**Занятие № 25**

Номер учебной группы: П-16

Фамилия, инициалы учащегося: Хомич В.И.

Дата выполнения работы: 7.12.2022

Тема работы: «Формирование тестов»

**Результат выполнения работы**

Таблица 1 – Тестирование интерфейса

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Выполненные действия | Шаги | Ожидаемый результат |
| 1 | Сворачивание приложения | 1. Запустить программу  2. Нажать кнопку «Свернуть» на устройстве. | 1. Программа запустится  2. Программа свернётся |
| 2 | Открытие свёрнутого приложения | 1. Нажать кнопку «Свернуть» на устройстве.  2. Выбрать программу IBM Editor | 1. Откроется список со свёрнутыми приложениями  2. Программа откроется. |
| 3 | Открытие программы | 1. Запустите программу | 1. Программа запустится |
| 4 | Закрытие программы | 1. Запустите программу  2. Дважды нажать кнопку «Назад» | 1. Программа запустится  2. Программа закроется |
| 5 | Проверка режима нескольких окон | 1. Запустить программу  2. Удержать кнопку «Свернуть» на устройстве | 1. Программа запустится  2. Программа растянется на половину экрана. |
| 6 | Проверка работы режима «В отдельном окне» | 1. Запустить программу  2. Включить режим «В отдельном окне» | 1. Программа запустится  2. Программа перейдёт в режим «В отдельном окне» |

Таблица 2 – Тестирование функционала

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Выполненные действия | Шаги | Ожидаемый результат |
| 7 | Ввод текста русского алфавита | 1. Запустить программу  2. Установить курсор на текстовом поле  3. Ввести текст на русском языке | 1. Программа запустится  2. Текст отобразился, стабильная работа программы |
| 8 | Ввод текста английского алфавита | 1. Запустить программу  2. Установить курсор на текстовом поле  3. Переключиться на английскую раскладку  4. Ввести текст на английском языке | 1. Программа запустится  2. Текст отобразился, стабильная работа программы |
| 9 | Регистрация в программе | 1. Запустить программу  2. Заполнить поля с данными  3. Нажать на кнопку «Регистрация» | 1.Программа запустится  2.Пользователь зарегистрируется |
| 10 | Авторизация в программе | 1. Запустить программу  2. Заполнить поля с данными  3. Нажать на кнопку «Авторизация» | 1.Программа запустится  2. Пользователь авторизуется |
| 11 | Добавление записей | 1. Запустить программу  2. Заполнить поля  3. Нажать на кнопку «Добавить» | 1. Программа запустится  2. Произойдёт добавление записи |
| 12 | Изменение записей | 1. Запустить программу  2. Заполнить поля  3. Нажать на кнопку «Изменить» | 1.Программа запустится  2. Произойдёт изменение записи |
| 13 | Удаление записей | 1. Запустить программу  2. Выбрать поле  3. Нажать на кнопку «Удалить» | 1.Программа запустится  2. Произойдёт удаление записи |
| 14 | Ввод цифр | 1. Запустить программу  2. Ввести в поле цифру | 1.Программа запустится  2. Отобразиться поле с цифрой |
| 15 | Запрет на ввод ненужных символов в поля | 1. Запустить программу  2. Ввести произвольный текст в разные поля | 1.Программа запустится  2.Произойдёт ввод только допустимых символов |
| 16 | Ограничение длинны введённых символов | 1. Запустить программу  2. Найти поля ввода  3. Ввести длинное предложение | 1.Программа запустится  2.В определённый момент текст перестанет вводиться в поле |
| 17 | Поиск по критериям | 1. Запустить программу  2. Ввести произвольный текст в поле поиска и выбрать критерий | 1. Программа запустится  2. Появятся варианты подходящие по критерию |
| 18 | Нажатие на кнопку «О программе» | 1. Запустить программу  2. Нажать на кнопку «О программе» | 1. Программа запустится  2. Появиться окно с дополнительной информацией |
| 19 | Нажатие на кнопку «Cправка» | 1. Запустить программу  2. Нажать на кнопку «Справка» | 1. Программа запустится  2. Откроется файл с описанием программного продукта и инструкцией по пользованию |
| 20 | Нажатие на кнопку «Назад» | 1. Запустить программу  2. Нажать на любой форме кнопку «Назад» | 1. Программа запустится  2. Откроется предыдущее окно |
| 21 | Нажатие на кнопку «Запись» | 1. Запустить программу  2. Нажать на кнопку «Запись» | 1. Программа запустится  2. Появится новое окно с информацией о записях |
| 22 | Нажатие на кнопку «Показания» | 1. Запустить программу  2. Нажать на кнопку «Показания» | 1. Программа запустится  2. Появится новое окно с информацией о показаниях |
| 23 | Нажатие на кнопку «Создать» на форме «Запись» | 1. Запустить программу  2. Зайти на форму «Запись»  3. Заполнить данные в поля  4.Нажать на кнопку «Создать» | 1. Программа запустится  2. Запись появится в таблице и добавится в базу данных |
| 24 | Нажатие на кнопку «Препараты» | 1. Запустить программу  2. Нажать на кнопку «Препараты» | 1. Программа запустится  2. Появится новое окно с информацией о препаратах |
| 25 | Нажатие на кнопку «На лечении» | 1.Запустить программу  3. Нажать на кнопку «На лечении» | 1. Программа запустится  2. Появится новое окно с информацией о пациентах на лечении |
| 26 | Добавление записи на форме «Препараты» | 1. Запустить программу  2. Зайти на форму «Препараты»  3. Заполнить все поля  4. Нажать на кнопку «Добавить» | 1. Программа запустится  2. Запись появится в таблице и добавится в базу данных |
| 27 | Изменение записи на форме «Препараты» | 1. Запустить программу  2. Зайти на форму «Препараты»  3. Заполнить все поля  4.Нажать на кнопку «Изменить» | 1. Программа запустится  2. Запись изменится как в таблице так и в базе данных |
| 28 | Удаление записи на форме «Препараты» | 1. Запустить программу  2. Зайти на форму «Препараты»  3.Выбрать запись  4. Нажать на кнопку «Удалить» | 1. Программа запустится  2. Запись удалиться как с таблицы так и с базы данных |
| 29 | Поиск по критериям на форме «Препараты» | 1. Запустить программу  2. Зайти на форму «Препараты»  3.Написать текст в поле поиска и выбрать критерий | 1. Программа запустится  2. Появится информация по критериям поиска |
| 30 | Ввод цифр в поле ввода ФИО | 1. Запустить программу  2. Ввод в текстовые поля цифр | 1. Программа запустится  2. Введение цифр не произойдёт |

