Верификация ПО

Введение. Назначение тестирования

К.А.Кулаков

Петрозаводск — 2017

Литература

- В.П. Котляров Основы тестирования программного обеспечения http://www.intuit.ru/department/se/testing/
- С. Канер, Д.Фолк Тестирование ПО
- Э. Дастин, Д. Рэшка, Д. Пол "Автоматизированное тестирование программного обеспечения"
- Р. Калбертсон, К. Браун, Г. Кобб "Быстрое тестирование"
- Д. Макгрегор, Д. Сайкс "Тестирование объектно-ориентированного программного обеспечения"
- Л. Тамре "Введение в тестирование программного обеспечения"
- Р. Савин "Тестирование Дот Ком, или пособие по жесткому обращению с багами в интернет-стартапах"
- Э. Хант, Д. Томас "Программист-прагматик. Путь от подмастерья к мастеру"

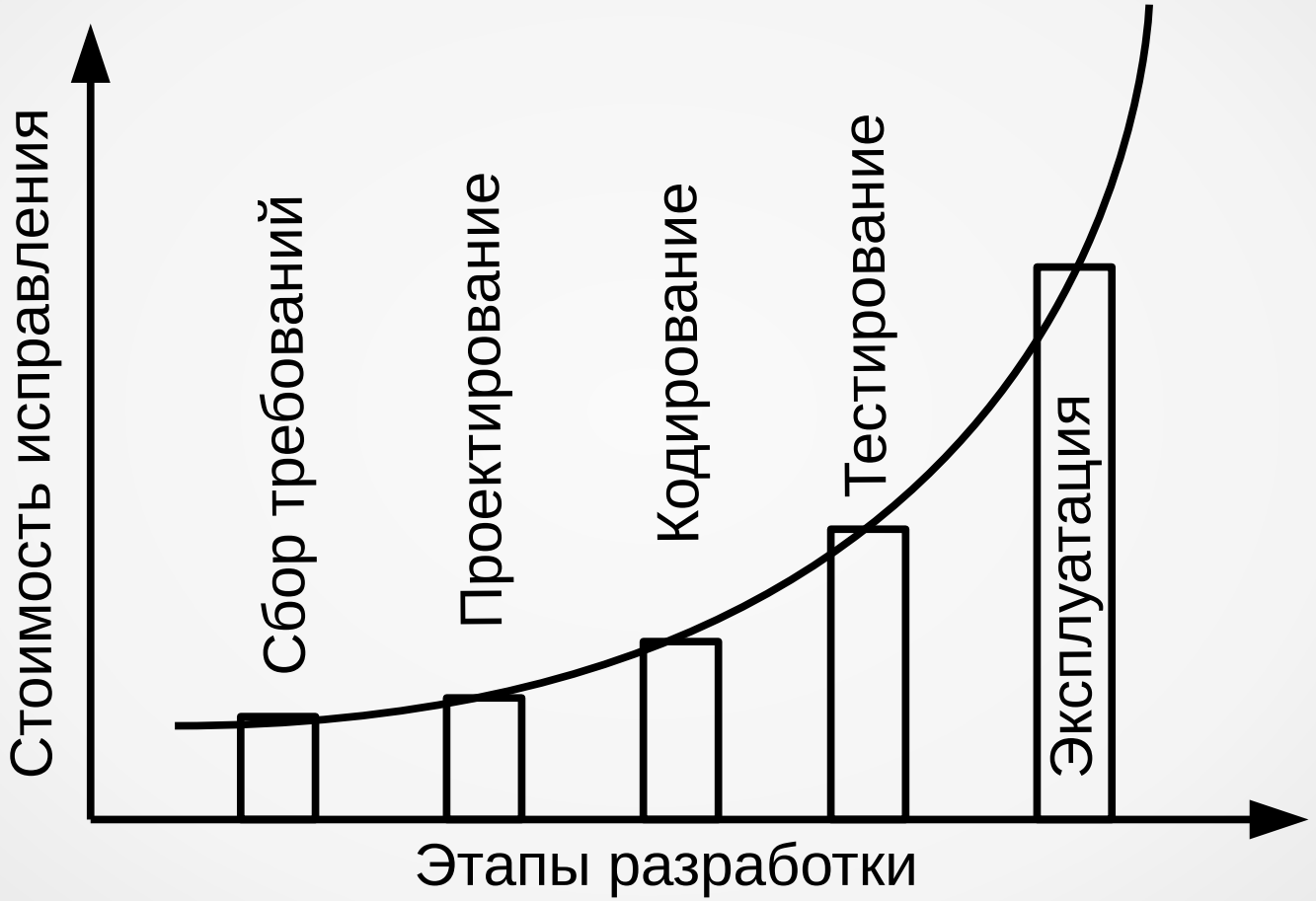
Практика

- http://cs.petrsu.ru/~kulakov/courses/testing
- Отчет
  - Выбор и согласование объекта тестирования
