

КРАТКИЙ СПРАВОЧНИК ТЕСТИРОВЩИКА



Дмитрий Самойлов

Дмитрий Самойлов

**Краткий справочник
тестировщика**

«Издательские решения»

Самойлов Д.

Краткий справочник тестировщика / Д. Самойлов —
«Издательские решения»,

ISBN 978-5-00-598529-3

Данная книга является кратким справочником для тестировщиков, который позволит ознакомиться с основами тестирования, методами и инструментами тестирования, а также с процессом тест-планирования и тест-дизайна. Кроме того, в книге описаны особенности тестирования в Agile, приведен краткий обзор автоматизации тестирования и инструментов, обзор тестирования безопасности, производительности и особенностей тестирования мобильных и web-приложений.

ISBN 978-5-00-598529-3

© Самойлов Д.
© Издательские решения

Содержание

Введение	6
Основные термины и определения	7
Основы тестирования	9
Жизненный цикл разработки ПО	9
Основные постулаты тестирования	10
Классификация видов тестирования	11
Этапы тестирования	13
Тест-планирование	14
Цели и задачи тест-планирования	14
Разработка тест-плана	15
Оценка рисков	16
Тест-дизайн	17
Цели и задачи тест-дизайна	17
Тест-кейсы и их структура	18
Чек-листы	20
Конец ознакомительного фрагмента.	21

Краткий справочник тестировщика

Дмитрий Самойлов

© Дмитрий Самойлов, 2023

ISBN 978-5-0059-8529-3

Создано в интеллектуальной издательской системе Ridero

Введение

Тестирование является важной частью процесса разработки программного обеспечения, которая позволяет выявлять ошибки и дефекты в работе ПО. Каждый тестировщик должен иметь хорошие знания и навыки в области тестирования, чтобы обеспечить высокое качество продукта.

Данная книга является кратким справочником для тестировщиков, который поможет ознакомиться с основами тестирования, методами и инструментами тестирования, а также с процессом тест-планирования и тест-дизайна. Кроме того, в книге описаны особенности тестирования в Agile, приведен краткий обзор автоматизации тестирования и инструментов, обзор тестирования безопасности, производительности и особенностей тестирования мобильных и web приложений.

Книга написана на понятном языке, и ее структура позволяет быстро находить необходимую информацию. Надеюсь, что данный путеводитель будет полезен тем, кто начинает свой путь в тестировании программного обеспечения.

Основные термины и определения

- **Тестирование** (Testing): процесс проверки соответствия продукта его требованиям и ожиданиям пользователей.
- **Тестировщик** (Tester): специалист, занимающийся тестированием продукта, отвечающий за обнаружение дефектов и обеспечение качества продукта.
- **Качество** (Quality): свойство продукта, которое определяет его способность удовлетворять потребности и ожидания пользователей в соответствии с заданными требованиями и стандартами. Качество в тестировании связано с тем, насколько хорошо продукт соответствует его целям и требованиям, а также насколько эффективно он выполняет свои функции и задачи.
- **Тест-план** (Test Plan): документ, описывающий общий подход к тестированию продукта, содержащий информацию о целях, методах, ресурсах и расписании тестирования.
- **Тестовый набор** (Test Suite): группа связанных тест-кейсов, которые выполняются последовательно, чтобы проверить определенную функциональность продукта.
- **Тестовый сценарий** (Test Scenario): последовательность шагов, необходимых для проверки определенной функциональности продукта.
- **Тест-кейс** (Test Case): набор инструкций для проведения конкретного тестирования и проверки корректной работы определенного функционала продукта.
- **Покрытие тестами** (Test Coverage): процентное соотношение между количеством выполненных тестовых сценариев или тест-кейсов и общим количеством функциональности продукта, проверяемой в этих тестах.
- **Дефект** (Defect): отклонение от требований или ожиданий пользователей, обнаруженное в процессе тестирования.
- **Баг** (Bug): термин, используемый для обозначения дефекта или ошибки в работе программного обеспечения.
- **Тестовое окружение** (Test Environment): среда, в которой проводится тестирование продукта, включающая в себя программное и аппаратное обеспечение, настройки и конфигурации.
- **Регрессионное тестирование** (Regression Testing): процесс повторного тестирования уже протестированных функций или участков кода после внесения изменений, чтобы проверить, что изменения не привели к появлению новых дефектов.
- **Автоматизированное тестирование** (Automated Testing): процесс проведения тестирования с использованием специальных программных инструментов для автоматизации выполнения тестовых сценариев и тест-кейсов.
- **Интеграционное тестирование** (Integration Testing): процесс проверки работоспособности компонентов системы в совокупности, в том числе взаимодействия между ними.
- **Юнит-тестирование** (Unit Testing): процесс тестирования отдельных компонентов программного обеспечения (например, функций, классов), чтобы проверить их корректность и работоспособность.
- **Тестирование производительности** (Performance Testing): процесс тестирования, направленный на оценку работоспособности и производительности продукта, включая проверку его способности обрабатывать большое количество запросов и обеспечивать быстрый отклик.
- **Тестирование безопасности** (Security Testing): процесс тестирования, направленный на оценку уровня защищенности продукта от внешних угроз, таких как хакерские атаки, вирусы и т. д.