**Контрольные вопросы**

1. Что такое модульное тестирование?

Ответ: Модульное тестирование (Unit Testing) – это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом.

2. Каким по очередности выполняется модульное тестирование?

Ответ: Данный вид тестирование выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность.

3. Перечислите правила планирования тестов.

Ответ: Требования для теста указывают, что будет тестироваться. Они относятся к конкретной цели теста. При подготовке требований придерживайтесь следующих общих правил:

· Требование должно относиться к наблюдаемой и измеряемой величине. Если требование относится к величине, которую нельзя наблюдать или измерить, то невозможно будет определить, выполнено ли требование.

· Требования варианта использования или вспомогательные требования не связаны однозначно с требованиями для теста. Часто варианты использования представляют более одного требования для теста, а вспомогательные требования могут представить одно требование, несколько требований или вообще ни одного (например, требования, относящиеся к маркетингу или созданию пакетов).

Источниками требований для тестов могут быть варианты использования, модели вариантов использования, вспомогательные спецификации, проектные требования, варианты бизнес-процесса, интервью с пользователями и документ архитектуры программного обеспечения. Все эти источники должны быть критическим образом пересмотрены, чтобы сформулировать требования для теста.

4. Что такое интеграционное тестирование?

Ответ: Интеграционное тестирование – это тип тестирования, при котором программные модули объединяются логически и тестируются как группа. Как

правило, программный продукт состоит из нескольких программных модулей, написанных разными программистами.

5. В чем отличие модульного тестирования от интеграционного?

Ответ: Модульное тестирование: это тестирование, фактически выполняемое разработчиками, обладающими знаниями в области программирования. Это тестирование проводится на этапе кодирования и является частью тестирования белого ящика. Когда программное обеспечение поступает на разработку, оно разрабатывается в виде фрагмента кода или фрагментов кода, известных как единое целое.

Функциональное тестирование: это тестирование проводится на этапе тестирования (QA) и является частью тестирования черного ящика. Фактическое выполнение ранее написанных тестовых случаев. Это тестирование фактически проводится тестировщиками, они находят фактический результат любой функциональности на сайте и сравнивают этот результат с ожидаемым результатом. Если они обнаружили какое-либо несоответствие, то это ошибка.

6. Какие методы сборки модулей вы знаете?

Ответ: Тем не менее, если вы придерживаетесь не нативных систем модулей, которые не могут интерпретироваться браузерами, например, CommonJS или AMD (или даже нативный ES6), то вам необходимо использовать специальный инструмент для преобразования модулей в упорядоченный и понятый браузеру код. Вот здесь то и вступают в игру Browserify, RequireJS, Webpack и другие "сборщики модулей" или "загрузчики модулей". В дополнении к сборке и/или загрузке ваших модулей, сборщики модулей предлагают массу дополнительных возможностей, например, авто-перекомпиляция кода при внесении изменений

7. Опишите восходящее тестирование.

Ответ: Восходящее тестирование - это прекрасный способ локализации ошибок. Если ошибка обнаружена при тестировании единственного модуля, то очевидно, что она содержится именно в нем - для поиска ее источника не нужно анализировать код всей системы.

8. Опишите нисходящее тестирование.

