Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»
Рязанский станкостроительный колледж

Отчёт о практической работе №7
«МДК 05.03»

Выполнила:
Студентка группы ИСП-32
Фролова Е.О.
Проверил:
Родин Е.Н.

Рязань 2025

Цель работы

Изучение понятий чек-лист и тест-кейс и получение навыков их составления.

Краткие теоретические сведения Чек-лист (check-list) – набор идей тестов. **Тест-кейс (test case)** – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства. **Тестовый сценарий, тест-сьют (test scenario, test-suite)** – набор тест-кейсов, собранных в группу (последовательность) для достижения некоторой цели.

Разработка тестов

Существует много подходов к проектированию тестов. Например, можно создавать:

1) Тесты на основе требований (requirements based tests)
2) Функциональные тесты (functional test)
3) Сравнительные («параллельные») тесты (parallel testing)
4) Сценарные тесты (scenario tests)
5) Тесты ошибочных ситуаций (fault injection tests)
6) Тесты интерфейса (interface tests, GUI tests)
7) Тесты удобства использования (usability tests)
8) Тесты упаковки и документации (packaging/documentation tests)
9) Стрессовые тесты (stress tests)
10) Тесты производительности (performance tests)
11) Конфигурационные тесты (configuration tests)
12) Законодательные тесты (regulation tests)

Классы эквивалентности и граничные условия

Класс эквивалентности (equivalence class) – набор тестов, полное выполнение которого является избыточным и не приводит к обнаружению новых дефектов. Признаки эквивалентности (несколько тестов эквивалентны, если): Они направлены на поиск одной и той же ошибки.

1) Если один из тестов обнаруживает ошибку, другие её тоже не обнаружат.
2) Если один из тестов НЕ обнаруживает ошибку, другие её тоже, скорее всего, НЕ обнаружат.
3) Тесты используют схожие наборы входных данных.
4) Для выполнения тестов мы совершаем одни и те же операции.
5) Тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние.
6) Все тесты приводят к срабатыванию одного и того же блока обработки ошибок («error handling block»).
7) Ни один из тестов не приводит к срабатыванию блока обработки ошибок («error handling block»).

Граничные условия (border conditions) – это те места, в которых один класс эквивалентности переходит в другой. Граничные условия очень важны, и их обязательно следует проверять в тестах, т.к. именно в этом месте чаще всего и обнаруживаются ошибки.

Основные понятия

-Чек-лист (Check-list)– это список проверок, которые необходимо выполнить для подтверждения работоспособности функционала. Он не содержит детальных шагов, но помогает систематизировать тестирование.

- Пример: Проверить, что приложение запускается без ошибок.

- Тест-кейс (Test Case) – это последовательность шагов с ожидаемым результатом, направленная на проверку конкретной функции.

- Пример:
  - Шаг 1: Открыть приложение.
  - Шаг 2: Нажать "Файл" → "Открыть".
  - Ожидаемый результат: Изображение загружается и отображается в интерфейсе.

- Тестовый сценарий (Test Scenario) – это набор тест-кейсов, объединённых общей логикой (например, полный процесс обработки изображения).

Подходы к тест-дизайну

В работе использовались следующие методы проектирования тестов:

- Тестирование на основе требований (Requirements-based testing) – проверка соответствия приложения заявленным требованиям.
- Функциональное тестирование (Functional testing) – проверка работы функций (загрузка, сохранение, обработка изображений).
- Тестирование граничных условий (Boundary value analysis) – проверка поведения приложения на крайних значениях (например, загрузка изображения 10 МБ).
- Негативное тестирование (Negative testing) – проверка обработки ошибок (например, попытка загрузить неподдерживаемый формат).

Практическая часть

Анализ требований

Перед составлением тестов я проанализировал требования к приложению и выявил несколько недостатков:

- Не указан конкретный алгоритм выделения границ (Canny, Sobel и др.).
- Не описано, как именно должны комбинироваться границы при выборе нескольких направлений.
- Нет чётких критериев производительности (максимальное время обработки изображения).

Разработка тестовой документации

Чек-лист (Smoke-тест)
Составлен для проверки критического пути – основных функций, без которых приложение не может работать.

№	Проверка	Ожидаемый результат
1	(размытие/контраст)|	Открывается без ошибок
2	Загрузка PNG/JPG/BMP	Изображение отображается
3	Попытка загрузить GIF	Выводится сообщение об ошибке
4	Нажатие кнопки "Фильтр"	Изображение изменяется(размытие/контраст)
5	Выделение границ без направления	Границы выделяются по всему изображению

Тест-кейсы
Разработаны 11 тест-кейсов, охватывающих:
- Загрузку и сохранение файлов (включая конфликты имён).
- Работу фильтров (размытие, контрастность).
- Выделение границ (по направлениям, комбинирование).
- Обработку ошибок (некорректные форматы, сбои при сохранении).

Пример тест-кейса:
Название:Проверка сохранения файла с существующим именем.
Предусловия: В папке уже есть файл `test.jpg`.
Шаги:
1. Открыть изображение в приложении.
2. Нажать "Файл" → "Сохранить".
3. Ввести имя `test.jpg`.
4. Подтвердить сохранение.
Ожидаемый результат: Появляется предупреждение о перезаписи.

Заключение

Работа позволила закрепить навыки тест-дизайна и показала важность детального анализа требований. В дальнейшем можно углубиться в автоматизацию тестирования(например, с использованием Selenium или PyTest) и тестирование производительности (JMeter, Locust).