1. ПРАКТИЧЕСКАЯ РАБОТА. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА МЕТОДОМ «ЧЕРНОГО ЯЩИКА»

1.1. Цель и задачи практической работы

Цель работы заключается в знакомстве студентов с процессом тестирования программного обеспечения, включая подготовку технической документации, выявление ошибок и их документирование. Развитие навыков командной работы при тестировании программного продукта.

Для достижения поставленной цели работы студентам необходимо выполнить ряд задач:

1) научиться разрабатывать программный продукт с учетом требований для дальнейшего тестирования;
2) овладеть методом тестирования «черного ящика»;
3) разработать и протестировать техническое задание и документацию;
4) научиться находить ошибки в чужом программном продукте и документировать их;
5) оформить итоговый отчет по проделанной работе в соответствии с установленными стандартами;
6) освоить работу с инструментами для управления тестированием и отслеживания багов;
7) изучить распределение ролей в команде тестировщиков.

1.2. Теоретический раздел

1.2.1. Методы тестирования

Тестирование программного обеспечения — это неотъемлемая часть жизненного цикла разработки, которая направлена на выявление ошибок, улучшение качества продукта и обеспечение соответствия его функциональности требованиям. Методы тестирования, такие как «черный ящик», «белый ящик» и «серый ящик», появились как результат эволюции подходов к анализу программных систем. Они стали ответом на необходимость проверки сложных продуктов, начиная от первых мейнфреймов и заканчивая современными распределенными приложениями. Каждый из методов отражает определенные аспекты тестирования, от анализа поведения до изучения внутренней структуры системы.

Метод **«черного ящика»** впервые нашел широкое применение в 1950-х годах с развитием первых крупных компьютерных систем, таких как ENIAC и UNIVAC. Тестировщики обнаружили, что функциональные требования системы можно проверять без анализа её внутренней структуры. Концепция была формализована в 1970-х годах специалистами по качеству программного обеспечения. Одним из них был Борис Бейзер, который позже написал книгу «Software Testing Techniques» (рис. 1), ставшей на сегодняшний день классикой в области тестирования.

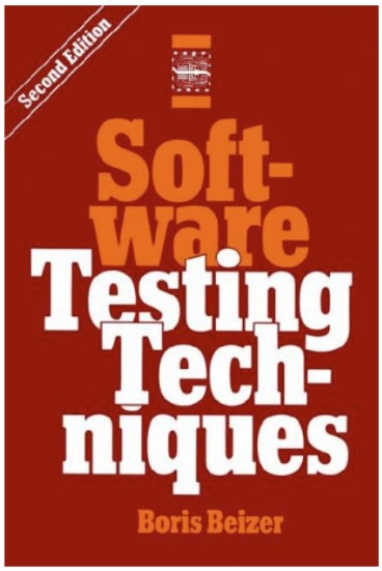

Рисунок 1. Обложка книги «Software Testing Techniques»

Метод **«белого ящика»** начал развиваться в 1960-х годах с появлением языков высокого уровня, таких как FORTRAN и COBOL. Основная идея заключалась в том, чтобы анализировать внутренние структуры программ для поиска ошибок. Метод активно использовался инженерами IBM для улучшения производительности и надежности их систем. В 1970-х годах методология была расширена за счет внедрения техник покрытия кода, таких как покрытие ветвлений и путей.

Метод **«серого ящика»** появился позже, в 1990-х годах, когда начался активный рост сложных распределенных систем. Потребовался гибридный подход, который учитывал как функциональность системы, так и её внутренние механизмы. Этот метод активно использовался при тестировании веб-приложений и API, где разработчики могли получить ограниченную информацию о внутреннем устройстве системы.

Метод «черного ящика» заключается в тестировании, которое проводится без знания внутренней структуры или реализации программного продукта. Ос-

новное внимание уделяется входным данным и выходным результатам. Тестировщик анализирует поведение программы, основываясь на спецификациях и пользовательских требованиях.