– **Тестирование совместимости** (Compatibility Testing): процесс тестирования, направленный на проверку работоспособности продукта в различных окружениях и на разных платформах.

– **Тестирование пользовательского интерфейса** (User Interface Testing): процесс тестирования, направленный на проверку корректности работы пользовательского интерфейса продукта, включая взаимодействие с пользователем и удобство использования.

Основы тестирования

Жизненный цикл разработки ПО

Жизненный цикл разработки ПО (Software Development Life Cycle, SDLC) – это процесс разработки ПО, который включает в себя различные фазы, начиная от анализа требований и заканчивая сопровождением и поддержкой ПО после его внедрения. Жизненный цикл разработки ПО обычно включает следующие основные фазы:

- **Анализ требований:** определение требований к ПО на основе потребностей заказчика и пользователей, описание функциональных и нефункциональных требований, создание спецификаций требований.

- **Проектирование:** разработка архитектуры ПО, создание диаграмм и схем, определение функциональности и интерфейсов.

- **Разработка:** создание кода ПО на основе заданных требований и дизайна, тестирование кода на соответствие требованиям.

- **Тестирование:** проверка работоспособности ПО на соответствие требованиям, выявление дефектов и ошибок в работе ПО, исправление и повторное тестирование.

- **Внедрение:** установка ПО на рабочие станции пользователей, настройка и интеграция с другими системами.

- **Сопровождение и поддержка:** обеспечение работоспособности ПО, исправление ошибок и дефектов, обновление ПО для улучшения его функциональности и совместимости с другими системами.

Важно отметить, что жизненный цикл разработки ПО может варьироваться в зависимости от методологии разработки (например, водопадной, Agile и т.д.) и конкретного проекта. Тестирование является важной частью жизненного цикла разработки ПО, и тестировщики должны уметь эффективно взаимодействовать с разработчиками и другими членами команды в каждой фазе процесса.

Основные постулаты тестирования

Постулаты тестирования – это основные принципы и подходы, которые лежат в основе тестирования программного обеспечения. Вот некоторые из главных постулатов тестирования:

– **Полное тестирование невозможно:** невозможно провести тестирование, которое охватит все возможные сценарии использования продукта. Поэтому тестировщики должны использовать свой опыт и знания, чтобы выбрать тестовые сценарии, которые наиболее вероятно приведут к проблемам.

– **Дефекты накапливаются:** в ходе тестирования всегда будут находиться новые дефекты, и количество дефектов может накапливаться со временем. Поэтому тестировщики должны непрерывно тестировать продукт и исправлять найденные проблемы.

– **Тестирование не может доказать отсутствие ошибок:** тестирование может показать наличие ошибок, но не может гарантировать, что их нет. Тестирование может улучшить качество продукта и уменьшить риск ошибок, но не может исключить их полностью.

– **Тестирование зависит от контекста:** каждый проект уникален, и контекст тестирования может сильно варьироваться. Тестирование должно учитывать требования и цели проекта, а также особенности окружения, в котором будет использоваться продукт.

– **Безусловная вера в тестирование опасна:** тестирование не является абсолютным инструментом для обнаружения всех дефектов. Поэтому необходимо использовать и другие методы контроля качества, например, код-ревью и анализ требований.

– **Тестирование должно начинаться как можно раньше:** тестирование должно начинаться с самого начала разработки продукта и должно включать тестирование на каждом этапе разработки. Это поможет обнаружить проблемы и ошибки раньше, когда они легче исправляются.

