# Лабораторная работа №25 Тестирование и отладка программного продукта

**Цель работы:**

1. Изучить виды, стратегии и уровни тестирования.
2. Научиться составлять тесты на использование для проведения функционального тестирования
3. Научиться оформлять отчет о результатах тестирования.

# ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. **Изучить краткие теоретические сведения по теме «Тестирование и отладка программного продукта».**

**Выполнить задание №1\_Изучение стратегий, уровней и видов тестирования.**

Заполнить следующие таблицы:

# Таблица 1-Стратегии тестирования (По доступу к коду и архитектуре приложения)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название стратегии** | **Сущность**  **(характеристика стратегии)** | **Преимущества** | **Недостатки** | **Кто осуществляет** |
| **1** | **Метод белого**  **(прозрачного) ящика** | это метод тестирования, который проверяет внутреннее функционирование системы. В этом методе тестирование основано на охвате операторов кода, ветвей, путей или условий. Тестирование белого ящика считается тестированием низкого уровня. | очень легко выяснить, какой тип ввода / данных может помочь в эффективном тестировании приложения.  помогает оптимизировать код | для проведения такого тестирования требуется квалифицированный тестировщик  почти невозможно изучить каждый кусок кода, чтобы обнаружить скрытые ошибки | Осуществляется разработчиками |
| **2** | **Метод черного**  **ящика** | это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы. *Метод тестирования «черного ящика»* – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство. | Тестирование методом «черного ящика» позволяет найти ошибки, которые невозможно обнаружить методом «белого ящика»  Тестировщику не нужна дополнительная квалификация  Составлять тест-кейсы можно сразу после подготовки спецификации | Можно протестировать только небольшое количество возможных вводных (входящих) значений  Тесты могут быть избыточными, если разработчик уже проверил данную функциональность | Тестировщиками |
| **3** | **Метод серого ящика** | специальный метод тестирования программного обеспечения с неполным знанием его внутреннего устройства. Чтобы выполнить подобный вид тестов, не нужно иметь доступ к исходному коду ПО. Все тесты создаются на базе простого знания алгоритмов, архитектуры и иных высокоуровневых характеристик поведения продукта. | * QA-специалист может создавать и применять более сложные тестовые сценарии. * Данная проверка позволяет программисту заручиться достаточным количеством времени для исправления багов.   Имеет некоторые особенности черного и белого ящика | * Анализ программного кода ограничен, так как доступа к исходному коду у тестировщика нет. * Нет времени тестировать все потоки ввода и вывода информации, так как это займет очень много времени. * Возможна ситуация, когда тестировщики могут стать лишними (когда не только QA-специалист, но и программист проверяет свой код с помощью юнит-тестов). | Тестировщик |

**Таблица 2-Уровни тестирования**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название уровня** | **Сущность**  **(характеристика уровня)** | **Когда осуществляется** | **Кто осуществляет** |
| **1** | **Модульное (компонентное)** | это **подготовка автоматизированных тестов** до начала основного кодирования (разработки) программного обеспечения. При этом подходе создаются и интегрируются небольшие куски кода, напротив которых запускаются тесты, написанные до начала кодирования. Разработка ведется до тех пор пока все тесты не будут успешно пройдены. | При разработке кода | разработчик (редко тестировщик) |
| **2** | **Интеграционное** | Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы | При разработке кода | разработчик и тестировщик |
| **3** | **Системное** | Основной задачей системного тестирования является **проверка**[**как функциональных, так и не функциональных требований**](http://www.protesting.ru/testing/testtypes.html)**в системе в целом**. При этом выявляются [**дефекты**](http://www.protesting.ru/testing/bugreport.html), такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д. Для минимизации рисков, связанных с особенностями поведения системы в той или иной среде, **во время тестирования рекомендуется использовать окружение максимально приближенное к тому, на которое будет установлен продукт после выдачи**. | При разработке кода | тестировщик |
| **4** | **Приемочное** | Приемочное тестирование **выполняется на основании набора типичных**[**тестовых случаев**](http://www.protesting.ru/testing/testcase.html)**и сценариев**, разработанных на основании требований к данному приложению. **Решение о проведении приемочного тестирования** принимается, когда:   * продукт достиг необходимого уровня качества; * заказчик ознакомлен с **Планом Приемочных Работ** (**Product Acceptance Plan**) или иным документом, где описан набор действий, связанных с проведением приемочного тестирования, дата проведения, ответственные и т.д. | На этапе выпуска ПО | заказчик / клиент / бизнес-аналитик / product owner и тестировщик |
| **5** | **Выходное** | *Выходное тестирование* осуществляется с целью проверки готовности программного обеспечения для поставки заказчику/пользовате- лям. Это завершающий этап тестирования, проводимый независимым тестировщиком, включающий в себя проверку на корректность инструкций по инсталляции, а также проверку комплектности документации. Ошибки, выявленные при выходном тестировании, заносятся в базу данных проекта. | На этапе выпуска ПО | Тестировщик |

