**Практическая работа N 8 Тестирование «белым ящиком».**

*Цель работы*: *изучить метод тестирования «Белым ящиком»*

Сегодня тестирование – это обязательная часть процесса разработки программного обеспечения (далее – ПО). Это связано с жесткими правилами конкуренции для компаний, производящих программные продукты (ПП).

Раньше таких компаний на рынке было мало и пользователи программных продуктов

были продвинутыми и заменяли тестеров. Если в программе обнаруживались баги, то

пользователь звонил или отправлял письмо в компанию, где ошибку исправляли и по почте

отправляли дискетку со свежим релизом. Но начиная с 1990 года согласно статистики продажи

персональных компьютеров с каждым годом удваивались. И появилась армия пользователей,

которая не готова была что-то тестировать. Если что-то не устроило было проще обменять на

другой софт, т.к. число компаний производящих ПО тоже увеличивалось с каждых готом. И у

пользователей появился выбор что покупать и чем пользоваться.

Таким образом, тестирование ушло внутрь компаний, и появилась профессия

тестировщика.

Тестирование ПО – это проверка соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранном определенным образом.

[IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004].

Все виды тестирования можно условно разделить на две большие группы:

Статическое тестирование (static testing).

Динамическое тестирование (dynamic testing).

Статическое тестирование – это процесс анализа самой разработки программного обеспечения, т. е. тестирование без запуска программы.

К данной группе можно отнести анализ кода. Данный вид тестирования осуществляется в основном программистами. Проводят тестирование артефактов разработки программного обеспечения, таких как требования, дизайн или программный код, проводимое без исполнения этих артефактов. Например, с помощью рецензирования или статического анализа.

Статический анализ кода (static code analysis) – это анализ исходного кода, производимый без его исполнения.

Динамическое тестирование – это тестовая деятельность, предусматривающая эксплуатацию (запуск) программного продукта.

Динамическое тестирование предполагает запуск программы, выполнение всех ее функциональных модулей и сравнение фактического ее поведения с ожидаемым.

Статическое тестирование позволяет обнаружить дефекты, которые являются результатом ошибки и привести к сбоям в программном обеспечении. Динамическое тестирование позволяет продемонстрировать непосредственно сбои в программном обеспечении.

Существует несколько признаков, по которым принято производить классификацию видов тестирования.

По знанию системы выделяют:

* тестирование «черного ящика» (black box testing);
* тестирование «белого ящика» (white box testing);
* тестирование «серого ящика» (grey box testing).

Метод белого ящика (white box testing, open box testing, clear box testing, glass box testing) – у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.

Разработка тестов методом белого ящика (white-box test design technique): Процедура разработки или выбора тестовых сценариев на основании анализа внутренней структуры компонента или системы.

Техники, основанные на структуре, или методе белого ящика

* тестирование операторов;
* тестирование альтернатив.

Альтернатива (decision): Точка программы, в которой управление имеет два или более альтернативных путей. Узел с двумя или более связями для разделения ветвей.

Тестирование условий альтернатив (decision condition testing): Разработка тестов методом белого ящика, при котором тестовые сценарии проектируются для исходов условий и результатов альтернатив.

Покрытие (coverage): Уровень, выражаемый в процентах, на который определенный элемент покрытия был проверен набором тестов.

Покрытие альтернатив (decision coverage): Процент результатов альтернативы, который был проверен набором тестов. Стопроцентное покрытие решений подразумевает стопроцентное покрытие ветвей и стопроцентное покрытие операторов.

Покрытие кода (code coverage): Метод анализа, определяющий, какие части программного обеспечения были проверены (покрыты) набором тестов, а какие нет, например, покрытие операторов, покрытие альтернатив или покрытие условий. Еще выделяют серый

ящик.

**Задание 1. Разработать программу на Python.**

Даны длины сторон треугольника, определить вид треугольника и его площадь.

Выполнить контроль вводимых чисел.

1. Разносторонний треугольник

2. Равнобедренный треугольник

3. Равносторонний треугольник

Ограничения:

- три числа не могут быть определены как стороны треугольника;

- если хотя бы одно из них меньше или равно 0;

- сумма двух из них меньше третьего.

**Задание 2. Подготовить набор тестовых вариантов для обнаружения ошибок в программе.**

Результат оформить в следующем виде:

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| А | В | С | Ожидаемый  результат | Объект  проверки |
| Значение | Значение | Значение | Что должно  получится | Значения  вводимых  данных, либо  ожидаемый  результат  ... ... ... ... ... |

**Задание 3. Разработать программу на Python.**

Даны длины сторон треугольника, определить вид треугольника и его площадь.

Выполнить контроль вводимых чисел.

1. Остроугольный треугольник

2. Тупоугольный треугольник

3. Прямоугольный треугольник

Ограничения:

- три числа не могут быть определены как стороны треугольника;

- если хотя бы одно из них меньше или равно 0;

- сумма двух из них меньше третьего.

Подготовить набор тестовых вариантов для обнаружения ошибок в программе и оформить результат.

**Задание 4. На основании проведенных тестов составьте рекомендации по исправлению ошибок, выявленных в ходе тестирования в виде отчета.**

Пример:

1 тест. В ходе проведения первого теста было обнаружено, что при в ведении не корректных данных площадь все равно высчитывается.

Рекомендуется: в случае, если пользователь введет не корректные данные, следует выводить сообщение с просьбой исправить введенные значения. Добавить в программу проверку введенных значений на соответствие ограничения.