  - Разработка плана тестирования
  - Тестирование (инспекция) проектной документации
  - Статическая верификация
  - Реализация блочных тестов, запуск
  - Реализация интеграционных тестов, запуск
  - Реализация аттестационных тестов, запуск
  - Анализ результатов тестирования
- 3 выступления
  - Обзор объекта тестирования
  - Обзор плана тестирования
  - Примеры разработанных тестов

Качество ПО

- Качество — набор свойств которые определяют насколько продукт хорош.
- У каждого участника проекта различное представление качества
- Задача обеспечения качества:
  - определение заинтересованных лиц
  - определение критериев качества заинтересованных лиц
  - нахождения оптимального решения, удовлетворяющего этим критериям
- Тестирование — способ обеспечения качества

Стоимость исправления ошибки



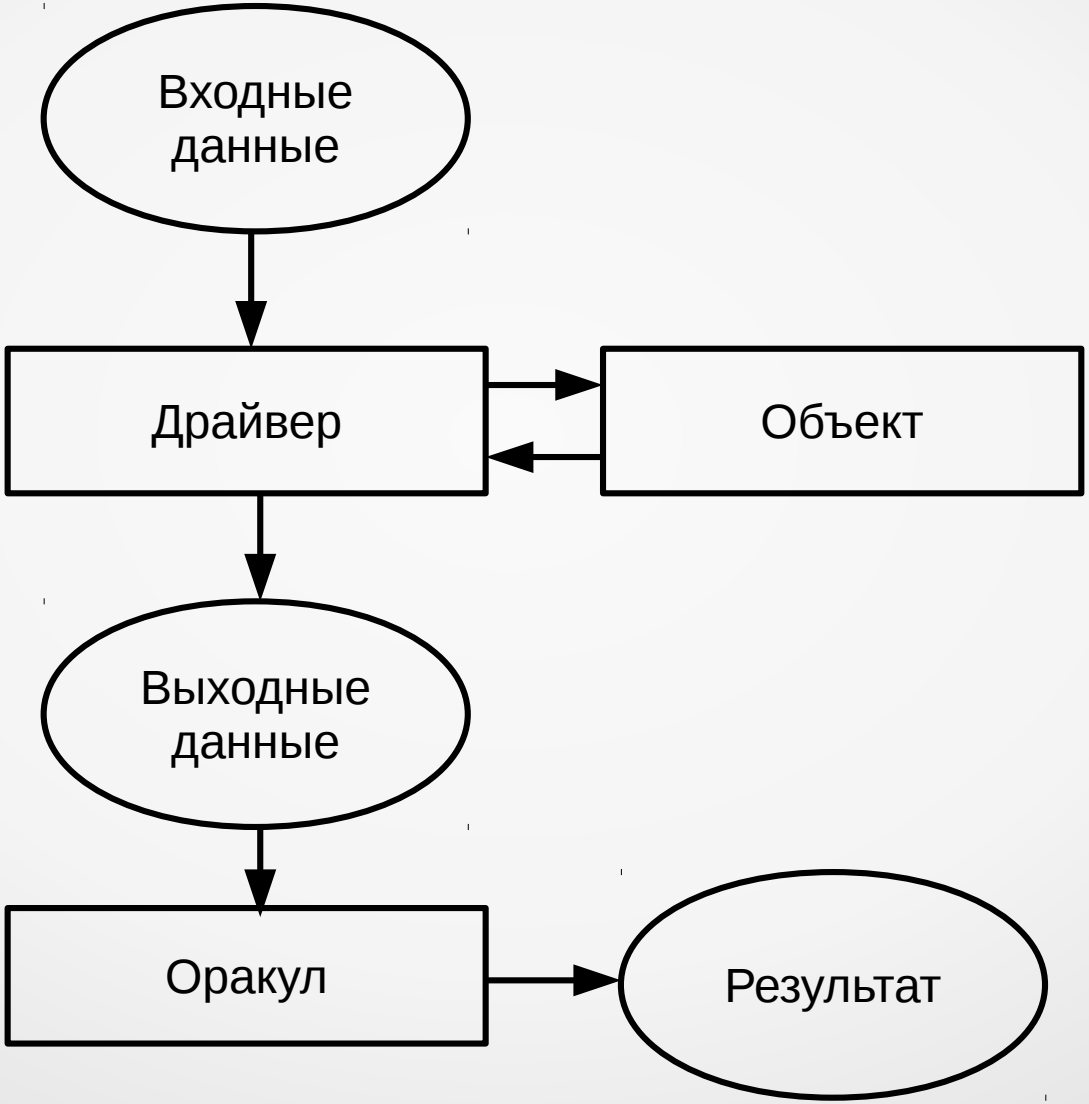
Тестирование — это:

- Процесс выполнения ПО системы или компонента в условиях анализа или записи получаемых результатов с целью проверки (оценки) некоторых свойств тестируемого объекта.
- Процесс анализа пункта требований к ПО с целью фиксации различий между существующим состоянием ПО и требуемым (что свидетельствует о проявлении ошибки) при экспериментальной проверке соответствующего пункта требований.
- Контролируемое выполнение программы на конечном множестве тестовых данных и анализ результатов этого выполнения для поиска ошибок

Терминология

- **Тестирование** обеспечивает выявление (констатацию наличия) фактов расхождений с требованиями (ошибок)
  - **Статическое тестирование** выявляет формальными методами анализа без выполнения тестируемой программы неверные конструкции или неверные отношения объектов программы (ошибки формального задания) с помощью специальных инструментов контроля кода
  - **Динамическое тестирование** (собственно тестирование) осуществляет выявление ошибок только на выполняющейся программе с помощью специальных инструментов автоматизации тестирования
- **Отладка** (debug, debugging) – процесс поиска, локализации и исправления ошибок в программе

Схема тестирования



Отладка

- "Выполнение программы в уме"(deskchecking).
- Вставка операторов протоколирования (печати) промежуточных результатов (logging).
- Пошаговое выполнение программы (single-step running).
- Выполнение с заказанными остановками (breakpoints), анализом трасс (traces) или состояний памяти - дампов (dump).
- Реверсивное (обратное) выполнение (reversible execution).

Этапы тестирования

1) Создание тестового набора (test suite) путем ручной разработки или автоматической генерации для конкретной среды тестирования (testing environment)
2) Прогон программы на тестах, управляемый тестовым монитором (test monitor, test driver) с получением протокола результатов тестирования (test log).
3) Оценка результатов выполнения программы на наборе тестов с целью принятия решения о продолжении или остановке тестирования.

Проблемы тестирования

- Тестирование программы на всех входных значениях невозможно.
- Невозможно тестирование и на всех путях.
- Надо отбирать конечный набор тестов, позволяющий проверить программу на основе наших интуитивных представлений
- В теории алгоритмов доказано, что не существует общего метода для решения этого вопроса, а также вопроса, достигнет ли программа на данном тесте заранее фиксированного оператора.

- **Основная проблема тестирования** — определение достаточности множества тестов для истинности вывода о правильности реализации программы, а также нахождения множества тестов, обладающего этим свойством.

Пример (Задача)

- Входные данные:
  - Назначение ПО: сложить 2 введенных числа.
  - Описание: В каждом числе могут быть 1 или 2 цифры. ПО отображает вводимые числа и затем отображает сумму. Ввод каждого числа заканчивается клавишей Enter
  - Запуск: команда ADDER.

Пример (Шаг 1)

- Проверка работоспособности: проверяем стабильность ПО для проведения тестирования.
  - Input: ADDER<Enter> Output: ?
  - Input: 2 Output: ?2
  - Input: <Enter> Output:?2\n?
  - Input: 3 Output: ?2\n?3
  - Input: <Enter> Output: ?2\n?3\n5\n\n?

Пример (Шаг 1)

- Найденные ошибки
  - (Проектирование) нет указаний с какой программой работаю
  - (Проектирование) нет инструкции по использованию на экране
  - (Проектирование) нет указаний как остановить ПО
  - (кодирование) сумма выведена в стороне слагаемых

Пример (Шаг 2)

- Составление плана тестирования
  - 99 + 99 = 198 (самые большие значения)
  - -99 + -99 = -198 (в описании не сказано что числа положительные)
  - 99 + -14 = 85 (большое положительное + отрицательное)
  - -38 + 99 = 61
  - 56 + 99 = 155 (влияние второго большого на первое)
  - 9 + 9 = 18 (одноцифровые числа)
  - 0 + 0 = 0 (очень часто в нуле ошибка)
  - 0 + 23 = 23
  - -78 + 0 = -78
- Число всевозможных комбинаций — 39601 без учета других символов.
  - группировка тестов по общим признакам
- Нужно выбирать те тесты, в которых наибольшая вероятность появления ошибки

