МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський політехнічний інститут»

Кафедра стратегічного управління

ЗВІТ

з лабораторної роботи №4

з дисципліни «Якість і тестування програмного забезпечення»

Перевірила

Виконав:

ст. гр. КН-27

Харків – 2019

**Тема:** Интеграционное тестирование.

**Цель:** Знакомство с методами интеграционного тестирования на реальном примере программного кода.

**Теоретическая информация:**

Рассматриваемый в лабораторной работе программный код выполнен с помощью объектно-ориентированного подхода. Объектно-ориентированное программное обеспечение является событийно управляемым. Передача управления внутри программы осуществляется не только путем явного указания последовательности обращений одних функций программы к другим, но и путем генерации сообщений различным объектам, разбора сообщений соответствующим обработчиком и передача их объектам, для которых данные сообщения предназначены. Следовательно, происходит переход от модели тестирования на основе описания структуры программы, к модели, описывающей поведение программы. Отрицательным аспектом совершаемого перехода для применения рассмотренных ранее моделей является потеря заданных в явном виде связей между модулями программы.

Поэтому для описания поведение программы следует применить наборы диаграмм языка UML – модели поведения системы (InteractionDiagrams). В эту группу входят диаграммы последовательности (SequenceDiagrams) и взаимодействия (CollaborationDiagrams).

Основной проблемой при интеграционном тестировании является тот факт, что на момент его выполнения отдельные модули (методы и классы) могут отсутствовать или быть в разработке. Поэтому необходимо на их место разработать заглушку (stub), которая имитирует их поведение.

При интеграционном тестировании в случае объектно-ориентированного подхода важным моментом является исследование так называемых Р-путей и ММ-путей. Первый описывает прямой вызов метода класса, а второй – косвенно через событие – рис. 1.

Следовательно, на основе списка Р- и ММ-путей формируются тесты. Каждый тест оформляется в виде – таблица 1. Далее в ручном или автоматическом режиме происходит формирование серии тестов и выполнение каждого из них. Результаты тестирования также заносятся в таблицу 1. Если обнаружена ошибка, она также добавляется в систему отслеживания ошибок (bugtrackingsystem).

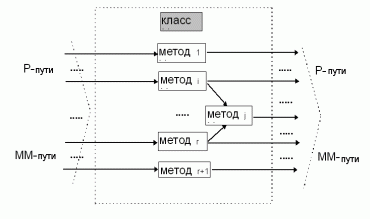


Рисунок 1 – Структура класса с учетом путей тестирования.

Таблица 1 – Пример описания теста в интеграционном тестировании

|  |  |  |  |
| --- | --- | --- | --- |
| **Названия взаимодействующих классов:** | TСommandQueue, Tcommand | **Название теста:** | Тест № 1 |
| **Описание теста:** | тест проверяет возможность создания объекта типа Tcommand и добавления его в очередь при вызове метода AddCommand | | |
| **Начальные условия:** | очередь команд пуста | | |
| **Ожидаемый результат:** | в очередь будет добавлена одна команда | | |
| **Результат теста:** | Тест проведен 10 раз – ошибок не выявлено | | |

Диаграммы действий последовательности и взаимодействия: диаграммы представлены на рисунках 1-3.

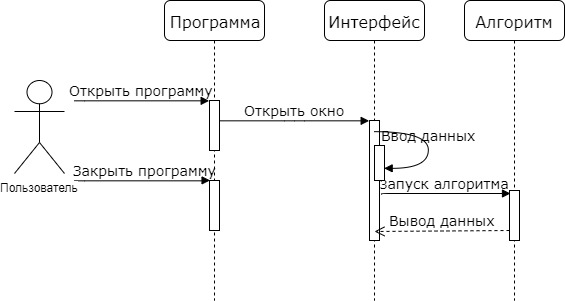


Рисунок 1 – Sequence diagram для программ на Java и C#

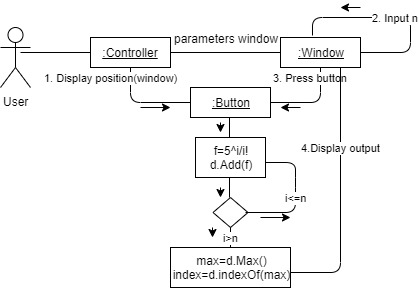


Рисунок 2 – Collaboration diagram для программы на Java

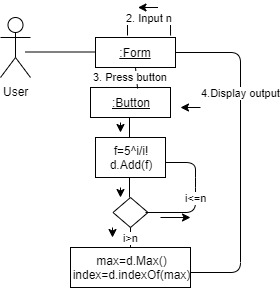


Рисунок 3 – Collaboration diagram для программы на C#

Программный код заглушки на C# и Java:

Java:

**public interface** IProgram {

**public double** onClickMethod();

}

**public class** ProgramInterface **implements** IProgram {

@Override

**Public double** onClickMethod() {

**return** 0;

}

}

C#:

interface IProgram

{

double onClickMethod();

}

class ProgramInterface: IProgram

{

public double onClickMethod()

{

thrownewNotImplementedException();

}

}

Р-путь используется только в случае прямого вызова приложения, которое обрабатывает алгоритм и выгружается из памяти. При этом оно может создать ММ-путь косвенно через событие. В нашем случае мы вызываем приложение на прямую. Т.е. существует 1 Р-путь, но обработчик исключений может перехватить работу программы для обработки ошибки. Таким образом мы имеем 1-Р и 1-ММ путь.

Таблицы результатов тестирования программы:

Таблица результатов для С#:

|  |  |  |  |
| --- | --- | --- | --- |
| **Названия взаимодействующих классов:** | ProgramInterface,UnitTest1 | **Название теста:** | Тест № 1 |
| **Описание теста:** | тест проверяет является ли значение max не равным null. Assert.IsNotNull(max) | | |
| **Начальные условия:** | определение значения max | | |
| **Ожидаемый результат:** | значение max не будет равно null | | |
| **Результат теста:** | Тест проведен 10 раз – ошибок не выявлено | | |
| **Названия взаимодействующих классов:** | ProgramInterface,UnitTest1 | **Название теста:** | Тест № 2 |
| **Описание теста:** | тест проверяет совпадает ли значение метода onClickMethod после его выполнения со значением указанным заранее (actual).Assert.AreEqual(max, 26,041667) | | |
| **Начальные условия:** | Инициализация значения через запуск метода onClickMethod | | |
| **Ожидаемый результат:** | onClickMethod будет равна по значению со значением указанным заранее. Значение результата равно true | | |
| **Результат теста:** | Тест проведен 10 раз – ошибок не выявлено | | |

Таблица результатов для Java:

|  |  |  |  |
| --- | --- | --- | --- |
| **Названия взаимодействующих классов:** | ProgramInterface, TestClass | **Название теста:** | Тест № 1 |
| **Описание теста:** | тест проверяет является ли значение max не равным null. assertNotNull(expectedResult, "error") в противном случаи выводит ошибку - "error" | | |
| **Начальные условия:** | Переменная инициализирована в классеProgramInterface | | |
| **Ожидаемый результат:** | значение max не будет равно null | | |
| **Результат теста:** | Тест проведен 10 раз – ошибок не выявлено | | |
| **Названия взаимодействующих классов:** | ProgramInterface, TestClass | **Название теста:** | Тест № 2 |
| **Описание теста:** | тест проверяет значение, которое будет получено в результате выполнения метода onClickMethod со значением указанным в assertEqual(26.04167, max, 0.001) | | |
| **Начальные условия:** | инициализация переменной, которая будет получена в следствие выполнения метода onClickMethod | | |
| **Ожидаемый результат:** | инициализированная переменная совпадет с ожидаемым результатом | | |
| **Результат теста:** | Тест проведен 10 раз – ошибок не выявлено | | |
| **Названия взаимодействующих классов:** | ProgramInterface, TestClass | **Название теста:** | Тест № 3 |
| **Описание теста:** | тест проверяет значение, которое будет получено в результате выполнения метода onClickMethod в классе Program, со значением указанным в assertTrue(max==26.041(6)8) | | |
| **Начальные условия:** | инициализация переменной, которая будет получена в следствие выполнения метода onClickMethod | | |
| **Ожидаемый результат:** | инициализированная переменная совпадет с ожидаемым результатом | | |
| **Результат теста:** | Тест проведен 10 раз – ошибок не выявлено | | |

**Выводы:** в результате выполнения лабораторной работы, были получены навыки работы с методами и техниками интеграционного тестирования на реальном примере программного кода.