**Тестирование**

Таблица 1— Стратегии тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Название стратегии | Сущность | Преимущества | Недостатки | Кто осуществляет |
| 1 | Метод белого ящика | Метод тестирования ПО, который предполагает, что внутренняя структура/устройство/реализация системы известны тому, кто её тестирует. | - Оптимизация кода путем нахождения скрытых ошибок  - Доступность структуры кода позволяет выбрать тип входных данных, необходимых для эффективного тестирования  - Возможность автоматизирования тест-кейсов | - Поскольку знание кода и внутренней структуры является необходимым условием, для проведения такого тестирования требуется квалифицированный тестировщик, что увеличивает стоимость  - И почти невозможно изучить каждый кусок кода, чтобы обнаружить скрытые ошибки, что может создать проблемы, приводящие к сбою приложения | Метод белого ящика часто используется на стадии, когда приложение еще не собрано воедино, но необходимо проверить каждый из его компонентов, модули процедур и подпрограмм. Компонентным тестирование чаще всего занимается программист, хорошо понимающий код или тестировщик, имеющий прекрасные знания в области программирования. |
| 2 | Метод черного ящика | Это стратегия, в которой тестирование основано исключительно на требованиях и спецификациях, при этом мы не знаем, как устроена внутри тестируемая система и работаем исключительно с внешними интерфейсами тестируемой системы или компонента. | - Простота: облегчает тестирование проектов высокого уровня и сложных приложений  - Экономия ресурсов: тестеры сосредоточены на функциональности программного обеспечения  - Тестовые случаи: Сосредоточение внимания на функциональности программного обеспечения для облегчения быстрой разработки тестовых случаев  - Обеспечивает гибкость: специальные знания программирования не требуются | - В действительности выполняется избранное число тестовых сценариев, результатом чего является ограниченный охват  - Отсутствие четкой спецификации затрудняет разработку тестовых сценариев  - Низкая эффективность | При тестировании по стратегии чёрного ящика исследуются все отдельные компоненты, например, интерфейс пользователя и UX, веб-сервер или сервер приложения, база данных, зависимости и интегрированные системы. |

Продолжение таблицы 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | Метод серого ящика | Это метод тестирования программного обеспечения для тестирования программного продукта или приложения с частичным знанием внутренней структуры приложения. Целью тестирования серого ящика является поиск и выявление дефектов, вызванных неправильной структурой кода или неправильным использованием приложений. | - Позволяет быстро выявить ошибки в функциональных спецификациях  - Тестировщику не нужна дополнительная квалификация  - Тестирование проходит «с позиции» пользователя  - Составлять тест-кейсы можно сразу после подготовки спецификации | - Связывание дефектов затруднено, когда тестирование серого выполняется для распределенных систем  - Ограниченный доступ к внутренней структуре приводит к ограниченному доступу для обхода пути кода  - Поскольку доступ к исходному коду невозможен, полное тестирование белого ящика невозможно  - Тестирование серая коробка не подходит для тестирования алгоритма  - Большинство тестовых случаев сложно спроектировать | Тестирование серого ящика - это продукт тестирования черного ящика и тестирования белого ящика. Black Box Testing означает, что тестер не знает, как работает программное обеспечение внутри. Этот тип тестирования выполняется на уровне пользователя. Таким образом, тестер проверяет, получен ли конечный результат, и не знает, правильно ли работает код в циклах и разрывах внутри. Таким образом, именно тестировщики программного обеспечения отвечают за тестирование черного ящика. |

Таблица 2— Уровни тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название уровня | Сущность | Когда осуществляется | Кто осуществляет |
| 1 | Модульное | Уровень тестирования, на котором тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. | Чтобы проверить, правильно ли написан модуль, проводят юнит-тесты, или модульное тестирование, — проверку не всего приложения, а одного модуля. Главная причина написания юнит-тестов — тестирование отдельных модулей. | Модульные тесты — это, как правило, автоматические тесты, написанные и выполняемые разработчиками программного обеспечения для обеспечения того, чтобы раздел приложения соответствовал его дизайну и вел себя должным образом. |

