Модульное тестирование[[1]](#footnote-1)

Задача:

1. Создать тестовый набор для каждого класса, каждой нетривиальной функции или метода консольного приложения Get3DModel.exe.
2. Прогнать программы на тестах с получением протокола результатов тестирования.
3. Оценить результаты выполнения программы на наборе тестов с целью принятия решения о продолжении или остановке тестирования.
4. После очередного изменения кода - регрессионное тестирование (обнаружение ошибок в уже протестированных участках исходного кода).

Цель:

1. Выявить локализованных ошибки в реализации алгоритмов.
2. Выявить ошибки кодирования (ошибки работы с условиями, счетчиками циклов, использования локальных переменных и ресурсов)
3. Определить степени готовности системы к переходу на следующий уровень разработки и тестирования.

Обзор:

1. Расположение тестов:

Все тесты будут находиться в проекте Get3DModel.Tests

~~//Все тесты будут находиться в папке UnitTests и являться частью контроля версий.~~

~~//2. Именование проекта с тестами: Get3DModel. Tests~~

1. Тестовые классы.

Именование: [Имя класса]Tests

Каждый тестовый класс тестирует только одну сущность. Класс проекта соответствует одному тестовому классу.

1. Тестовые методы:

Именование: [Тестируемый метод]\_[Сценарий]\_[Ожидаемое поведение].

1. Оформление теста: «arrange-act-assert» (организация – действие – утверждение).

Суть в том, чтобы в модульном тесте чётко определить предусловия (инициализация тестовых данных, предварительные установки), действие (собственно то, что тестируется) и постусловия (что должно быть в результате выполнения действия).

1. Один тест проверяет только одну вещь. Тест является спецификацией метода класса, контрактом: какие входные параметры ожидает этот метод, и что остальные компоненты системы ждут от него на выходе.
2. Спецификация теста

Название класса:

Название тестового случая:

Описание тестового случая:

Начальные условия:

Ожидаемый результат:

1. Замечания по функционалу оформляются в таблицу:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемый функционал | Результат теста,   (уд./неуд.) | Содержание несоответствия, в случае неудовлетворительного результата |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Покрытие:
2. **Тестирование на основе потока управления.**

Элементы, которые должны быть покрыты при прохождении тестов, определяются на основе структурных(a-d) и функциональных(e-f) критериев:

1. **Критерий тестирования команд** (критерий С0) - набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза. Это слабый критерий, он, как правило, используется в больших программных системах, где другие критерии применить невозможно.
2. **Критерий тестирования ветвей** (критерий С1) - набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза. Это достаточно сильный и при этом экономичный критерий, поскольку множество ветвей конечно.
3. **Критерий тестирования путей** (критерий С2) - набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раза. Если программа содержит цикл (в особенности с неявно заданным числом итераций), то число итераций ограничивается константой (часто - 2, или *числом классов* выходных путей).
4. **Критерий покрытия условий** - покрытие всех логических (булевских) условий в программе. Критерии покрытия решений (ветвей - С1) и условий не заменяют друг друга, поэтому на практике используется комбинированный критерий покрытия условий/решений, совмещающий требования по покрытию и решений, и условий.
5. **Критерий покрытие функций программы** - каждая функция программы должна быть вызвана хотя бы один раз.
6. **Критерий покрытия вызовов функций** - каждый вызов каждой функции в программе должен быть осуществлен хотя бы один раз. Критерий покрытия вызовов известен также как критерий покрытия пар вызовов (call pair coverage).



Рис. 1 Виды критериев и их функциональность

Структурное тестирование заключается в следующем:

1. Конструирование УГП.

Управляющий граф программы (УГП) – граф G(V,A), где V(V1,… Vm) – множество вершин (операторов), A(A1,… An) – множество дуг (управлений), соединяющих операторы-вершины. Пути, различающиеся хотя бы числом прохождений цикла – разные пути, поэтому число путей в программе может быть не ограничено.

Управляющий граф программы строится на основе статистического анализа программы.

Путь – последовательность вершин и дуг УГП, в которой любая дуга выходит из вершины Vi и приходит в вершину Vj. Пути, различающиеся хотя бы числом прохождений цикла – разные пути, поэтому число путей в программе может быть не ограничено.

Ветвь – путь (V1, V2, … Vk), где V1 - либо первый, либо условный оператор программы, Vk - либо условный оператор, либо оператор выхода из программы, а все остальные операторы – безусловные. Ветви - линейные участки программы, их конечноe число.

1. Выбор тестовых путей.

Необходимо выбрать конечное множество путей, которое необходимо покрыть тестами.

Подходы к построению тестовых путей:

**Статические методы**. Построение каждого пути посредством постепенного его удлинения за счет добавления дуг, пока не будет достигнута выходная вершина управляющего графа программы.

Недостаток: не учитывается возможная нереализуемость построенных путей тестирования. Кроме того, переход от покрывающего множества путей к полной системе тестов пользователь должен осуществить вручную.

Достоинство: Сравнительно небольшое количество необходимых ресурсов, как при использовании, так и при разработке.

**Динамические методы**. Такие методы предполагают построение полной системы тестов, удовлетворяющих заданному критерию, путем одновременного решения задачи построения покрывающего множества путей и тестовых данных. При этом можно автоматически учитывать реализуемость или нереализуемость ранее рассмотренных путей или их частей. Основной идеей *динамических методов* является подсоединение к начальным реализуемым отрезкам путей дальнейших их частей так, чтобы: 1) не терять при этом реализуемости вновь полученных путей; 2) покрыть требуемые элементы структуры программы.

Недостатки: затраты большого количества ресурсов как при разработке, так и при эксплуатации.

Достоинства: Учитывается возможная нереализуемость *построенных путей* тестирования.

**Методы реализуемых путей**. Данная методика заключается в выделении из *множества* путей подмножества всех реализуемых путей. После чего покрывающее множество путей строится из полученного подмножества реализуемых путей.

Достоинство: Методы реализуемых путей дают самый лучший результат

1. Генерация тестов, соответствующих тестовым путям.
2. **Тестирование на основе потока данных**.

Этот вид тестирования направлен на выявление неинициализированных переменных и избыточные присваивания.

1. Версия 1.0 [↑](#footnote-ref-1)