Данный метод используется для функционального тестирования, проверки работы пользовательских интерфейсов, тестирования производительности. Например, в банковских приложениях для проверки правильности обработки переводов и отображения баланса или в Web-сайтах для тестирования регистрации, авторизации, функционала корзины в интернет-магазине и т.д.

Преимущество данного метода заключаются в том, что от тестировщика не требует знания кода. А также «черный ящик» эффективен для проверки соответствия системы требованиям. Исходя из подхода к данному методу тестирования можно выделить следующие ограничения:

– невозможность выявить ошибки внутри кода;

– пропуск сценариев, связанных с внутренними механизмами работы программы.

Метод «белого ящика» заключается в тестировании с полным знанием структуры кода. Тестировщик анализирует и проверяет внутренние механизмы программы, такие как логика выполнения, ветвления и циклы.

Данный метод позволяет, например, тестировать алгоритмы сортировки, проверяя корректность выполнения на разных наборах данных, модули обработки данных и т.д.

Метод «белого ящика» позволяет выявить ошибки в логике работы программы, обеспечивая при этом полное покрытие кода. Но необходимо учитывать тот факт, что данный подход требует от тестировщика глубокого понимания кода и технологий разработки. Также данный метод может быть трудоемким и менее полезным для тестирования интерфейсов.

Метод «серого ящика» является сочетанием методов «черного» и «белого». Тестировщик имеет ограниченное знание структуры кода и использует это знание для разработки тестов. Этот метод позволяет учитывать, как поведение системы, так и её внутреннюю логику. Подходит для тестирования сложных систем, где важны как внутренние механизмы, так и пользовательские сценарии. Например, API тестирование, при котором осуществляется проверка правильности взаимодействия с внешними системами с ограниченным доступом к коду.

Метод «серого ящика» комбинирует сильные стороны обоих методов. Имеет более целостный подход к тестированию системы, но зачастую требует больше ресурсов и времени.

1.2.2. Техническое задание (ТЗ)

Техническое задание (ТЗ) — это ключевой документ, фиксирующий требования к программному продукту и определяющий его параметры. Оно служит основой для взаимодействия между заказчиком и исполнителем, минимизируя риски непонимания и упрощая контроль процесса разработки. Качественно составленное ТЗ способствует успешной реализации проекта, снижая вероятность срыва сроков или перерасхода бюджета.

В России создание и оформление ТЗ регламентируется стандартами ГОСТ, среди которых:

– **ГОСТ 19.201-78** «Техническое задание. Требования к содержанию и оформлению»;

– **ГОСТ 34.602-2020** «Межгосударственный стандарт. Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».

На международном уровне действует стандарт ISO/IEC/IEEE 29148:2018 «Системная и программная инженерия — Процессы инженерии требований». Этот стандарт описывает подходы к составлению и проверке требований, в том числе элементов ТЗ.

На основе представленных выше ГОСТ и стандарта можно выделить основные элементы технического задания.

1. **Введение** представляет краткую характеристику программы и её области применения.

2. **Основания для разработки** включают в себя причины начала проекта, ссылки на нормативные и исходные документы.

3. **Назначение разработки** формулирует задачи, которые должна решить программа. Пример: автоматизация учёта заказов для малого бизнеса с целью сокращения времени обработки заявок на 30%.

4. **Требования к программе**. В данной разделе можно выделить функциональные требования, надежность, условия эксплуатации и совместимость.

Функциональные требования описывают перечень функций с указанием их взаимосвязей. Например, возможность авторизации через социальные сети или интеграция с платёжными системами.

Надежность включает требования к отказоустойчивости и восстановлению после сбоев.

Условия эксплуатации описывают среды и параметры, в которых будет работать программа.

Совместимость определяет поддержку конкретных ОС или взаимодействие с другими системами.

5. **Требования к интерфейсу** включают прототипы экранов или текстовые описания интерфейсов. Например, кнопка «Добавить в корзину» должна быть заметной и активироваться при наведении курсора.

6. **Критерии приемки** содержат параметры, по которым заказчик оценивает соответствие продукта требованиям. Например, успешное выполнение 95% всех тест-кейсов.

7. **Требования к документации** представляют перечень обязательных документов, таких как руководства пользователя или технические описания.