Продолжение таблицы 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Интеграционное | Это этап тестирования программного обеспечения, на котором отдельные программные модули объединяются и тестируются как группа. | Целью нашего тестирования является выявление багов при взаимодействии между этими программными модулями и в первую очередь направлен на проверку обмена данными между этими самими модулями. | Как правило, программный продукт состоит из нескольких программных модулей, написанных разными программистами |
| 3 | Системного | Системное тестирование означает тестирование всей системы в целом, оно выполняется после интеграционного тестирования, чтобы проверить, работает ли вся система целиком должным образом. | Системное тестирование выполняется для всей системы в контексте либо спецификаций функциональных требований (FRS), либо спецификаций системных требований (SRS), либо и того, и другого. | Поскольку системное тестирование - процесс, требующих значительных ресурсов, для его проведения часто выделяют отдельный коллектив тестировщиков, а зачастую системное тестирование выполняется организацией, которая не связана с коллективом разработчиков и тестировщиков, выполнявших работы на предыдущих этапах тестирования |
| 4 | Приемочное | Это комплексное тестирование, необходимое для определения уровня готовности системы к последующей эксплуатации. | Оно проводится после тестирования системы и является последним этапом процесса тестирования программного обеспечения. | Иногда приемочное тестирование выполняет специальная группа тестирования, включающая представителей конечных пользователей. В других случаях приемочное тестирование выполняется группой, состоящей только из представителей заказчика или уполномоченных им. |

Продолжение таблицы 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | Выходное | Осуществляется с целью проверки готовности программного обеспечения для поставки заказчику/пользователям. | Выходное тестирование осуществляется с целью проверки готовности программного обеспечения для поставки заказчику/пользователям. | Это завершающий этап тестирования, проводимый независимым тестировщиком, включающий в себя проверку на корректность инструкций по инсталляции, а также проверку комплектности документации. |

Таблица 3— Виды тестирования

|  |  |  |
| --- | --- | --- |
| № | Название вида | Сущность |
| 1 | **По запуску кода на исполнение:** | |
| 1.1. Статическое | **1.1** Оно представляет собой процесс или технику, которые выполняются для поиска потенциальных дефектов в программном обеспечении. Это также процесс обнаружения и устранения ошибок и дефектов в различных сопроводительных документах |
| 1.2 Динамическое | **1.2** Основная идея этого вида тестирования состоит в том, что проверяется реальное поведение (части) приложения.  Проще говоря, динамическое тестирование выполняется путем фактического использования приложения и определения того, работает ли функциональность так, как ожидается.  Динамическое тестирование включает в себя тестирование ПО в режиме реального времени путем предоставления входных данных и изучения результата поведения программы. Проверка осуществляется с помощью ручного или автоматического выполнения заранее подготовленного набора тестов. Оно является частью процесса валидации программного обеспечения. |

Продолжение таблицы 3.

|  |  |  |
| --- | --- | --- |
| 2 | **Функциональные виды** | |
| 2.1Функциональное | **2.1** Это вид тестирования, при котором выявляется некорректная /неправильная работа функционала программы. Проверка функций и характеристик разрабатываемого ПО.  Этот вид тестирования занимает 90% времени, отведённого на тестирование. Функциональное тестирование предполагает проверку функциональных требований: логики и бизнес-правил приложения или системы. |
| 2.2. Тестирование безопасности | **2.2** Тестирование безопасности — это процесс, направленный на выявление недостатков в механизмах безопасности информационной системы, которые защищают данные и поддерживают функциональность по назначению. Из-за логических ограничений тестирования безопасности прохождение процесса тестирования безопасности не является признаком того, что никаких недостатков не существует или что система адекватно удовлетворяет требованиям безопасности. |
| 2.3. Тестирование взаимодействия | **2.3** Тестирование взаимодействия – это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости, и интеграционное тестирование.  Программное обеспечение с хорошими характеристиками взаимодействия может быть легко интегрировано с другими системами, не требуя каких–либо серьезных модификаций. |

Продолжение таблицы 3.

|  |  |  |
| --- | --- | --- |
| 3 | **Нефункциональные виды:** | |
| 3.1 Тестирование производительности (нагрузочное  тестирование, стрессовое тестирование, тестирование  стабильности или надежности, объемное  тестирование | **3.1** Тестирование, которое проводится с целью определения, как быстро работает вычислительная система или её часть под определённой нагрузкой. Также может служить для проверки и подтверждения других атрибутов качества системы, таких как масштабируемость, надёжность и потребление ресурсов. |
| 3.2. Тестирование установки | **3.2** Тестирование установки может быть направлено на поиск ошибок, возникающих в процессе установки, которые влияют на восприятие пользователем и способность использовать установленное программное обеспечение. Существует множество событий, которые могут повлиять на установку программного обеспечения, и тестирование установки может проверять правильность установки, одновременно проверяя ряд связанных действий и событий. |
| 3.3. Тестирование удобства пользования | **3.3** Это метод оценки интерфейса со стороны удобства и эффективности его использования. Чаще всего оно проводится, когда у страницы, сайта, приложения низкая посещаемость или есть жалобы на проблемы в работе интернет-ресурса. Например, пользователи пишут в поддержку: «Я не могу оформить заказ, товар не отображается в корзине», и мы проводим тестирование, чтобы разобраться с этой жалобой. |
| 3.4. Тестирование на отказ и восстановление | **3.4** Подвид тестирования производительности, проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками ПО, отказами оборудования или проблемами связи/сети. |
| 3.5. Конфигурационное тестирование | **3.5** специальный вид тестирования, направленный на проверку работы ПО при различных аппаратных и программных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т. д.). |