– **Тестирование должно быть систематическим и документированным:** тестирование должно проводиться систематически и документироваться, чтобы обеспечить повторяемость и воспроизводимость результатов тестирования. Это также помогает обеспечить качество продукта и улучшить процесс разработки.

Эти постулаты помогают тестировщикам ориентироваться в работе и сделать процесс тестирования более эффективным.

Классификация видов тестирования

Когда разработчики создают программное обеспечение (ПО), необходимо проводить тестирование, чтобы убедиться в правильной работе приложения и соответствии его функциональным и нефункциональным требованиям. Соответственно, существует два основных вида тестирования ПО: функциональное и нефункциональное.

Функциональное тестирование – это тестирование, которое направлено на то, чтобы убедиться в том, что ПО способно выполнять свои функции и соответствовать функциональным требованиям, установленным для него.

Нефункциональное тестирование включает в себя проверку соответствия нефункциональным требованиям, таким как удобство использования для конечных пользователей, масштабируемость, производительность, безопасность, надежность и др. Виды нефункционального тестирования:

– **Тестирование пользовательского интерфейса (UI Testing)** – это процесс проверки интерфейса на соответствие заданным требованиям, таким как размер, шрифт, цвет и последовательность действий.

– **Тестирование удобства использования (Usability Testing)** – это метод проверки, который оценивает удобство использования, обучаемость, понятность и привлекательность продукта для пользователей в соответствии с контекстом использования. Он состоит из UX (User Experience), который описывает взаимодействие пользователя с продуктом, и UI (User Interface), который представляет собой инструмент для взаимодействия между пользователем и веб-ресурсом.

– **Тестирование безопасности (Security Testing)** – это стратегия, которая используется для проверки безопасности системы и анализа рисков, связанных с защитой приложения от атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным и другими видами угроз.

– **Инсталляционное тестирование (Installation Testing)** – это проверка успешной установки и настройки, обновления или удаления приложения на различных платформах.

– **Конфигурационное тестирование (Configuration Testing)** – это специальный вид тестирования, который проверяет работу программного обеспечения в разных конфигурациях системы, включая заявленные платформы, поддерживаемые драйверы и различные конфигурации компьютеров.

– **Тестирование на отказ и восстановление (Failover and Recovery Testing)** – это проверка, которая оценивает способность продукта успешно восстанавливаться после возникновения сбоев, связанных с ошибками программного обеспечения, отказами оборудования или проблемами связи.

– **Тестирование локализации (Localization Testing)** – это проверка адаптации программного обеспечения для определенной аудитории в соответствии с ее культурными особенностями, включая язык, формат даты и времени, валюту и т. д.

Однако виды тестирования можно классифицировать по различным параметрам. Например:

Классификация тестирования **по позитивности сценария**:

– **Позитивное тестирование** – использует только корректные данные для проверки правильности функций приложения.

– **Негативное тестирование** – использует как корректные, так и некорректные данные для проверки исключительных ситуаций и часто включает некорректные операции.

Классификация тестирования **по знанию системы**:

– **Тестирование белого ящика (White Box)** – тестирование ПО с полным доступом к коду проекта, при котором тестировщик знает внутреннюю структуру, устройство и реализацию системы.

– **Тестирование серого ящика** – метод тестирования ПО, который предполагает частичный доступ к коду проекта, объединяя в себе White Box и Black Box методы.

– **Тестирование чёрного ящика (Black Box)** – метод тестирования ПО, при котором тестировщик не имеет доступа к внутренней структуре и реализации системы, а основывается на работе с внешним интерфейсом тестируемой системы.

Классификация тестирования **по исполнителям тестирования:**

– **Альфа-тестирование** – ранняя версия программного продукта, которая тестируется внутри организации-разработчика, а также может привлекать конечных пользователей для частичного тестирования.

– **Бета-тестирование** – готовое или практически готовое ПО, которое выпускается для ограниченного числа пользователей для получения отзывов и реакции клиентов на продукт и внесения соответствующих изменений в ПО.

Классификация тестирования **по уровню тестирования:**

– **Модульное (компонентное) тестирование** – проводится разработчиками, чтобы проверить отдельные компоненты системы, такие как объекты, классы, функции и т. д. и выявить дефекты в коде.

