Технология разработки программного обеспечения 2015-2016 гг.

Лекции 5-6: тестирование ПО

Пудов Сергей Григорьевич

Немного истории

В 1960-х много внимания уделялось «исчерпывающему» тестированию, которое должно проводиться с использованием всех путей в коде или всех возможных входных данных. Было отмечено, что в этих условиях полное тестирование программного обеспечения невозможно, потому что, во-первых, количество возможных входных данных очень велико, во-вторых, существует множество путей, в-третьих, сложно найти проблемы в архитектуре и спецификациях. По этим причинам «исчерпывающее» тестирование было отклонено и признано теоретически невозможным.

Немного истории

В начале 1970-х годов тестирование программного обеспечения обозначалось как «процесс, направленный на демонстрацию корректности продукта» или как «деятельность по подтверждению правильности работы программного обеспечения».

Во второй половине 1970-х тестирование представлялось как выполнение программы с намерением найти ошибки, а не доказать, что она работает. Успешный тест — это тест, который обнаруживает ранее неизвестные проблемы. Данный подход прямо противоположен предыдущему. Указанные два определения представляют собой «парадокс тестирования», в основе которого лежат два противоположных утверждения: с одной стороны, тестирование позволяет убедиться, что продукт работает хорошо, а с другой — выявляет ошибки в программах, показывая, что продукт не работает.

Немного истории

В 1980-е годы тестирование расширилось таким понятием, как предупреждение дефектов. Проектирование тестов — наиболее эффективный из известных методов предупреждения ошибок. В это же время стали высказываться мысли, что необходима методология тестирования, в частности, что тестирование должно включать проверки на всем протяжении цикла разработки, и это должен быть управляемый процесс. В ходе тестирования надо проверить не только собранную программу, но и требования, код, архитектуру, сами тесты.

«Традиционное» тестирование, существовавшее до начала 1980-х, относилось только к скомпилированной, готовой системе (сейчас это обычно называется системное тестирование), но в дальнейшем тестировщики стали вовлекаться во все аспекты жизненного цикла разработки. Это позволяло раньше находить проблемы в требованиях и архитектуре и тем самым сокращать сроки и бюджет разработки. В середине 1980-х появились первые инструменты для автоматизированного тестирования. Предполагалось, что компьютер сможет выполнить больше тестов, чем человек, и сделает это более надёжно. Поначалу эти инструменты были крайне простыми и не имели возможности написания сценариев на скриптовых языках.

Немного истории

В начале 1990-х годов в понятие «тестирование» стали включать планирование, проектирование, создание, поддержку и выполнение тестов и тестовых окружений, и это означало переход от тестирования к обеспечению качества, охватывающего весь цикл разработки программного обеспечения. В это время начинают появляться различные программные инструменты для поддержки процесса тестирования: более продвинутые среды для автоматизации с возможностью создания скриптов и генерации отчетов, системы управления тестами, ПО для проведения нагрузочного тестирования.

Пример разработки тестов

Спецификация программы

На вход программа принимает два параметра: x - число, n – степень. Результат вычисления выводится на консоль.

Значения числа и степени должны быть целыми.

Значения числа, возводимого в степень, должны лежать в диапазоне – [0..999].

Значения степени должны лежать в диапазоне – [1..10].

Если числа, подаваемые на вход, лежат за пределами указанных диапазонов, то должно выдаваться сообщение об ошибке.

Пример разработки тестов

Определим области эквивалентности входных параметров.

Для x – числа, возводимого в степень, определим классы возможных значений:

1. x < 0 (ошибочное)
2. x > 999 (ошибочное)
3. x - не число (ошибочное)
4. 0 <= x <= 999 (корректное)Для n – степени числа:
5. n < 1 (ошибочное)
6. n > 100 (ошибочное)
7. n - не число (ошибочное)
8. 1 <= n <= 100 (корректное)

Пример разработки тестов

Анализ тестовых случаев

- Входные значения: (x = 2, n = 3) (покрывают классы 4, 8).Ожидаемый результат: The power n of x is 8.
- Входные значения: {(x = -1, n = 2),(x = 1000, n = 5)} (покрывают классы 1, 2).Ожидаемый результат: Error : x must be in [0..999].
- Входные значения: {(x = 100, n = 0),(x = 100, n = 200)} (покрывают классы 5,6).Ожидаемый результат: Error : n must be in [1..10].
- Входные значения: (x = *ADS* n = ASD) (покрывают классы эквивалентности 3, 7).Ожидаемый результат: Error : Please enter a numeric argument.
- Проверка на граничные значения:
  - Входные значения: (x = 999, n = 1).Ожидаемый результат: The power n of x is 999.
  - Входные значения: (x = 0, n = 100).Ожидаемый результат: The power n of x is 0.

Виды и направления тестирования

Классификация по запуску кода на исполнение

- **Статическое тестирование** – тестирование без запуска кода на исполнение.
  - Документы
  - Графические прототипы
  - Код приложения
  - Тестовые данные, параметры среды
- **Динамическое тестирование** – тестирование с запуском кода на исполнение. Запускаться может:
  - Все приложение
  - Несколько взаимосвязанных частей
  - Отдельные части (модульное, компонентное тестирование)

Виды и направления тестирования

Классификация по доступу к коду

- Метод **белого ящика** – у тестировщика есть доступ к внутренней структуре и коду приложения
- Метод **черного ящика** – у тестировщика либо нет доступа, либо недостаточно знаний для понимания, либо так задумано. Другими словами – тестирование на основе документации.
- Метод **серого ящика** – комбинация методов белого и черного ящиков.

Виды и направления тестирования

Классификация по степени автоматизации

- Ручное тестирование
- Автоматизированное тестирование

Виды и направления тестирования

Классификация по уровню детализации приложения

- **Модульное** тестирование (unit testing)
- **Интеграционное** тестирование – или тестирование взаимодействия частей
- **Системное** тестирование – проверка приложения как единого целого

Виды и направления тестирования

Классификация по степени важности

- Дымовое тестирование (**smoke test**) – проверка работоспособности ключевой функциональности
- Тестирование **критического пути** – функционал, используемый типичными пользователями
- Расширенное тестирование

Виды и направления тестирования

Классификация по принципам работы с приложением

- **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).
- **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).

Виды и направления тестирования

Классификация по привлечению конечных пользователей

- **Альфа-тестирование** (alpha testing) выполняется внутри организации-разработчика с возможным частичным привлечением конечных пользователей. Может являться формой внутреннего приёмочного тестирования
- **Бета-тестирование** (beta testing) выполняется вне организации-разработчика с активным привлечением конечных пользователей/заказчиков
- **Гамма-тестирование** (gamma testing) — финальная стадия тестирования перед вы пуском продукта, направленная на исправление незначительных дефектов, обнаруженных в бета-тестировании

Виды и направления тестирования

Нагрузочное тестирование или тестирование производительности

Тестирование удобства пользования

Тестирование безопасности

Форма входа в систему имеет 2 поля - имя и пароль.

SELECT Username FROM Users WHERE Name = 'tester' AND Password = 'testpass';

Введем вместо пароля: testpass' OR '1'='1

SELECT Username FROM Users WHERE Name = 'tester' AND Password = 'testpass' OR '1'='1';

Литература

1. https://ru.m.wikipedia.org/wiki/Тестирование_программного_обеспечения
2. Основы тестирования программного обеспечения http://www.intuit.ru/studies/courses/48/48/info