Ответ: Нисходящее тестирование (нисходящая разработка). Изолированно тестируется только головной модуль. После завершения тестирования этого модуля с ним соединяются модули, непосредственно вызываемые им, и тестируется полученная комбинация.

9. Перечислите плюсы и минусы восходящего и нисходящих тестирований.

Ответ: Верификация и аттестация системной архитектуры. При нисходящем тестировании больше возможностей выявить ошибки в архитектуре системы на раннем этапе процесса разработки. Обычно это структурные ошибки, раннее выявление которых предполагает их исправление без дополнительных затрат. При восходящем тестировании структура высокого уровня не утверждается вплоть до последнего этапа разработки системы.

Демонстрация системы. При нисходящей разработке незаконченная система вполне пригодна для работы уже на ранних этапах разработки. Этот факт является важным психологическим стимулом использования нисходящей модели разработки систем, поскольку демонстрирует осуществимость

управления системой. Аттестация проводится в начале процесса тестирования путем создания демонстрационной версии системы. Но если система создается из повторно используемых компонентов, то и при восходящей разработке также можно создать ее демонстрационную версию.

Реализация тестов. Нисходящее тестирование сложно реализовать, так как необходимо моделировать программы-заглушки нижних уровней. Программы-заглушки могут быть упрощенными версиями представляемых компонентов. При восходящем тестировании для того, чтобы использовать компоненты нижних уровней, необходимо разработать тестовые драйверы, которые эмулируют окружение компонента в процессе тестирования.

Наблюдение за ходом испытаний. При нисходящем и восходящем тестировании могут возникать проблемы, связанные с наблюдениями за ходом тестирования. В большинстве систем, разрабатываемых сверху вниз, более верхние уровни системы, которые реализованы первыми, не генерируют выходные данные, однако для проверки этих уровней нужны какие-либо выходные результаты. Испытатель должен создать искусственную среду для генерации результатов теста. При восходящем тестировании также может возникнуть необходимость в создании искусственной среды (тестовых драйверов) для исследования компонентов нижних уровней.

10. Что такое системное тестирование?

Ответ: Системное тестирование — это уровень тестирования, который проверяет законченный и полностью интегрированный программный продукт. Целью системного теста является оценка сквозных технических характеристик системы.

11. Какие виды тестов используются при системном тестировании?

Ответ:

• функциональное тестирование;

• тестирование производительности;

• нагрузочное или стрессовое тестирование;

• тестирование конфигурации;

• тестирование безопасности;

• тестирование надежности и восстановления после сбоев;

• тестирование удобства использования

12. Какую последовательность действий, необходимо выполнить для генерации программного кода в среде IBM Rational Rose?

Ответ: Общая последовательность действий, которые необходимо выполнить для генерации программного кода в среде IBM Rational Rose 2003, состоит из следующих этапов: Проверка модели на отсутствие ошибок. Создание компонентов для реализации классов. Отображение классов на компоненты.

13. Как произвести проверку модели в IBM Rational Rose?

Ответ: В общем случае проверка модели может выполняться на любом этапе работы над проектом. Однако после завершения разработки графических диаграмм она является обязательной, поскольку позволяет выявить целый ряд ошибок разработчика. К числу таких ошибок и предупреждений относятся, например, не используемые ассоциации и классы, оставшиеся после удаления

отдельных графических элементов с диаграмм, а также операции, не являющиеся именами сообщений на диаграммах взаимодействия.

Для проверки модели следует выполнить операцию главного меню: Tools \to Check Model (Инструменты \to Проверить модель). Результаты проверки разработанной модели на наличие ошибок отображаются в окне журнала. Прежде чем приступить к генерации текста программного кода разработчику следует добиться устранения всех ошибок и предупреждений, о чем должно свидетельствовать чистое окно журнала

14. В какой последовательности необходимо выполнять операции при выборе языка кодирования?

Ответ: Общая последовательность действий, которые необходимо выполнить для генерации программного кода в среде IBM Rational Rose 2003, состоит из следующих этапов:

· Проверка модели на отсутствие ошибок.

· Создание компонентов для реализации классов.

· Отображение классов на компоненты.

· Выбор языка программирования для генерации текста программного кода.

· Установка свойств генерации программного кода.

· Выбор класса, компонента или пакета.

· Генерация программного кода.

Особенности выполнения каждого из этапов могут изменяться в зависимости от выбора языка программирования или схемы базы данных.

15. Как произвести генерацию программного кода в среде IBM Rational Rose для отдельного класса или компонента?

Ответ: Генерация программного кода в среде IBM Rational Rose 2003 возможна для отдельного класса или компонента. Для этого нужный элемент модели предварительно следует выделить в браузере проекта и выполнить операцию контекстного меню: ANSI C++ \to Generate Code\_ (Язык ANSI C++ \to Генерировать код). В результате этого будет открыто диалоговое окно с предложением выбора классов для генерации программного кода на выбранном языке программирования (рис. 14.6). После выбора соответствующих классов и нажатия кнопки OK программа IBM Rational Rose 2003 выполняет кодогенерацию.