Продолжение таблицы 3.

|  |  |  |
| --- | --- | --- |
| 4 | **Связанные с изменениями** | |
| 4.1. Дымовое тестирование | **4.1** В тестировании программного обеспечения означает минимальный набор тестов на явные ошибки. Дымовой тест обычно выполняется программистом; не проходившую этот тест программу не имеет смысла отдавать на более глубокое тестирование. |
| 4.2. Регрессионное тестирование | **4.2** Это повторный запуск функциональных и нефункциональных тестов для обеспечения того, чтобы ранее разработанное и протестированное программное обеспечение по-прежнему работало после изменения. Если нет, то это называется регрессией. |
| 4.3. Тестирование сборки | **4.3** Это тестирование, направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования. По своим целям является аналогом Дымового Тестирования, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию. Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии. |
| 4.4. Санитарное тестирование или проверка  согласованности/исправности | **4.4** Это узконаправленное тестирование достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям. Является подмножеством регрессионного тестирования. Используется для определения работоспособности определенной части приложения после изменений, произведенных в ней или окружающей среде. Обычно выполняется вручную. |

Продолжение таблицы 3.

|  |  |  |
| --- | --- | --- |
| 5 | **По степени автоматизации:** | |
| 5.1. Ручное тестирование | **5.1** часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно производится тестировщиком без использования программных средств, для проверки программы или сайта путём моделирования действий пользователя. В роли тестировщиков могут выступать и обычные пользователи, сообщая разработчикам о найденных ошибках. |
| 5.2. Автоматизированное тестирование | 5.2 Процесс тестирования программного обеспечения, при котором основные функции и шаги теста, такие как запуск, инициализация, выполнение, анализ и выдача результата, производятся автоматически с помощью инструментов для автоматизированного тестирования. |
| 6 | **По времени проведения:** | |
| 6.1. Альфа-тестирование | **6.1** Это вид приемочного тестирования, которое обычно проводится на поздней стадии разработки продукта и включает имитацию реального использования продукта штатными разработчиками либо командой тестировщиков.  Обычно альфа тестирование заключается в систематической проверке всех функций программы с использованием техник тестирования «белого ящика» и «черного ящика»  Альфа-тестирование является методологией оценки качества и стабильности тестируемого продукта в тестовой среде. |
| 6.2. Бета-тестирование | **6.2** Эксплуатационное тестирование потенциальными и/или существующими клиентами/заказчиками на внешней стороне, никак не связанными с разработчиками, с целью определения действительно ли компонент или система удовлетворяет требованиям клиента/заказчика и вписывается в бизнес-процессы. |

Таблица 4— Тесты на использование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название теста | Действия | Исходная информация | Ожидаемая информация |
| 1 | 2 | 3 | 4 | 5 |
| Т1 | Переход по кнопкам меню | Нажатие на одну из кнопок | - | Выполнение соответствующего кнопке действия |
| Т2 | Ходьба | Ходьба при помощи стрелочек на клавиатуре | - | Перемещение по локации |
| Т3 | Меню персонажа | Открытие меню персонажа путем нажатия кнопки Esc на клавиатуре | - | Открытие меню |
| Т4 | Бой | Переход в режим боя при контакте персонажа с противником | - | Вступление в бой с противником |
| Т5 | Использование навыков | Использование навыков в бою путем выбора навыка при помощи стрелочек | - | Использование навыка |
| Т6 | Покупка товара | Покупка товара в игровом магазине путем контакта с продавцом NPC | - | Покупка товара после контакта с продавцом NPC |
| Т7 | Получение предмета | Получение случайного предмета после победы над врагом | - | Получение предмета |
| Т8 | Получение золота | Получение золота после победы над противником | - | Получение золота |
| Т9 | Повышение уровня | Повышение уровня после победы над определенным количеством противников | - | Повышение уровня |

**Отчёт о результатах тестирования**

Таблица 7 – результаты тестов

|  |  |
| --- | --- |
| № | Статус |
| Т1 | Выполнено успешно |
| Т2 | Выполнено успешно |
| Т3 | Выполнено успешно |
| Т4 | Выполнено успешно |
| Т5 | Выполнено успешно |
| Т6 | Выполнено успешно |
| Т7 | Выполнено успешно |
| Т8 | Выполнено успешно |
| Т9 | Выполнено успешно |