LUXOFT TRAINING

Разработка через тестирование Code Coverage

Test Driven Development

Ivan Dyachenko <IDyachenko@luxoft.com>

Module 3

Содержание

LUXOFT Training Center

Code Coverage

LUXOFT Training Center

- Одной из оценок качества системы тестов является полнота – величина той части функциональности системы, которая проверяется тестами
- Полная система позволяет утверждать, что система реализует всю функциональность, указанную в требованиях

Code Coverage

LUXOFT Training Center

- Кроме того, это позволяет утверждать, что система не реализует никакой другой функциональности
- Степень покрытия программного кода тестами – важный количественный показатель, позволяющий оценить качество как системы тестов, так и тестируемой системы

Code Coverage

LUXOFT Training Center

- Одним из наиболее часто используемых методов определения полноты системы тестов является определение отношения количества тест-требований, для которых существуют тесты, к общему количеству тест-требований
- В данном случае речь идет о покрытии тестовами тест-требований

Code Coverage

LUXOFT Training Center

- В качестве единицы измерения степени покрытия здесь выступает процент тест-требований, для которых существуют тесты
- Покрытие требований позволяет оценить степень полноты системы тестов по отношению к функциональности системы, но не позволяет оценить полноту по отношению к ее программной реализации

Code Coverage

LUXOFT Training Center

- Одна и та же функция может быть реализована при помощи совершенно различных алгоритмов, требующих разного подхода к организации тестирования
- Для более детальной оценки полноты системы тестов анализируется покрытие программного кода, называемое также структурным покрытием

Уровни покрытия кода

Существует несколько различных способов измерения покрытия, основные из них:

- покрытие операторов
- покрытие условий
- покрытие путей
- покрытие функций
- покрытие вход/выход

Покрытие операторов

LUXOFT Training Center

- Для обеспечения полного покрытия программного кода на уровне операторов необходимо, чтобы в результате выполнения тестов каждый оператор был выполнен хотя бы один раз
- Перед началом тестирования необходимо выделить переменные, от которых зависит выполнение различных ветвей условий и циклов в коде – управляющие входные переменные

Покрытие операторов

LUXOFT Training Center

```
if (i == 0 || i == 101) {
    if (showMessage) {
        MessageBox.Show("Входной параметр имеет недопустимое значение" +
                        i.ToString());
    } else {
        System.Out.Writeln("Входной параметр имеет недопустимое значение" +
                           i.ToString());
    }
    return -1;
}
```

Покрытие операторов

LUXOFT Training Center

- Для полного покрытия по операторам, достаточно двух тестов:
  - i = 0, showMessage = true
  - i = 0, showMessage = false
- Легко заметить, что при этом, тесты не покрывают всей функциональности (не протестировано поведение системы при i = 101)

Покрытие операторов

LUXOFT Training Center

- Также проблемы этого метода покрытия можно увидеть и на примерах других управляющих структур
- Например, при проверке циклов do … while – при данном уровне покрытия достаточно выполнение цикла только один раз, при этом метод совершенно нечувствителен к логическим операторам || и &&

Покрытие операторов

LUXOFT Training Center

- Другой особенностью данного метода является зависимость уровня покрытия от структуры программного кода
- Рассмотрим простейший пример:

```
if (condition)
    MethodA();
else
    MethodB();
```

Покрытие операторов

LUXOFT Training Center

- Если MethodA() содержит 99 операторов, а MethodB() — один оператор, то единственного теста, устанавливающего condition в true, будет достаточно для достижения 99%-го уровня покрытия
- При этом аналогичный тестовый пример, устанавливающий значение condition в false, даст слишком низкий уровень покрытия (1%)

Покрытие условий

LUXOFT Training Center

- Для обеспечения полного покрытия условий необходимо:
  - каждая точка входа и выхода в программе и во всех ее функциях должна быть выполнена по крайней мере один раз
  - все логические выражения в программе должны принять каждое из возможных значений хотя бы один раз
- Таким образом, для покрытия по веткам требуется как минимум два теста

Покрытие условий

LUXOFT Training Center

```
if (i == 0 || i == 101) {
    if (showMessage) {
        MessageBox.Show("Входной параметр имеет недопустимое значение" +
                        i.ToString());
    } else {
        System.Out.Writeln("Входной параметр имеет недопустимое значение" +
                           i.ToString());
    }
    return -1;
}
```

Покрытие условий

LUXOFT Training Center

- Для покрытия предыдущего примера кода по ветвям потребуется уже три теста
- Это связано с тем, что первый условный оператор if имеет неявную ветвь – пустую ветвь else
- Для обеспечения покрытия по ветвям необходимо покрывать и пустые ветви

Покрытие условий

LUXOFT Training Center

- Особенность данного уровня покрытия заключается в том, что на нем могут не учитываться логические выражения, значения которых получаются вызовом методов

- Рассмотрим пример кода:

```
if (condition1 && (condition2 || Method()))
    statement1;
else
    statement2;
```

Покрытие условий

LUXOFT Training Center

- Полное покрытие условий может быть достигнуто при помощи двух тестов:
  - condition1 = true, condition2 = true
  - condition1 = false, condition2 = true/false
- В обоих случаях не происходит вызова метода Method() (хотя покрытие будет полным)
- Для его проверки необходимо добавить еще один тест:
  - condition1 = true, condition2 = false