– **Интеграционное тестирование** – направлено на проверку корректности взаимодействия между несколькими модулями системы, объединенными в единое целое.

– **Системное тестирование** – это проверка функциональных и не функциональных требований всей системы в целом, с целью выявления дефектов и оценки качества системы.

– **Операционное тестирование** – финальный шаг валидации, который проводится в среде эксплуатации для проверки соответствия системы потребностям пользователей и выявления нефункциональных проблем.

Классификация тестирования **по исполнению кода:**

– **Статическое тестирование** – это процесс проверки артефактов разработки (кода, документации) путем анализа без его фактического выполнения. Цель этого тестирования – выявить проблемы и ошибки на ранних стадиях разработки, чтобы избежать их в будущем.

– **Динамическое тестирование** – это проверка работы системы во время ее выполнения, т.е. с запуском программного кода и проверкой его результатов. Этот вид тестирования направлен на проверку функциональности и нахождение проблем, которые могут возникнуть в процессе выполнения.

Классификация тестирования **по хронологии выполнения:**

– **Повторное/подтверждающее тестирование (re-testing/confirmation testing)** – это тестирование, при котором запускаются тестовые сценарии, которые выявили ошибки в предыдущем тестовом цикле, с целью подтверждения того, что ошибки были успешно исправлены.

– **Регрессионное тестирование (regression testing)** – это тестирование, которое проводится после внесения изменений в код приложения, чтобы убедиться, что изменения не вызвали ошибок в неизменных областях кода. Также проверяется, что исправление ошибок и любые изменения в коде приложения не повлияли на работу других модулей ПО и не вызвали новых ошибок.

– **Приемочное тестирование** – это проверка соответствия системы требованиям пользователя, бизнес-процессам и потребностям.

Этапы тестирования

Этапы тестирования являются важной частью жизненного цикла разработки ПО. Они позволяют эффективно проверять функциональность ПО, выявлять и исправлять ошибки, а также обеспечивать высокое качество конечного продукта. Ниже перечислены основные этапы тестирования:

– **Планирование тестирования:** на этом этапе определяются цели тестирования, составляется план тестирования, определяется состав тестовых случаев, определяются критерии окончания тестирования.

– **Проектирование тестов:** на этом этапе составляются тестовые сценарии, определяются входные данные и ожидаемые результаты для каждого тестового случая.

– **Подготовка тестовых данных:** на этом этапе создаются тестовые данные, которые будут использоваться при проведении тестирования.

– **Выполнение тестов:** на этом этапе проводятся тесты в соответствии с планом тестирования. Результаты тестирования записываются и анализируются.

– **Анализ результатов тестирования:** на этом этапе анализируются результаты тестирования, выявляются ошибки и проблемы, их описывают, классифицируют по уровню серьезности и приоритету.

– **Исправление ошибок:** на этом этапе исправляются выявленные ошибки и проблемы.

– **Повторное тестирование:** после исправления ошибок проводится повторное тестирование, чтобы убедиться, что ошибки действительно были исправлены и не появились новые.

– **Приемочное тестирование:** на этом этапе проводится финальное тестирование, которое выполняется с целью убедиться, что ПО полностью соответствует требованиям заказчика и готово к выходу в эксплуатацию.

Каждый этап тестирования имеет свои специфические особенности и выполняется в определенной последовательности, что позволяет обеспечить высокое качество продукта.

Тест-планирование

Цели и задачи тест-планирования

Цель тест-планирования – определить процесс и стратегию тестирования, которые позволят достичь заданных целей и обеспечить высокое качество конечного продукта.

Основные задачи тест-планирования:

– **Определение общей стратегии тестирования:** на этом этапе происходит определение подхода к тестированию и выбор методов и техник, которые будут использоваться при тестировании.

– **Определение объема тестирования:** на этом этапе определяется, какие части системы будут тестироваться, а также какие функциональные и нефункциональные требования должны быть протестированы.

– **Определение рисков и приоритетов:** на этом этапе происходит анализ возможных рисков и их влияния на систему, а также определение приоритетов для тестирования тех частей системы, которые являются наиболее критическими.

– **Определение тестовых сценариев и тест-кейсов:** на этом этапе создаются тест-кейсы и определяются тестовые сценарии, которые будут использоваться для проверки системы на соответствие требованиям.

