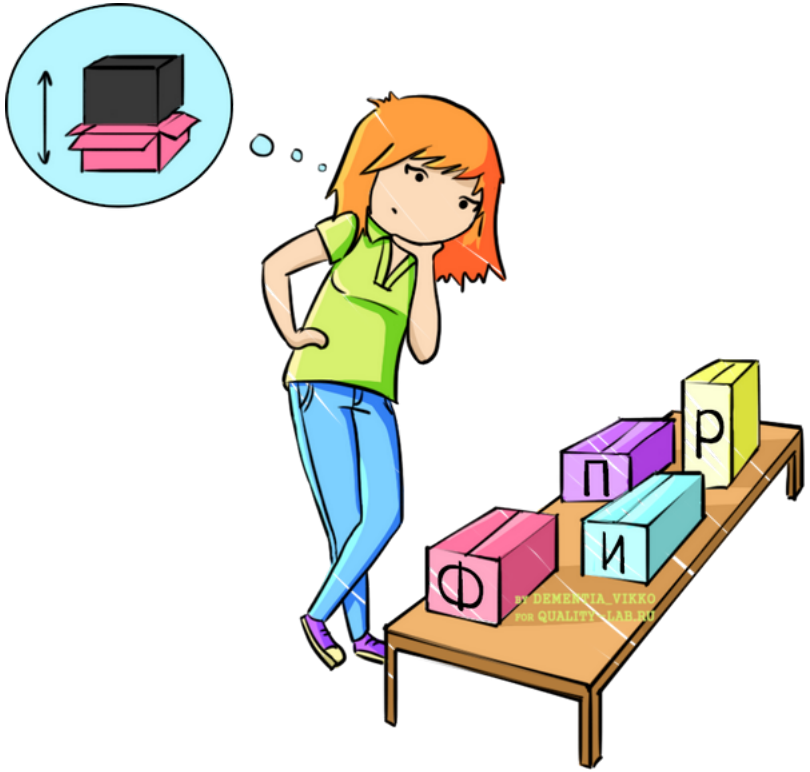
Так называемое «black-box тестирование» является методом тестирования программного обеспечения, внутренняя структура, дизайн и реализация которого неизвестна тестировщику (при подготовке тест-кейсов он опирается на требования и спецификацию). Хочу обратить внимание на то, что требования и спецификация не всегда существуют в письменном виде; тем не менее, при тестировании методом черного ящика мы можем опираться на устно описанные требования.

Что такое «черный ящик» согласно терминологии ISTQB?

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.