МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Новоуральский технологический институт –
филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»
(НТИ НИЯУ МИФИ)
Колледж НТИ

Цикловая методическая комиссия информационных технологий

Отчет 3
ПО АУДИТОРНОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЕ НА ТЕМУ

«Ознакомиться с основными понятиями и методами процесса тестирования ПО»

МДК 03.02 «Инструментальные средства разработки программного обеспечения»

Специальность СПО 09.02.03
«Программирование в компьютерных системах»

очная форма обучения
на базе основного общего образования

Выполнил
студент группы КПР–47 Д
Егорушкин И.А. 4.11.2020 (дата) (подпись)

Проверил
преподаватель
Лебедева А.Н. (дата) (подпись)

Новоуральск 2020

Цель: Ознакомиться с основными понятиями и методами процесса тестирования ПО

Контрольные вопросы:

1. Перечислить и охарактеризовать основные понятия связанные с процессом тестирования

Определение. Тестирование –это контролируемое выполнение программы на конечном множестве наборов данных и анализ результатов этого выполнения с целью обнаружения ошибок.

Часто тестирование программы в соответствии с этим определением называют динамическим тестированием, а статический анализ, не требующий выполнения программы (просмотр, инспекция), – статическим тестированием.

Принято выделять методы тестирования и критерии тестирования программного продукта.

Определение. Методы тестирования – это совокупность правил, регламентирующих последовательность шагов по тестированию.

Определение. Критерии тестирования – соображения, позволяющие судить о достаточности проведенного тестирования.

Под ошибкой принято понимать различие между вычисленным, обозреваемым или измеренным значением или условием и действительным, специфицированным или теоретически корректным значением или условием, т.е. в программе имеется ошибка, если ее выполнение не оправдывает ожиданий пользователя.

Любой программный продукт – от простейших приложений до сложных комплексов реального времени, – вряд ли можно считать свободным от ошибок.

2. Перечислить способы тестирования

Модульные тесты

Модульные тесты считаются низкоуровневыми, близкими к исходному коду вашего приложения. Они нацелены на тестирование отдельных методов и функций внутри классов, тестирование компонентов и модулей, используемых вашей программой. Модульные тесты в целом не требуют особых затрат на автоматизацию и могут отрабатывать крайне быстро, если задействовать сервер непрерывной интеграции (continuous integration server).

Интеграционные тесты

Интеграционные тесты проверяют хорошо ли работают вместе сервисы и модули, используемые вашим приложением. Например, они могут тестировать интеграцию с базой данных или удостоверяться, что микросервисы правильно взаимодействуют друг с другом. Эти тесты запускаются с бОльшими затратами, поскольку им необходимо, чтобы много частей приложения работало одновременно.

Функциональные тесты

Функциональные тесты основываются на требованиях бизнеса к приложению. Они лишь проверяют выходные данные после произведенного действия и не проверяют промежуточные состояния системы во время воспроизведения действия.

Иногда между интеграционными тестами и функциональными тестами возникают противоречия, т.к. они оба запрашивают множество компонентов, взаимодействующих друг с другом. Разница состоит в том, что интеграционные тесты могут просто удостовериться, что доступ к базе данных имеется, тогда как функциональный тест захочет получить из базы данных определенное значение, чтобы проверить одно из требований к конечному продукту.

Сквозные тесты (End-to-end tests)

Сквозное тестирование имитирует поведение пользователя при взаимодействии с программным обеспечением. Он проверяет насколько точно различные пользователи следуют предполагаемому сценарию работы приложения и могут быть достаточно простыми, допустим, выглядеть как загрузка веб-страницы или вход на сайт или в более сложном случае – подтверждение e-mail адреса, онлайн платежи и т.д.

Сквозные тесты крайне полезные, но производить их затратно, а еще их может быть сложно автоматизировать. Рекомендуется проводить несколько сквозных тестов, но все же полагаться больше на низкоуровневое тестирование (модульные и интеграционные тесты), чтобы иметь возможность быстро распознать серьезные изменения.

Приемочное тестирование

Приемочные тесты – это формальные тесты, которые проводятся, чтобы удостовериться, что система отвечает бизнес-запросам. Они требуют, чтобы приложение запускалось и работало, и имитируют действия пользователя. Приемочное тестирование может пойти дальше и измерить производительность системы и отклонить последние изменения, если конечные цели разработки не были достигнуты.

Тесты производительности

Тесты на производительности проверяют поведение системы, когда она находится под существенной нагрузкой. Эти тесты нефункциональные и могут принимать разную форму, чтобы проверить надежность, стабильность и доступность платформы. Например, это может быть наблюдение за временем отклика при выполнении большого количества запросов или наблюдение за тем, как система ведет себя при взаимодействии с большими данными.

Тесты производительности по своей природе проводить достаточно затратно, но они могут помочь вам понять, какие внешние факторы могут уронить вашу систему.

Дымовое тестирование (Smoke testing)

Дымовые тесты – это базовые тесты, которые проверяют базовый функционал приложения. Они отрабатывают достаточно быстро и их цель дать понять, что основные функции системы работают как надо и не более того. Такое тестирование направлено на выявление явных ошибок.

3. Охарактеризовать систему -Test Management System - TestIT
    - Удобная установка и поддержка.

- Удобный и понятный интерфейс.
- Создание и управление проектами.
- Создание пользователей и проектных ролей для пользователей.
- Удобная интеграция с автоматическими тестами.
- Создание тест-плана.
- Создание тест-кейса.
- Создание чек-листа.
- Создание общего шага
- Версионирование тест-кейса/чек-листа.
- Создание пользовательских атрибутов/конфигураций.
- Прогон тест-кейса/чек-листа.
- Понятная система отчётности.
- Встроенная система баг-трекинга.
- Возможность оповещения коллег внутри и вне системы.
- Возможность интеграции с другими инструментами.

Выполнить практическое задание:

Составить план тестирования для графического редактора

1. Реестровая графика +/-
2. Векторная графика +/-
3. Фрактальная графика +/-
4. Трехмерная графика +/-

Продумать план тестирования для курсового проекта

1. Отправка запросов и правильное их исполнение.

Создать чек-лист для графического редактора

1. Реестровая графика
2. Векторная графика
3. Фрактальная графика
4. Трехмерная графика

Продумать чек-лист для курсового проекта

1. Работа с пользователями
2. Работа с заказами