8. **Порядок контроля и приемки** описывает методы тестирования и приёмочные испытания.

9. **Этапы и сроки разработки** включают план-график реализации проекта.

Помимо технического задания в процессе создания программного продукта оформляются и другие виды документации примеры, которых представлены ниже.

Пользовательская документация — это инструкции и руководства, предназначенные для конечных пользователей, с целью облегчить освоение программного продукта. Такая документация должна быть понятной и наглядной.

Например, «Руководство пользователя» для CRM-системы, объясняющее, как добавить нового клиента.

Техническая документация — это подробное описание внутреннего устройства программного продукта, включая архитектуру системы, алгоритмы, интерфейсы и базы данных. Она необходима для разработчиков и специалистов, занимающихся поддержкой и развитием продукта.

Например, диаграмма классов для отображения структуры кода или спецификация REST API для интеграции с внешними сервисами.

Тестовая документация — это набор документов, обеспечивающих проверку качества продукта. Включает тест-планы, тестовые сценарии, чек-листы и отчёты о тестировании. Её цель — удостовериться, что программное обеспечение соответствует заявленным требованиям.

Например, тест-план для проверки корректности обработки заказов в онлайн-магазине.

Системная документация — это инструкции, описывающие процесс установки, настройки и эксплуатации программного обеспечения. Она предназначена для администраторов и технических специалистов.

Например, руководство по развертыванию и настройке веб-приложения на сервере Apache.

1.2.3. Тестирование документации

Важно отметить, что в процессе разработки программного продукта на всех этапах осуществляется не только тестирование ПО, но и всей сопутствующей документации.

Целью такого тестирования является обеспечение полноты и соответствия документации заявленным требованиям, выявление и устранение двусмысленных формулировок, обеспечение актуальности содержания документации.

Существуют различные методы и подходы к тестированию документации. Среди них можно, например, выделить следующие:

1) ***ревью документации*** — это процесс командного обсуждения содержания документации на предмет логичности, точности и соответствия требованиям;

2) ***использование чек-листов*** — это процесс пошаговой проверки документации по заранее составленному списку критериев и требований;

3) ***сопоставление требований с функциональностью*** — это подход, при котором осуществляется анализ соответствия, описанного в документации функциональности программного продукта и т.д.

Примером тестирования документации может быть проверка соответствия требований в ТЗ и их отражения в пользовательской документации, например, если в ТЗ указано, что приложение должно поддерживать многоязычность, тестируется, насколько правильно и полно переведены все текстовые элементы в интерфейсе. Другим примером является проверка технической документации на соответствие актуальным версиям используемых библиотек и фреймворков — если указаны устаревшие версии, это может привести к неправильной настройке или ошибкам при эксплуатации системы. Также важно тестировать, соответствуют ли диаграммы и схемы в технической документации реальному коду и архитектуре системы, чтобы избежать несоответствий, которые могут затруднить поддержку и развитие продукта.

1.2.4. Инструменты для работы с документацией

Для создания и управления техническим заданием (ТЗ) часто используются инструменты, такие как Confluence, Microsoft Word или Google Docs, которые обеспечивают удобное оформление, совместную работу и версионность документации. Для управления тестовой документацией популярны такие системы,

как TestRail, Zephyr и Jira, которые позволяют отслеживать тестовые сценарии, результаты тестирования и исправления ошибок в едином интерфейсе. Для проектирования архитектуры и создания схем, диаграмм и чертежей широко применяются инструменты, такие как Lucidchart, Draw.io и Enterprise Architect, которые предлагают мощные возможности для визуализации и проектирования сложных систем. Эти инструменты значительно облегчают процесс разработки, улучшая координацию команды и точность документации.

Для создания и управления техническим заданием (ТЗ) в России можно использовать отечественные решения, такие как 1С: Документооборот или Дело. Эти системы предоставляют функциональность для хранения, совместной работы и управления версией документации, соответствующих российским стандартам. Для управления тестовой документацией в России популярны такие продукты, как Testit и Testrail (имеющий локализованную версию). Эти платформы позволяют организовать процесс тестирования, создания тестовых сценариев и отслеживания ошибок. Для проектирования архитектуры и создания диаграмм можно использовать такие инструменты, как Archi (открытое российское решение для моделирования архитектуры предприятия) и Visual Paradigm с локализацией для российского рынка. Эти решения позволяют интегрировать стандарты и подходы, соответствующие российским нормативам и требованиям.

