Лекция 5

{ Тема }

ВИДЫ ТЕСТИРОВАНИЯ

Testing types

ПЛАН ЛЕКЦИИ

1. Виды тестирования по знанию об устройстве ПО: тестирование белого ящика, черного ящика
2. Виды тестирования, связанные с изменениями в ПО: дымовое, регрессионное

Black-box testing

Тестирование черного ящика

Методы тестирования черного ящика (black-box testing) основаны на знании о функциональных возможностях разрабатываемой системы. Источником этих знаний могут быть спецификации требований, сценарии использования системы или функциональные спецификации.

Также возможны случаи, когда функции системы нигде не описаны, но понятны тестировщикам:

- из предыдущий проектов
- из документации конкурентов, которые разрабатывают подобный продукт
- исходя из их жизненного опыта
- после обсуждения с заказчиком (в очень! крайнем случае с разработчиком)

White-box testing

Тестирование белого ящика

Методы тестирования белого ящика (white-box testing), или их еще называют структурный тестированием (structural testing), основаны на проверке внутреннего устройства ПО. Они могут применяться на всех уровнях тестирования: компонентном, интеграционном, системном, приемочном.

Техника белого ящика включает в себя следующие **методы определения покрытия**:

- покрытие операторов (statement)
- покрытие решений (decision)
- покрытие условий (condition)
- комбинаторное покрытие условий (multiple condition)

Покрытие – это часть структуры программы, которая была охвачена тестированием, выраженная в процентах.

Для каждого из методов определения покрытия имеется соответствующее название вида тестирования: statement testing, decision testing, condition testing, multiple condition testing.

Statement testing

Тестирование операторов

Тестирование операторов (statement testing) подразумевает, что для 100% покрытия кода необходимо, чтобы каждый оператор программы был выполнен хотя бы один раз.

Например, для 100% покрытия этого кода достаточно одного теста, где side1Length = 1, side2Length = 1, side3Length=1:

```
protected void btnGiveResultClick(object sender, EventArgs e)
  {
     String side1Length = side1.Text;
     String side2Length = side2.Text;
     String side3Length = side3.Text;

     if ((side1Length == side2Length) && (side2Length == side3Length))
     {
        lblResult.Text = "Треугольник - равносторонний!";
     }
  }
```

Но что, если пользователь не введет ничего в поля для сторон треугольника? Или введет разные значения? Или введет буквы?

100% покрытие кода не дает гарантии, что программа полностью протестирована.

Decision testing

Тестирование решений

Во время **тестирования решений** (decision testing) необходимо составить такое число тестов, при котором каждое условие в программе примет как истинное значение, так и ложное.

В следующем примере достаточно 2 тестов для 100% покрытия:
1 –> a = 3, b = 0, x = 4
2 –> a = 3, b = 1, x = 0

```
protected void tstFunc(int a, int b, float x)
  {
     if ((a > 1) && (b == 0))
        x = x / a;
     if ((a == 2) || (x > 1))
        x++;
  }
```

А что, если разработчик ошибся в условии a == 2 (допустим надо было написать a == 5)?

Condition testing

Тестирование условий

Во время **тестирования условий** (condition testing) для 100% покрытия условий необходимо, чтобы все условия принимали и ложное, и истинное значения.

В следующем примере необходимо такое количество тестов, чтобы условия a > 1, b == 0, a == 2, x > 1 принимали как истинное, так и ложное значение:

```
protected void tstFunc(int a, int b, float x)
  {
     if ((a > 1) && (b == 0))
        x = x / a;
     if ((a == 2) || (x > 1))
        x++;
  }
```

То есть достаточно двух тестов:
1 –> a = 2, b = 1, x = 2
2 –> a = 0, b = 0, x = 0

Но при этом строка кода «x = x / a;» не выполнится ни разу, хотя покрытие будет 100%-ым.

Multiple condition testing

Тестирование комбинаций условий

Во время **тестирования комбинаций условий** (multiple condition testing) для 100% необходимо полное покрытие всех условий и всех операторов.

То есть в предыдущем примере добавить еще один тест: a = 3, b = 0, x = -5.

```
protected void tstFunc(int a, int b, float x)
  {
     if ((a > 1) && (b == 0))
        x = x / a;
     if ((a == 2) || (x > 1))
        x++;
  }
```

В итоге получим 3 теста:
1 –> a = 2, b = 1, x = 2
2 –> a = 0, b = 0, x = 0
3 –> a = 3, b = 0, x = -5

Practice

Практическое задание

Подумайте, можно ли достичь 100%-го покрытия в предыдущем примере при multiple condition testing и обойтись только двумя тестами?

```
protected void tstFunc(int a, int b, float x)
   {
      if ((a > 1) && (b == 0))
         x = x / a;
      if ((a == 2) || (x > 1))
         x++;
   }
```

Testing related to changes

Виды тестирования связанные с изменениями

Регрессионное тестирование (regression testing) выполняется, когда в программное обеспечение или в его окружение вносятся изменения.
Глубина регрессионного тестирования оценивается риском пропуска дефектов в программном обеспечении, которое работало ранее.

Разновидности регрессионного тестирования:

- **дымовое тестирование** (smoke testing) – поверхностная проверка того, что основные функции системы работают
- **санитарное тестирование** (sanity testing)– проверка того, что какая-либо функция системы работает как следует
- регрессионное тестирование по дефектам, которые были исправлены в тестируемой версии ПО
- регрессионное тестирование по дефектам, которые были исправлены в предыдущих версиях ПО
- **тестирование побочного эффекта** (side-effect testing) – проверка того, что исправление ошибок в определенной функциональности не повлекло за собой новых ошибок в работе данной функциональности

Regression testing

Регрессионное тестирование

Регрессионные тесты – это первые кандидаты на автоматизацию.

Так как наборы дымовых тестов должны проводиться чаще остальных, то именно они должны быть автоматизированы в первую очередь.

Далее должны быть автоматизированы регрессионные тесты по новой функциональности.

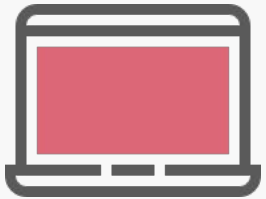
Затем должны автоматизироваться регрессионные тесты тех частей системы, где чаще бывают проблемы.

Регрессионные тесты должны периодически пересматриваться и изменяться, чтобы избежать эффекта пестицида.

Home Task

1. Написать smoke тесты для сайта deveducation.
   Использовать шаблон из предыдущего домашнего задания.

2. Написать smoke тесты для прибора из предыдущего домашнего задания.



dev {education}