– **Определение тестовых данных:** на этом этапе производится подготовка тестовых данных, которые будут использоваться при тестировании.

– **Определение критериев приемки:** на этом этапе происходит определение критериев приемки системы, которые должны быть выполнены, чтобы система была готова к релизу.

– **Определение расписания и ресурсов:** на этом этапе определяется расписание тестирования и ресурсы, необходимые для его выполнения.

– **Определение процесса управления ошибками:** на этом этапе происходит определение процесса управления ошибками и дефектами, включая их отслеживание, отчетность и управление жизненным циклом дефектов.

В результате тест-планирования создается план тестирования, который содержит информацию о целях, методах и процессах тестирования, а также о расписании, ресурсах и критериях приемки. Этот план является основным руководством для тестировщиков в процессе выполнения тестирования и позволяет обеспечить высокое качество конечного продукта.

Разработка тест-плана

Разработка тест-плана – это процесс создания документа, который описывает общий план тестирования для проекта. Тест-план является основным документом, который определяет, как будет проводиться тестирование системы, и содержит детальные инструкции для всех этапов тестирования. Он включает в себя следующие этапы:

– **Определение целей и задач тестирования:** на этом этапе определяются цели тестирования, например, проверка функциональности системы, производительности или безопасности. Также определяются задачи, необходимые для достижения этих целей, такие как создание тест-кейсов, проведение функционального тестирования, нагрузочного тестирования и т. д.

– **Определение области тестирования:** на этом этапе определяется область тестирования, которую необходимо покрыть в процессе тестирования. Это может быть конкретный функциональный модуль, компонент системы или вся система в целом.

– **Определение методов тестирования:** на этом этапе выбираются методы тестирования, которые будут использоваться для проведения тестов. Это может быть ручное или автоматизированное тестирование, функциональное или нефункциональное тестирование, нагрузочное тестирование и т. д.

– **Определение тестовых сценариев и тест-кейсов:** на этом этапе разрабатываются тестовые сценарии и тест-кейсы для проверки функциональности и нефункциональных требований системы. Тест-кейсы должны содержать подробные инструкции для проведения тестов, описывать ожидаемый результат и определять тестовые данные.

– **Определение критериев окончания тестирования:** на этом этапе определяются критерии, которые будут использоваться для оценки результатов тестирования и определения того, когда тестирование может быть завершено. Критерии могут включать в себя достижение определенного покрытия тестами, отсутствие критических ошибок или удовлетворение определенных требований к качеству.

– **Определение графика тестирования:** на этом этапе разрабатывается график тестирования, который определяет, когда будут проводиться тесты и какие ресурсы необходимы для проведения тестирования. График должен быть разработан с учетом сроков проекта и уровня рисков, связанных с проведением тестирования.

– **Определение ответственностей:** на этом этапе определяются роли и ответственности членов команды тестирования. Например, кто будет разрабатывать тест-кейсы, кто будет проводить тесты и кто будет отчитываться о результатах тестирования.

– **Определение ресурсов:** на этом этапе определяются ресурсы, необходимые для проведения тестирования, такие как персонал, оборудование и программное обеспечение.

– **Оценка рисков:** на этом этапе определяются риски, связанные с проведением тестирования, и разрабатываются планы по их управлению.

– **Утверждение тест-плана:** после того, как тест-план разработан, он должен быть утвержден соответствующими заинтересованными сторонами, такими как руководство проекта, заказчик или команда разработчиков.

Разработка тест-плана – это важный процесс в жизненном цикле разработки ПО, который помогает гарантировать, что тестирование будет проведено в соответствии с требованиями проекта и наилучшими практиками тестирования.

Оценка рисков

Оценка рисков является важной частью процесса тестирования и позволяет определить потенциальные проблемы, которые могут возникнуть во время тестирования, и разработать стратегии по их управлению. Оценка рисков позволяет уменьшить вероятность возникновения проблем и минимизировать их воздействие на проект.

Процесс оценки рисков включает в себя следующие шаги:

– **Идентификация рисков**: определение потенциальных проблем, которые могут возникнуть во время тестирования. Для этого можно использовать опыт предыдущих проектов, анализ требований, общение с командой разработки и другие источники информации.

– **Оценка вероятности возникновения рисков**: оценка вероятности того, что каждый потенциальный риск возникнет во время тестирования. Это может быть сделано на основе данных из прошлых проектов, опыта команды и других факторов.

