МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
**УО «Институт бизнеса Белорусского государственного университета»  
Факультет повышения квалификации и переподготовки**   
1-40 01 73 **Программное обеспечение информационных систем**

**Отчёт к лабораторной работе № \_3\_**

Тема работы «Тестирование приложений методом «Белого ящика»

(наименование темы)

По курсу Тестирование программного обеспечения

(наименование курса)

Выполнили

Слушатель гр. ПВ2-20ПО

(номер группы)

Минтель Анна Вячеславовна

(Фамилия И.О.)

21.03.2022

(Дата сдачи работы)

Проверил

(Фамилия И.О.)

(Дата, оценка и подпись)

Минск 2022

## Лабораторная работа №3

## ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЙ МЕТОДОМ БЕЛОГО ЯЩИКА

### ***Задание***

1. Выполнить тестирование программы SaperTest.
2. Составить отчеты об ошибках.
3. Указать, кто и когда тестировал программный продукт.
4. Составить отчет по результатам лабораторной работы.

**Контрольные вопросы**

***1. В чём заключаются недостатки метода White Box.***

– для выполнения тестирования белого ящика необходимо большое количество специальных знаний;

– при использовании автоматизации тестирования на этом уровне, поддержка тестовых скриптов может оказаться достаточно накладной, если программа часто изменяется.

***2. Что можно тестировать методом белого ящика?***

Тестирование методом белого ящика подразумевает непосредственный доступ к исходному коду и проверка его тестировщиком на правильность реализации кода, а также сравнение фактического результата выполнения программы с ожидаемым.

***3. Перечислите основные виды тестирования, которые можно проводить по методу белого ящика.***

**Методы тестирования на основе стратегии белого ящика:**

**Ввод неверных значений**. При вводе неверных значений тестировщик заставляет коды возврата показывать ошибки и смотрит на реакцию кода. Это хороший способ моделирования определенных событий, например переполнения диска, нехватки памяти и т.д. Популярным методом является замена alloc() функцией, которая возвращает значение NULL в 10% случаев с целью выяснения, сколько сбоев будет в результате. Такой подход еще называют тестированием ошибочных входных данных. При таком тестировании проверяется обработка как верных, так и неверных входных данных. Тестировщики могут выбрать значения, которые проверяют диапазон входных/выходных параметров, а также значения, выходящие за границу диапазона.

**Модульное тестирование**. При создании кода каждого модуля программного продукта проводится модульное тестирование для проверки того, что код работает верно и корректно реализует архитектуру. При модульном тестировании новый код проверяется на соответствие подробному описанию архитектуры; обследуются пути в коде, устанавливается, что экраны, ниспадающие меню и сообщения должным образом отформатированы; проверяются диапазон и тип вводимых данных, а также то, что каждый блок кода, когда нужно, генерирует исключения и возвращает ошибки (еггог returns). Тестирование каждого модуля программного продукта проводится для того, чтобы проверить корректность алгоритмов и логики и то, что программный модуль удовлетворяет предъявляемым требованиям и обеспечивает необходимую функциональность. По итогам модульного тестирования фиксируются ошибки, относящиеся к логике программы, перегрузке и выходу из диапазона, времени работы и утечке памяти.

**Тестирование обработки ошибок**. При использовании этого метода признается, что нереально на практике проверить каждое возможное условие возникновения ошибки. По этой причине программа обработки ошибок может сгладить последствия при возникновении неожиданных ошибок. Тестировщик обязан убедиться в том, что приложение должным образом выдает сообщение об ошибке. Так, приложение, которое сообщает о системной ошибке, возникшей из-за промежуточного программного обеспечения представляет небольшую ценность, как для конечного пользователя, так и для тестировщика.

**Утечка памяти**. При тестировании утечки памяти приложение исследуется с целью обнаружения ситуаций, при которых приложение не освобождает выделенную память, в результате чего снижается производительность или возникает тупиковая ситуация. Данная технология применяется как для тестирования версии приложения, так и для тестирования готового программного продукта. Возможно применение инструментов тестирования. Они могут следить за использованием памяти приложения в течение нескольких часов или даже дней, чтобы проверить, будет ли расти объем используемой памяти. С их помощью можно также выявить те операторы программы, которые не освобождают выделенную память.

**Комплексное тестирование**. Целью комплексного тестирования является проверка того, что каждый модуль программного продукта корректно согласуется с остальными модулями продукта. При комплексном тестировании может использоваться технология обработки сверху вниз и снизу-вверх, при которой каждый модуль, являющийся листом в дереве системы, интегрируется со следующим модулем более низкого или более высокого уровня, пока не будет создано дерево программного продукта. Эта технология тестирования направлена на проверку не только тех параметров, которые передаются между двумя компонентами, но и на проверку глобальных параметров и, в случае объектно-ориентированного приложения, всех классов верхнего уровня.