Пример (Шаг 3)

- Запуск тестов с допустимыми значениями:
  - Тесты с положительными числами и нулем прошли успешно
  - На тестах с отрицательными числами ПО зависает

Пример (Шаг 4)

- Тестирование в режиме свободного полета
  - 100 + 100 => 10 + 10 = 20
  - <Enter> + <Enter> => 10 + 10 = 20
  - 123456 + 0 => 12 + 0 = 12
  - 1,2 + 5 => 1 + 2 = 3; 5
  - A + b => зависание
  - <Ctrl + ...> => все кроме <Ctrl + c> привело к зависанию, последнее — выход

Пример (Шаг 5)

- Итоги
  - Программа работает
  - Ограниченный интерфейс
  - Не работает с отрицательными числами
  - Третий символ и , интерпретируются как <Enter>
  - Нет проверки на допустимость вводимых символов
- Перед следующей итерацией анализ резолюций программиста!
  - Например, программист добавил поддержку чисел от -9 до -1.

Цель тестировщика

- Цели новичков:
  - могут полностью протестировать любую программу;
  - выполнив такое тестирование, могут не сомневаться, что программа работает правильно;
  - задача тестировщика заключается в том, чтобы гарантировать правильность работы программы путем проведения полного тестирования.
- Полностью протестировать ПО — выявить все ошибки.
  - Исправления может и не быть
  - Существует популярное убеждение, что программу можно полностью протестировать.

Цель тестировщика

- Почему нельзя выполнить полное тестирование
  - Количество всех возможных комбинаций входных данных слишком велико, чтобы его можно было проверить полностью.
  - Количество всех возможных последовательностей выполнения кода программы также слишком велико, чтобы его можно было проверить полностью.
  - Пользовательский интерфейс программы (включающий все возможные комбинации действий пользователя и его перемещений по программе) обычно слишком сложен для полного тестирования.

Цель тестировщика

- Следует проверить все допустимые входные значения (как?)
- Следует проверить все недопустимые входные значения
- Следует проверить все способы редактирования входных данных
- Следует проверить реакцию программы на ввод в каждый момент ее работы (напр., нажатия клавиш когда она не ждет)

- => Агрегация данных и разбиение на области

Цель тестировщика

- Невозможно выявить все ошибки проектирования
  - Как и все остальные произведения рук человеческих, спецификации часто содержат ошибки. Одни из них случайны (2 + 2 = 5), другие же являются результатами заблуждений проектировщиков.
  - Именно с ошибками проектирования связаны очень многие недостатки пользовательского интерфейса. Но ошибка есть ошибка — и пользователю все равно, на какой стадии разработки она допущена.
  - Поэтому, если программа разработана по неверной спецификации, мы говорим, что она неверна

Цель тестировщика

- Правильность программы нельзя доказать логически
  - Основой работы компьютера является логика.
  - Проанализировав логику программы, можно определить ее состояние в любой точке кода и в любой момент времени.
  - Проблемы времени и количества всевозможных условий.
  - Что именно можно проверить, анализируя программный код? Только соответствие программы спецификации, но никак не правильность самой спецификации.
  - Доказательства — они ведь тоже строятся человеком, а значит, могут содержать ошибки и упущения. Где гарантия, что логика того, кто анализирует программу, точнее логики ее автора?

Цель тестировщика

- Цель тестировщика — проверка правильности программы?
- Тестирование часто определяют как процесс проверки правильности работы программы.
  - Это определение бессмысленно: невозможно так проверить программу, чтобы сделать заключение, что она работает правильно.
  - Кроме того, оно ошибочно: программа не работает абсолютно правильно.
  - Оно заранее предполагает неудачу: если цель тестировщика — показать, что программа работает правильно, то он терпит неудачу каждый раз, когда находит ошибку.
  - Опираясь на него, тестировщик действует неэффективно: если вы заранее настроились на то, что ошибок в программе нет, вероятность их найти значительно уменьшается.

Цель тестировщика

- для чего же тестируют программы?
  - Программу тестируют для того, чтобы найти в ней ошибки. Если тест позволил выявить проблему, значит, он успешный. А тест, не выявивший проблем, был потерей времени.
  - Ошибки ищут для того, чтобы их исправить