Техническое задание и качественная документация играют ключевую роль в успехе разработки программного обеспечения. Использование стандартов ГОСТ и международных методик, таких как ISO/IEC/IEEE 29148:2018, позволяет обеспечить полноту и качество требований. Регулярное тестирование и актуализация документации поддерживают её релевантность и высокую эффективность.

1.3. Описание работы

Практическая работа №1 состоит из 2-х частей.

1.3.1. Часть 1. Разработка технического задания и программного продукта

Для выполнения задания группа должна разделиться на команды численностью 2–4 человека. Каждая команда придумывает себе название, которое сохраняется на протяжении всего курса дисциплины.

Участники команды для выполнения 1 части практического задания подготавливает следующие материалы:

1) программный продукт (далее — ПП);

2) техническое задание (далее — ТЗ) на разработанный ПП, а также дополнительную документацию, если требуется.

При разработке программного необходимо учитывать следующие требования:

1. ПП должен быть исполняемым и предоставляться в удобном формате для тестирования другой командой.
2. В случае нетривиального исполнения необходимо приложить полное и достаточное описание запуска (*прописывается в разделе №7 ТЗ*).
3. ПП должен содержать 5–8 ошибок различной природы (логические, интерфейсные, синтаксические и т. д.).

Необходимо разработать Техническое задание (ТЗ) для создания программного продукта, который будет протестирован в рамках практической работы. В процессе разработки ТЗ необходимо учесть следующие аспекты:

1. ***Введение*** должно содержать краткое описание программы и области её применения. Команда должна ясно обозначить, для каких целей и в каких сферах будет использоваться разрабатываемый продукт.
2. ***Основания для разработки*** описывают причины начала проекта. Также в этом разделе должны быть указаны ссылки на исходные и нормативные документы, которые послужили основой для разработки программного продукта.
3. ***Назначение разработки*** формулирует задачи, которые должна решить программа. Например, можно указать цель автоматизации учёта заказов для малого бизнеса с целью сокращения времени обработки заявок на 30%.
4. ***Требования к программе*** включают несколько подпунктов. Во-первых, функциональные требования — это перечень функций с указанием их взаимосвязей. Например, возможность авторизации через социальные сети или интеграция с платёжными системами. Во-вторых, требования к надежности, которые описывают отказоустойчивость программы и её способность восстанавливаться после сбоев. В-третьих, условия эксплуатации, которые должны указать среду и параметры, в которых программа будет функционировать. В-четвертых, требования к совместимости, то есть поддержка конкретных операционных систем или взаимодействие с другими системами.
5. ***Требования к интерфейсу*** должны содержать описание элементов интерфейса, таких как кнопки, меню и другие элементы управления. Например, кнопка «Добавить в корзину» должна быть заметной и активироваться при наведении курсора.

6. ***Критерии приемки*** определяют параметры, по которым заказчик будет оценивать соответствие продукта требованиям. Например, успешное выполнение 95% всех тест-кейсов, может быть, одним из таких критериев.

7. ***Требования к документации*** включают перечень обязательных документов, таких как руководство пользователя, технические описания и другие виды документации, которые должны быть предоставлены вместе с продуктом.

Примечание: *Раздел реализуется при необходимости.*

8. ***Порядок контроля и приемки*** описывает методы тестирования и приемочные испытания, которые будут использоваться для проверки соответствия программного продукта требованиям, указанным в ТЗ.

Примечание: *Раздел описывается с учётом специфики теоретической части практической работы.*

9. ***Этапы и сроки разработки*** должны содержать план-график реализации проекта. В этом разделе указывается последовательность работ и сроки, в которые они должны быть выполнены.

1.3.2. Часть 2. Тестирование ПП

На момент выполнения второй части практического задания команды должны завершить выполнение 1 части.

