Тренинги Intel Delta Course

«Дополнительные главы по Software Engineering»

Несколько “простых” задач по тестированию

Морёнов О., Царева А.

Сложение двух чисел

Задача: проверить функцию сложения двух целых чисел. При этом мы не доверяем CPU (помним про Pentium 4 баг)

```
int sum(int x, int y);
```

Продвинутый вариант – проверьте сложение двух чисел с плавающей точкой

Важно:

- Перебор всех комбинаций входных значений – самый надежный способ
- Практически нереализуем
- Иногда альтернативы нет

Проверка на треугольник

Протестировать функцию, определяющую тип треугольника по длинам 3 отрезков, являющихся сторонами:

-1 : не треугольник

0 : обычный треугольник

1 : равнобедренный треугольник

2 : равносторонний треугольник

```
int is_triangle(int a, int b, int c)
```

Позитивные и негативные тесты

Позитивные тесты

- Тесты, предназначенные для проверки, что программа выполняет свое основное предназначение
- Тесты на основании «правильных» входных данных
- Тестирование с целью проверки соответствий требованиям

Негативные тесты

- Тесты для проверки устойчивости программы к негативным входным данным
- Тесты на проверки устойчивости программы к ошибкам пользователя
- Тесты на то что у программы нет неожиданных побочных эффектов
- Тестирование с целью «сломаем это!»

Следующий день

Протестируйте функцию, которая возвращает дату следующего дня

Date NextDay(Date today);

```
struct Date {
    unsigned char day;
    unsigned char month;
    int year;
};
```

Тестирование Черного Ящика

- Не знаем/Игнорируем устройство тестируемого объекта
- Можем управлять входными параметрами
- Среда, в которой проводим эксперименты, может считаться входным параметром
- Можем измерять выходные параметры

Классы эквивалентности - определение

Большая часть определений «классов эквивалентности» или «эквивалентного разбиения» фокусируются не на том, что это, а на их свойствах

- Определение: Класс эквивалентности (equivalence class) — набор данных, обрабатываемых одинаковым образом и приводящих к одинаковому результату [Lee Copeland, «A practitioner's guide to software test design»]
- Другое определение: Эквивалентный класс включает в себя определенное количество каких-то значений, которые предположительно будут обрабатываться и выдавать на выходе один и тот же результат для всех представителей этого эквивалентного класса.

Сумма чисел

$0 + 1 + 2 + \ldots + N, \quad N \in [0, 100]$

```
read(N);
If (( N< 0) || (N > 100)) {
   error(“Input value out of range”);
}
else
{
   sum=0; x=0;
   while (x < N)
   {
      x=x+1;
      if  (N==42) sum=1; else
        sum=sum+x;
   }
   print(sum);
}
```

Тестирование Белого Ящика

- Используем знание об устройстве тестируемого объекта
- В случае ПО – имеем полный доступ к тестируемому коду

Тестируем функцию

```
int foo (int a, int b, int c) {
   float e;
   if (a == 0) {
      return 0;
   }
   int x = 0;
   if ((a==b) && (c == a) || bug(a) ) {
     x=1;
   }
   e = 1/x;
   return e;
}
```

Тестируем функцию -2

```
float foo (int a, int b, int c) {
  float e;
  if (a == 0) {
     return 0;
  }
  float x = PI;
  if ((a==b) && ((c == a) || bug(a) )) {
    x=1;
  }
  e = 1/x;
  return e;
}
```

Виды покрытия кода

Функциональное (**Function coverage**) – каждая функция вызывается хотя бы раз

Строковое **(Statement coverage) –** каждая строка кода выполнялась хотя бы раз

Решения (**Decision/Branch coverage**) – в каждой условном операторе прошли по всем веткам выбора

Условия (**Condition coverage**) – каждое атомарное булево выражение приняло значения и «истина» и «ложь»

Параметров (**Parameter Value coverage**) – если метод имеет параметры, все значения параметра были использованы

Пути (**Path Coverage**) – все возможные пути в коде были пройдены

Циклы (**Loop coverage**) – Все циклы исполнялись 0,1,..N раз

Тест утилиты копирования

Необходимо протестировать утилиту копирования файлов для ОС Windows. Утилита принимает 2 аргумента:

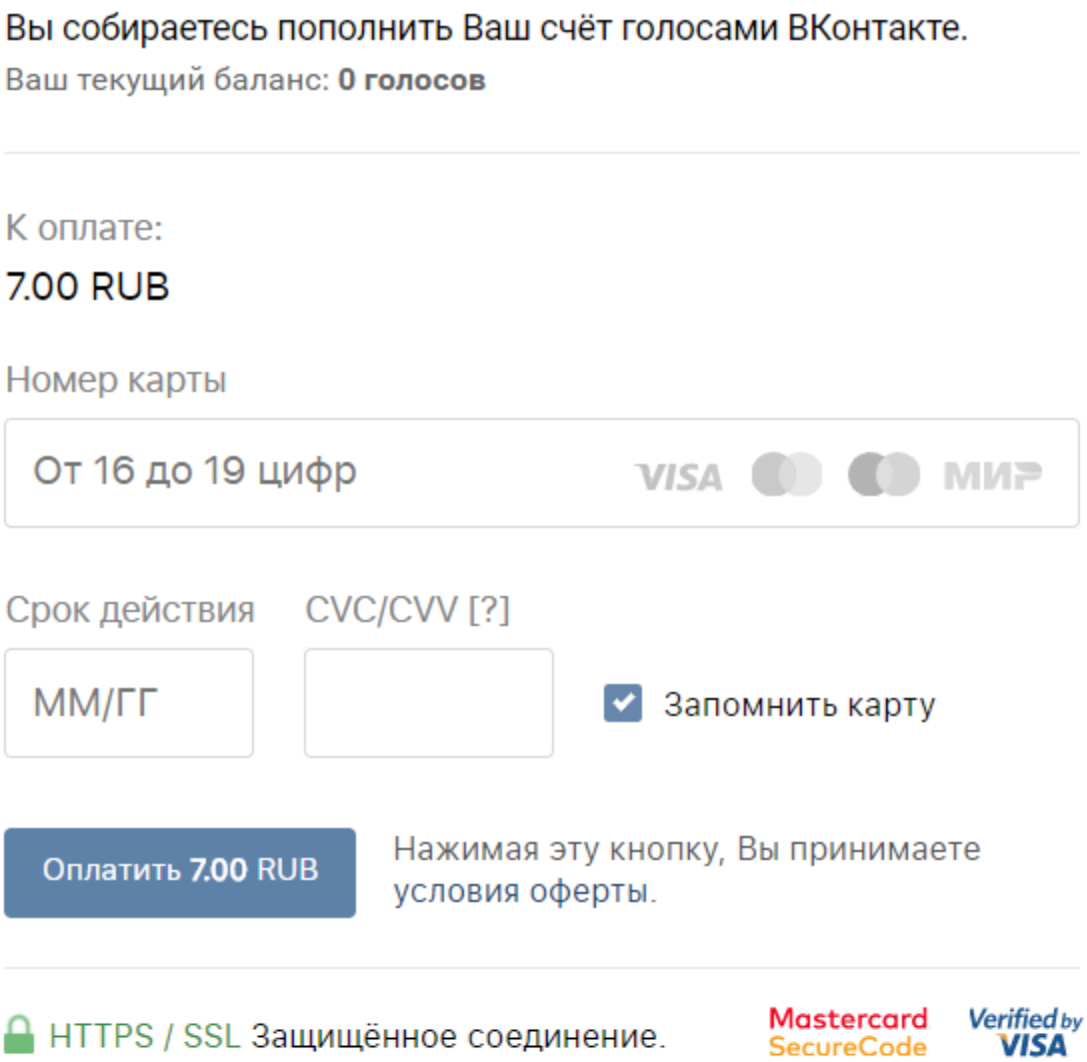
- Путь файла-источника
- Путь до нового файла

my_copy <source_file> <destination_file>

Отличия черного и белого ящиков

Критерий	Черный Ящик	Белый Ящик
Основной уровень применимости	Приемочное тестирование	Юнит-тестирование
Ответственный	Независимый тестировщик	Разработчик
Знание программирования	Не обязательно	Необходимо
Знание реализации	Не обязательно	Необходимо
Знание сценариев использования	Необходимо	Не обязательно
Основа тестовых сценариев	Спецификации	Код

Форма ввода платежной информации



Основы разработки тестовых сценариев

Тест имеет ЦЕЛЬ, МЕТОД, ОКРУЖЕНИЕ

Сценарий: Шаги, оформленные таблицей или списком

- Действия
- **Ожидаемые результаты**

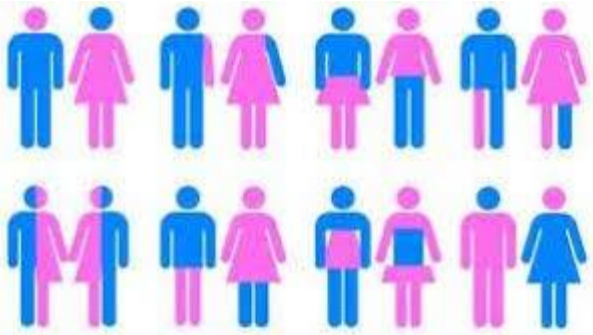
Рекомендации для хорошего теста:

- Существует обоснованная вероятность выявления тестом дефекта
- Определены входные данные
- Определен ожидаемый результат, считаемый «хорошим»
- Воспроизводимость

Распознавание пола по фотографии

Мы сделали систему ИИ, которая по введенной фотографии человека делает классификацию пола.

Как протестировать нашу систему?



Проверка систем обучения

1. Разбиваем всю данные на обучающую выборку/проверочную в заданном соотношении (например 80% / 20%)
2. Используем обучающие выборку для тренировки метода
3. Измеряем качество распознавания на тестовой выборке
4. Повторяем шаги 1-3 заданное число раз (обычно 5-10)
5. Усредняем выбранным методом результат

Не забываем, что качество обучения зависит от качества входных данных. По сути мы тестируем качество интерполяции.

Тестирование камеры

Мы делаем систему удаленного видеонаблюдения для нужд отдела безопасности. Как будем тестировать передачу видео с камеры в программу наблюдения?

- Как подключена камера (провод/радиосигнал) ?
- Какой протокол передачи данных ?
- Насколько он защищен от считывания данных?
- А от подмены данных?
- Что считать качественной картинкой?
- Какое количество кадров/секунду считать приемлемым?
- Надо сохранять данные с камеры? Как долго?
- Сколько камер сможет поддержать система одновременно?
- Сможет ли система работать месяц без сбоев?

Тестирование сервиса-советчика

Мы разработали функцию для онлайн магазина книг. На основании истории покупок он рекомендует новые книги для покупки.

Ваши идеи по тестированию этой функции?