– **Оценка влияния рисков**: оценка влияния каждого риска на проект, если он произойдет. Например, некоторые риски могут привести к задержке сроков, повышению затрат или к критическим ошибкам в приложении.

– **Разработка стратегий управления рисками**: разработка плана действий для управления каждым риском. Это может включать в себя разработку запасных планов, увеличение тестового покрытия для конкретных функций, использование инструментов тестирования и т. д.

– **Мониторинг и управление рисками**: мониторинг и управление рисками во время тестирования. Это может включать в себя регулярный анализ статуса рисков, обновление стратегий управления рисками и другие меры.

Цель оценки рисков – определить и управлять рисками, связанными с тестированием, чтобы минимизировать негативное влияние на проект и улучшить качество продукта. Важно помнить, что оценка рисков – это непрерывный процесс, который должен проводиться на протяжении всего процесса тестирования.

Тест-дизайн

Цели и задачи тест-дизайна

Цель тест-дизайна – разработка тестовых случаев, которые позволят эффективно проверить функциональность программного продукта и обнаружить ошибки. Тест-дизайн является важной частью процесса тестирования и может существенно повлиять на качество продукта.

Основные задачи тест-дизайна включают:

– **Разработка тестовых случаев:** определение входных данных и ожидаемых результатов для каждого функционала продукта. Это позволяет создать тестовые случаи, которые будут эффективно проверять функциональность продукта и помогать обнаруживать ошибки.

– **Определение приоритетов:** определение приоритетов для каждого функционала продукта, чтобы определить, какие тесты должны быть выполнены первыми и какие могут быть выполнены позже.

– **Определение покрытия:** определение покрытия тестами для каждого функционала продукта. Это позволяет определить, как много тестов необходимо выполнить для каждого функционала, чтобы быть уверенным, что она работает правильно.

– **Создание тестовых данных:** создание набора тестовых данных, которые будут использоваться для выполнения тестовых случаев. Тестовые данные должны быть достаточно разнообразными, чтобы проверить различные варианты использования продукта.

– **Определение ожидаемых результатов:** определение ожидаемых результатов для каждого тестового случая. Это позволяет определить, работает ли продукт правильно и соответствует ли он требованиям.

– **Проверка тестовых случаев:** проверка тестовых случаев, чтобы убедиться, что они покрывают все функции продукта и соответствуют требованиям.

– **Обновление тестовых случаев:** обновление тестовых случаев при изменении требований или функций продукта.

Цели и задачи тест-дизайна направлены на создание тестовых случаев, которые эффективно проверяют функциональность продукта и помогают обнаруживать ошибки. Хороший тест-дизайн может значительно повысить качество продукта и сэкономить время и деньги на более поздних стадиях тестирования и разработки.

Тест-кейсы и их структура

Составление тест-кейсов – это важный процесс, который помогает эффективно тестировать программное обеспечение и обнаруживать ошибки. Следующие шаги могут помочь правильно составлять тест-кейсы:

– **Определение целей тестирования:** определение целей тестирования позволяет определить, какие тесты необходимо провести. Необходимо определить, что именно должен делать продукт, какие функции он должен выполнять и какие проблемы могут возникнуть. Важно также уделить внимание негативному тестированию.

– **Определение входных данных:** определение входных данных для каждого тестового случая позволяет проверить, работает ли продукт правильно. Необходимо определить, какие данные будут использоваться для каждого теста, какие значения будут использоваться и как они будут вводиться.

– **Определение ожидаемых результатов:** определение ожидаемых результатов для каждого тестового случая позволяет определить, работает ли продукт правильно и соответствует ли он требованиям. Необходимо определить, какие результаты должны быть получены, как они должны быть отображены и какие сообщения об ошибках должны появиться, если они есть.

– **Создание шагов тестирования:** создание шагов тестирования для каждого тестового случая позволяет определить, какие шаги необходимо выполнить для тестирования каждой функции продукта. Шаги должны быть конкретными и четкими, чтобы тестирование было эффективным.

– **Определение предусловий и постусловий:** определение предусловий и постусловий для каждого тестового случая позволяет определить, какие условия должны быть выполнены перед запуском теста и что должно произойти после его завершения.