Покрытие путей

LUXOFT Training Center

- В данном случае считаются все пути, которые выполняются в процессе работы тестируемого метода
- Путь - уникальная последовательность выполнения операторов, с учетом условных операторов
- Метод, содержащий в себе N условий, имеет 2^N путей
- Метод, содержащий цикл, может иметь бесконечное число путей

Покрытие путей

LUXOFT Training Center

- Т.о. в большинстве случаев 100%-е покрытие путей обеспечить невозможно
- Для решения этой проблемы, может быть применен метод покрытия основных (базисных, линейно-независимых) путей
- Основные пути – минимальный набор путей, комбинация которых может обеспечить все возможные пути выполнения метода

Покрытие путей

LUXOFT Training Center

- Число таких путей равно числу уникальных условных операторов, увеличенное на 1
- Рассмотрим следующий пример:

```
if (condition1)
    statement1;
if (condition2)
    statement2;
if (condition3)
    statement3;
```

Покрытие путей

LUXOFT Training Center

- Для достижения 100% покрытия основных путей, нам потребуется 4 линейно-независимых пути
- Первый путь выбирается случайно (пусть это будет путь, когда все условные выражения принимают значение true)
- Оставшиеся пути получаются поочередным инвертированием одного из условных выражений первого пути

Покрытие путей

LUXOFT Training Center

- Таким образом, получаем четыре основных пути, которые необходимо покрыть:

	condition1	condition2	condition3
Path 1	true	true	true
Path 2	false	true	true
Path 3	true	false	true
Path 4	true	true	false

Покрытие путей

LUXOFT Training Center

В случае наличия циклов, может использоваться следующий подход:

- Выделяем классы путей (к одному классу можно отнести пути, отличающиеся количеством итераций в конкретном цикле)
- Класс считается покрытым, если покрыт хотя бы один путь из него
- 100% покрытие достигнуто, если покрыты все классы путей

Покрытие функций

LUXOFT Training Center

- Покрытие функций – каждая ли функция тестируемого модуля является выполненной хотя бы один раз
- Является одним из самых простых методов расчета покрытия, и дает довольно общее представление о качестве тестируемого модуля
- С одной стороны, данное покрытие говорит нам о том, что тестами покрыт весь реализованный функционал модуля

Покрытие функций

С другой стороны, оно не гарантирует нам адекватное поведение модуля, поскольку:

- Не проверяется реакция функций на все возможные входные параметры
- Не проверяется реакция системы на все возможные возвращаемые функцией значения

Покрытие вход/выход

- Покрытие вход/выход – все ли возможные варианты вызова функций и возврата из них были выполнены
- На данном уровне обеспечивается тестирование как самих функций (все возможные варианты вызова), так и их взаимодействие в составе модуля (все возможные варианты возврата)

Цели и задачи анализа

- К анализу покрытия программного кода можно приступать только после полного покрытия требований
- Полное покрытие программного кода не гарантирует того, что тесты проверяют все требования к системе
- Целью анализа полноты покрытия кода является выявление участков кода, которые не выполняются при выполнении тестов

Цели и задачи анализа

- В идеальном случае при полном покрытии функциональных требований должно получаться 100% покрытие кода
- Однако на практике такое происходит только в случае очень простого кода
- Причины «недопокрытия» кода могут быть различными

Причины плохого покрытия кода

LUXOFT Training Center

- Недостатки в формировании тестов, основанных на требованиях.
  - Тестовый набор должен быть дополнен недостающими тестами
- Неадекватности в требованиях.
  - Требования должны быть модифицированы, после чего разработаны и выполнены дополнительные тесты, покрывающие новые требования

Причины плохого покрытия кода

- «Мертвый код»
  - этот код должен быть удален, и проведен анализ для оценки эффекта удаления и необходимости перепроверки
- Дезактивируемый код – код, работающий только в определенных конфигурациях окружения

Причины плохого покрытия кода

LUXOFT Training Center

Дезактивируемый код

- Для такого кода должна быть установлена нормальная эксплуатационная среда, в которой он выполняется
- Написаны тесты, покрывающие его
- Написаны тесты, проверяющие, что данный код не может быть преднамеренно выполнен в других конфигурациях

Причины плохого покрытия кода

LUXOFT Training Center

- Избыточные условия
  - пример такого условия – выражение !b || (a && b)
  - при b = false, значение переменной a не имеет значения, т.е. условие избыточно и вторая его часть не будет проверяться
- Защитный код

Защитное программирование

Защитное программирование - это метод организации программного кода таким образом, чтобы при работе системы последствия проявления дефектов в ней не приводили к сбоям, отказам и авариям (проверка входных данных, обработка исключений и т.д.)

Защитное программирование

LUXOFT Training Center

- Например, это может быть ветка default в операторе выбора switch
- Входное условие оператора switch может принимать определенные значения
- Как следствие, ветка default, возможно никогда не будет выполнена

Причины плохого покрытия кода

Также существуют случаи, когда модульное тестирование кода сильно затруднено, либо вообще невозможно:

- генерация случайных чисел
- сложные математические алгоритмы
- параллельные алгоритмы

Результаты анализа

Помните, что тесты пишутся для повышения качества кода и лучшего его понимания, а не для повышения показателей метрик!

Вопросы ?

Разработка через тестирование

IDyachenko@luxoft.com

git clone git://github.com/ivan-dyachenko/Trainings.git

https://github.com/ivan-dyachenko/Trainings