Все созданные ПП размещаются в общем репозитории или передаются преподавателю. Преподаватель может назначить команды для тестирования ПП другой группы с учетом сложности и объема работы, или команды могут провести самостоятельный обмен программными продуктами между собой. Пример процесса распределения проиллюстрирован на рис. 2.

Команда-разработчик	Команда-тестер
Команда A	Команда B
Команда B	Команда C
Команда C	Команда A

Рисунок 2. Пример распределения программных продуктов между командами

При выполнении второй части практической работы командам необходимо выполнить следующие действия:

1) изучить предоставленные ПП, ТЗ и документацию;

2) провести анализ полноты и качества описания ПП в ТЗ и документации;

3) протестировать ПП, используя метод «черного ящика», т.е. члены команды должны подобрать входные данные и условия, которые могут вызвать некорректную работу программы;

4) проанализировать, соответствуют ли результаты работы ПП требованиям и адекватно ли оно реагирует на непредусмотренные ситуации;

5) задокументировать выявленные ошибки в ПП и прилагаемой документации.

Пример документирования найденной ошибки в ПП:

1. Идентификатор.

ТС-001

2. Название.

Проверка авторизации с корректными учетными данными.

3. Описание.

Убедиться, что пользователь может успешно войти в систему при вводе корректного логина и пароля.

4. Предварительные условия:

а) пользователь зарегистрирован в системе;

б) браузер открыт, страница авторизации загружена.

5. Шаги выполнения:

а) в поле «Логин» ввести зарегистрированный логин;

б) в поле «Пароль» ввести корректный пароль;

в) нажать кнопку «Войти».

6. Ожидаемый результат.

Пользователь успешно авторизуется и перенаправляется на главную страницу приложения. Отображается сообщение «Добро пожаловать, *Имя пользователя*!».

7. Фактический результат.

Заполняется после выполнения теста.

8. Статус.

Passed или Failed (*заполняется после выполнения теста*).

1.4. Итоговый отчёт

По результатам командной работы отчёт должен содержать:

1. Титульный лист, включающий в себя наименование работы, название команды, состав команды, дата выполнения, название учебной дисциплины.

2. Техническое задание (ТЗ) собственного программного продукта, содержащее введение, основания для разработки, назначение разработки, требования к программе (функциональные, к интерфейсу, условия эксплуатации, надежность, совместимость), критерии приемки, порядок контроля и приемки, а также этапы и сроки разработки.

3. Дополнительную документацию на собственный программный продукт, включающую руководство пользователя, описание архитектуры системы, схемы или диаграммы.

4. Описание внесённых ошибок в собственное ПО, в котором указываются преднамеренно добавленные ошибки, их типы (логические, интерфейсные, синтаксические и т. д.), а также способы их обнаружения (например, при определённых условиях выполнения).

5. Техническое задание (ТЗ) и документацию программного продукта другой команды, включая анализ их полноты, логичности и соответствия требованиям. Здесь же фиксируются замечания и рекомендации.

6. Результаты тестирования программного продукта другой команды, с указанием:

а) обнаруженных ошибок (критические, средние, незначительные);

б) пошагового описания, как они были выявлены (входные данные, ожидаемый результат, фактический результат);

в) подтверждений в виде скриншотов, логов и другого.

7. Анализ документации другой команды, где подробно описываются выявленные недочёты: неактуальная информация, двусмысленные формулировки, отсутствие деталей.

8. Заключение, включающее общую оценку качества программного продукта и документации, выводы о соответствии продукта ТЗ и рекомендации по улучшению.

9. Приложения (при необходимости): дополнительные материалы (коды, логи тестирования, скриншоты и т. д.).

Отчёт должен быть оформлен в соответствии с ГОСТ 19.401-78 и ГОСТ 34.602-2020. Требования включают стандарты на титульный лист, использование единообразного шрифта, оформление таблиц и диаграмм, наличие нумерации страниц.

Итоговая оценка работы будет зависеть от полноты отчёта, качества выполнения задания, соответствия оформления стандартам и презентации результатов на защите.