**Таблица 3-Виды тестирования**

|  |  |  |
| --- | --- | --- |
| **№** | **Название вида** | **Сущность (характеристика вида)** |
| **1** | **По запуску кода на исполнение:**  1.1.Статическое  1.2 Динамическое | Статическое тестирование – тип тестирования, который предполагает, что программный код во время тестирования не будет выполняться  Динамическое тестирование – тип тестирования, который предполагает запуск программного кода. Таким образом, анализируется поведение программы во время ее работы. |
| **2** | **Функциональные виды**  2.1Функциональное   * 1. Тестирование безопасности   2. Тестирование взаимодействия | Функциональное тестирование в основном включает тестирование черного ящика и не касается исходного кода приложения. Это тестирование проверяет пользовательский интерфейс, API, базу данных, безопасность, связь клиент / сервер и другие функциональные возможности тестируемого приложения  Что входит в тестирование безопасности?  **Тестирование безопасности** проверяет фактическую реакцию защитных механизмов, встроенных в систему, на проникновение.  **Тестирование взаимодействия** (**Interoperability Testing**) – это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости (compatibility testing) и интеграционное тестирование |
| **3** | **Нефункциональные виды:**  3.1 Тестирование производительности (нагрузочное тестирование, стрессовое тестирование, тестирование  стабильности или надежности, объемное тестирование   * 1. [Тестирование установки](http://www.protesting.ru/testing/types/installation.html)   2. [Тестирование удобства пользования](http://www.protesting.ru/testing/types/usability.html)   3. [Тестирование на отказ и восстановление](http://www.protesting.ru/testing/types/failover.html)   4. [Конфигурационное тестирование](http://www.protesting.ru/testing/types/configuration.html) | **Тестирование производительности в инженерии программного обеспечения — тестирование, которое проводится с целью определения, как быстро работает вычислительная система или её часть под определённой нагрузкой.**  **Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.**  **Тестирование удобства пользования** - это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.  **Тестирование на отказ и восстановление** (**Failover and Recovery Testing**) проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказ сети). **Целью** данного **вида тестирования** является проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта.  Конфигурационное тестирование (Configuration Testing) — специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т. д.) |
| **4** | **Связанные с изменениями**   * 1. [Дымовое тестирование](http://www.protesting.ru/testing/types/smoke.html)   2. [Регрессионное тестирование](http://www.protesting.ru/testing/types/regression.html)   3. [Тестирование сборки](http://www.protesting.ru/testing/types/bvt.html)   4. [Санитарное тестирование или проверка согласованности/исправности](http://www.protesting.ru/testing/types/sanity.html) | **Дымовой тест — в тестировании программного обеспечения означает минимальный набор тестов на явные ошибки.**  **Регрессио́нное тести́рование — собирательное название для всех видов тестирования программного обеспечения, направленных на обнаружение ошибок в уже протестированных участках исходного кода.**  **Тестирование сборки**- это набор тестов, запускаемых при каждой новой сборке, чтобы убедиться, что сборка тестируема, прежде чем она будет передана группе тестирования для дальнейшего тестирования  **Санитарное тестирование** - 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. |
| **5** | **По степени автоматизации:**   * 1. Ручное тестирование   2. Автоматизированное тестирование | Ручное тестирование программного обеспечения – это процесс проверки ПО, выполняемый специалистами вручную. Это значит, что для его проведения не используются какие-либо специальные автоматизированные средства.  **Автоматизированное тестирование программного обеспечения** — часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс. |
| **6** | **По времени проведения:**   * 1. Альфа-тестирование   2. Бета-тестирование | Что такое альфа тестирование?  **Альфа**-**тестирование** – **это** ручное **тестирование** потенциальными пользователями, заказчиками или независимой командой тестирования на стенде разработки.    Бе́та-тести́рование — интенсивное использование почти готовой версии продукта с целью выявления максимального числа ошибок в его работе для их последующего устранения перед окончательным выходом продукта на рынок, к массовому потребителю. |

1. **Выполнить практическую часть работы**

**Задание №2\_ Научиться составлять тесты на использование и по ним проводить**

**тестирование разработанного прототипа методом «белого ящика»**

* 1. Для этого необходимо придумать тесты, предназначенные в дальнейшем для проведения **функционального тестирования программного продукта (тесты разработать для программного продукта в соответствии с вашим индивидуальным заданием, по которому разрабатывалась постановка задачи).**

Тесты оформить в виде таблицы следующего вида:

Таблица 1-Тесты на использование

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Наименование функции** | **Описание тестового сценария с исходными данными** | **Ожидаемый результат** |
| 1 | Проверка пункта меню «Печать» | 1. На главной форме программы выбираем пункт меню   «Файл»/ «Печать».   1. Выбираем настройки параметров печати. 2. Нажимаем на кнопку «ОК» | Документ, распечатанный принтером. |
| 2 | Расчет значения выражения по формуле V=a\*b\*c | 1. В окне «Расчет объема тела» вводим значения для a=3, b=20, c=10. 2. Нажимаем на кнопку   «Рассчитать».   1. В поле V= смотрим результат расчета. | V=180 |

При разработке тестов предусмотреть ситуации, когда программа может сработать со сбоями и ошибками (например, ошибки в вычислениях, некорректные данные и прочее). Для этих случаев также должны быть разработаны тесты.

* 1. Протестировать разработанный при выполнении лабораторной работы №24 прототип с помощью разработанных тестов.
  2. Оформить отчет о результатах тестирования.