– **Проверка тест-кейсов:** проверка тест-кейсов позволяет убедиться, что они покрывают все функции продукта и соответствуют требованиям. Необходимо также проверить, что тест-кейсы легко читаются и понятны для других членов команды.

– **Обновление тест-кейсов:** обновление тест-кейсов при изменении требований или функций продукта позволяет сохранить их актуальность.

Важно помнить, что тест-кейсы должны быть написаны так, чтобы их можно было легко понять и выполнить другими членами команды. Каждый тест-кейс должен покрывать только одну функцию продукта. При составлении тест-кейсов не забывайте составлять негативные кейсы. Негативное тестирование является важной частью процесса тестирования, поскольку направлено на проверку того, как система обрабатывает неправильные или неверные данные и действия пользователей. Негативные тесты помогают обнаружить потенциальные уязвимости, ошибки и неожиданное поведение системы.

Структура тест-кейсов

Хороший тест-кейс должен иметь структуру, которая обеспечивает понятность, четкость и легкость его использования. Ниже представлена стандартная структура хорошего тест-кейса:

– **Идентификатор:** уникальный номер тест-кейса для идентификации и связывания с требованиями или другими документами.

– **Название:** краткое описание функциональности, которую тестирует данный тест-кейс.

– **Окружение:** на каком окружении (устройство/браузер/ОС) должно проводиться тестирование.

– **Описание:** подробное описание того, что тестируется в данном тест-кейсе.

– **Предусловия:** любые условия, которые должны быть выполнены перед запуском тест-кейса, например, наличие определенных данных или доступ к определенной системе.

– **Шаги:** последовательность действий, которые тестировщик должен выполнить, чтобы протестировать определенную функциональность.

– **Ожидаемый результат:** четкое описание того, что ожидается в результате выполнения каждого шага тест-кейса или тест-кейса в целом.

– **Фактический результат:** фактический результат выполнения каждого шага тест-кейса или тест-кейса в целом.

– **Статус:** текущее состояние тест-кейса, например, пройден, провален или в процессе выполнения.

– **Постусловия:** любые условия, которые должны быть выполнены после завершения тест-кейса, например, восстановление данных или перезагрузка системы.

– **Заключение:** краткий комментарий о результатах тестирования, включая проблемы, которые были обнаружены в процессе выполнения тест-кейса, и рекомендации для дальнейших шагов.

Важно отметить, что структура тест-кейса может различаться в зависимости от проекта, инструментов тестирования и методологии разработки. Однако, необходимо следовать общим принципам, чтобы тест-кейсы были эффективными и максимально информативными для тестировщиков и других членов команды разработки.

Ниже приведен пример тест-кейса для тестирования функции отправки электронной почты в веб-приложении:

TK001. Отправка электронной почты

Окружение: Android 12, Chromev.111

Описание: Тестирование функции отправки электронной почты в веб-приложении.

Предусловия: Авторизоваться в веб-приложении.

Шаги:

- 1. Нажать на кнопку «Написать письмо».*
- 2. Ввести адрес электронной почты получателя в поле «Кому».*
- 3. Ввести тему письма в поле «Тема».*
- 4. Ввести текст письма в поле «Текст письма».*
- 5. Нажать на кнопку «Отправить».*

Ожидаемый результат:

Письмо успешно отправлено.

Получатель успешно получает письмо.

На странице отображается сообщение об успешной отправке письма.

Фактический результат:

Письмо не отправлено.

При отправке письма возникает ошибка.

Получатель не получает письмо.

Чек-листы

Чек-листы являются важным инструментом при тестировании ПО, позволяя тестировщикам систематизировать и упростить процесс тестирования. Вот некоторые рекомендации, как правильно составлять чек-листы при тестировании ПО:

– **Определите цель чек-листа:** определите, какую функциональность или компонент ПО вы будете тестировать, и какие критерии качества вы хотите проверить.

– **Составьте список элементов для проверки:** определите конкретные элементы, которые вы будете проверять, например, поля ввода, кнопки, сообщения об ошибках, а также взаимодействие компонентов между собой.

– **Определите ожидаемые результаты:** для каждого элемента определите, какое поведение ожидается при его проверке. Например, если вы проверяете кнопку «Отправить», то ожидаемый результат может быть «При нажатии на кнопку отправляется форма».

Конец ознакомительного фрагмента.

Текст предоставлен ООО «ЛитРес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на ЛитРес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.