**Тестирование цепочек**. Тестирование цепочек подразумевает проверку группы модулей, составляющих функцию программного продукта. Эти действия известны еще как модульное тестирование, с его помощью обеспечивается адекватное тестирование компонентов системы. Данное тестирование выявляет, достаточно ли надежно работают модули для того, чтобы образовать единый модуль, и выдает ли модуль программного продукта точные и согласующиеся результаты.

**Исследование покрытия**. При выборе инструмента для исследования покрытия важно, чтобы группа тестирования проанализировала тип покрытия, необходимый для приложения. Исследование покрытия можно провести с помощью различных технологий. Метод покрытия операторов часто называют С1, что также означает покрытие узлов. Эти измерения показывают, был ли проверен каждый исполняемый оператор. Данный метод тестирования обычно использует программу протоколирования (profiler) производительности.

**Покрытие решений**. Метод покрытия решений направлен на определение (в процентном соотношении) всех возможных исходов решений, которые были проверены с помощью комплекта тестовых процедур. Метод покрытия решений иногда относят к покрытию ветвей и называют С2. Он требует: чтобы каждая точка входа и выхода в программе была достигнута хотя бы единожды, чтобы все возможные условия для решений в программе были проверены не менее одного раза и чтобы каждое решение в программе хотя бы единожды было протестировано при использовании всех возможных исходов.

**Покрытие условий**. Покрытие условий похоже на покрытие решений. Оно направлено на проверку точности истинных или ложных результатов каждого логического выражения. Этот метод включает в себя тесты, которые проверяют выражения независимо друг от друга. Результаты этих проверок аналогичны тем, что получают при применении метода покрытия решений, за исключением того, что метод покрытия решений более чувствителен к управляющей логике программы.

***Содержание отчета:***

* Цель работы.
* Итоговый отчет о результатах тестирования web-приложения SaperTest.

***Цель работы***:

Протестировать web-приложение SaperTest. методом «Белого ящика».

***Итоговый отчет о результатах тестирования web-приложения SaperTest***

***ОТЧЁТ ОБ ОШИБКЕ №1***

|  |  |
| --- | --- |
| **Project:** | SaperTest |
| **Component:** | Body |
| **Version:** | v 1.0 |
| **Summary:** | Закомментирован тег body |
| **Severity:** | Minor |
| **Priority:** | Low |
| **Status:** | Open |
| **Author:** | Минтель А.В. |
| **Assigned To:** | Ханько А.В. |
| **Environment:** | Linux (Fedora), laptop HP |
| **Steps to Reproduce:** | Анализ архитектуры html документа |
| **Result:** | Неверная структура html документа |
| **Expected Result:** | Убрать комментарий |
| **Attachment:** | <!--<BODY BGCOLOR="#002040">--> |

***ОТЧЁТ ОБ ОШИБКЕ №2***

|  |  |
| --- | --- |
| **Project:** | SaperTest |
| **Component:** | Body |
| **Version:** | v 1.0 |
| **Summary:** | Теги и свойства css с большой буквы |
| **Severity:** | Minor |
| **Priority:** | Low |
| **Status:** | Open |
| **Author:** | Минтель А.В. |
| **Assigned To:** | Ханько А.В. |
| **Environment:** | Linux (Fedora), laptop HP |
| **Steps to Reproduce:** | Анализ архитектуры html документа |
| **Result:** | Неверная запись тегов и свойств css |
| **Expected Result:** | Привести все теги и свойства css к нижнему регистру |
| **Attachment:** | <BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>,  'CENTER' |

***ОТЧЁТ ОБ ОШИБКЕ №3***

|  |  |
| --- | --- |
| **Project:** | SaperTest |
| **Component:** | Body |
| **Version:** | v 1.0 |
| **Summary:** | Неверная запись свойства css, неверное отображение приложения относительно элемента div |
| **Severity:** | Minor |
| **Priority:** | Low |
| **Status:** | Open |
| **Author:** | Минтель А.В. |
| **Assigned To:** | Ханько А.В. |
| **Environment:** | Linux (Fedora), laptop HP |
| **Steps to Reproduce:** | Анализ архитектуры html документа |
| **Result:** | Неверная запись css стилей |
| **Expected Result:** | Привести в соответствии с общепринятой записью:  <div align-items='center'> |
| **Attachment:** | <div align='CENTER'> |

***ОТЧЁТ ОБ ОШИБКЕ №4***

|  |  |
| --- | --- |
| **Project:** | SaperTest |
| **Component:** | Title |
| **Version:** | v 1.0 |
| **Summary:** | Не отображается название игры |
| **Severity:** | Minor |
| **Priority:** | Low |
| **Status:** | Open |
| **Author:** | Минтель А.В. |
| **Assigned To:** | Ханько А.В. |
| **Environment:** | Linux (Fedora), laptop HP |
| **Steps to Reproduce:** | Анализ тега font |
| **Result:** | Отсутствие отображения, неверная запись тега font |
| **Expected Result:** | Заменить цвет (в соответствии с дизайном )и изменить написание тега:  <font color=’black’>!!!FUNNY GAME!!!</font> |
| **Attachment:** | <FONT COLOR='WHITE'>!!!FUNNY GAME!!! |